Pakiet edukacyjny

O projekcie

Na jakość szkolnictwa zawodowego i jego absolwentów znacząco wpływa przygotowanie nauczycieli, w tym w szczególności ich wiedza o funkcjonowaniu firm i oczekiwaniach pracodawców wobec absolwentów. Szybkie zmiany w branży IT, która wyróżnia się największą dynamiką rozwoju oraz zatrudnienia, postęp społeczno-gospodarczo-technologiczny sprawiają, że nauczyciele powinni stale doskonalić swoje umiejętności, aby do ucznia trafiała aktualna wiedza i niezbędne umiejętności, zgodne z potrzebami rynku pracy.
Projekt „Mistrz Kształcenia Zawodowego" zakładał przygotowanie i wdrożenie programu doskonalenia zawodowego w formie staży w przedsiębiorstwach dla nauczycieli branży IT. Staż w firmie, pokazy najnowszych technologii, zajęcia praktyczne, zapoznanie się ze stanowiskami pracy i procedurami, pozwoliły na praktyczne poznanie środowiska pracy. Staże odbyły się w przedsiębiorstwach branży IT oraz firmach start-up, działających na terenie Krakowskiego Parku Technologicznego i Technoparku w Gliwicach, które zaprezentowały stażystom najnowsze technologie stosowane w swojej działalności.

Głównym celem projektu było opracowanie i pilotażowe wdrożenie programu doskonalenia zawodowego w formie staży w przedsiębiorstwach dla nauczycieli kształcenia w branży IT w województwie małopolskim i śląskim poprzez:
- opracowanie modelowego programu doskonalenia zawodowego poprzez staże nauczycieli,
- pilotażowe wdrożenie i przeprowadzenie modelowego programu staży,
- podniesienie wiedzy merytorycznej i umiejętności praktycznych i dydaktycznych nauczycieli kształcących w branży IT,
- zwiększenie wiedzy na temat funkcjonowania i wymagań stawianych pracownikowi w branży IT,
- upowszechnienie rezultatów wśród szkół i przedsiębiorstw z 5 województw.

Szkoła zawodowa w środowisku pracodawców

Konieczność współpracy szkolnictwa zawodowego z pracodawcami i instytucjami rynku pracy /IRP/

Każda szkoła kreując swoją strategię rozwoju /program rozwoju/ powinna precyzyjnie określić swoje cele działania, zdiagnozować swoje możliwości wewnętrzne /potencjał jakościowy/ oraz możliwości wsparcia zewnętrznego. W przypadku szkoły zawodowej szczególne znaczenie ma rzeczywiste zakorzenienie jej w środowisku gospodarczym /środowisku pracodawców/ oraz wśród otaczających ją instytucji rynku pracy. Współpraca szkoły z tymi podmiotami może odbywać się w takich obszarach, jak:

- kreowanie i doskonalenie oferty edukacyjnej szkoły
- pozyskiwanie i wykorzystywanie informacji o sytuacji na rynku pracy oraz o pojawiających się tendencjach zmian
- kształcenie zawodowe, w tym praktyczna nauka zawodu
- kształcenie w zakresie przedsiębiorczości i edukacji dla rynku pracy
- doskonalenie i doradztwo dla nauczycieli przedmiotów zawodowych i praktycznej nauki zawodu, a także dla nauczycieli przedsiębiorczości, doradców zawodowych oraz pedagogów szkolnych
- pozyskiwanie przez szkołę sprzętu i pomocy dydaktycznych związanych z kształceniem zawodowym.

Dlatego też pragnąc dobrze realizować fundamentalny swój cel, czyli właściwe przygotowanie uczniów do wykonywania zadań na otaczającym ją rynku pracy szkoła zawodowa powinna działać w pewnego rodzaju symbiozie z podmiotami gospodarczymi i IRP. Trzeba jednak pamiętać, że w Polsce brak jest unormowań prawnych i mechanizmów sprzyjających powstawaniu takiej sytuacji. Stąd też współpraca szkół zawodowych z pracodawcami oraz IRP ma najczęściej epizodyczny charakter i związana jest bardziej ze spektakularnymi przedsięwzięciami niż z bieżącą, stałą współpracą. Czynnikiem nie sprzyjającym tej współpracy jest często występująca na współczesnym rynku pracy przewaga podaży pracowników nad popytem na ich pracę /chociaż w przypadku branży IT nie mamy generalnie do czynienia z taką sytuacją/.

Pracodawcy podejmują współpracę ze szkołą w sytuacji, gdy dostrzegają bezpośrednie korzyści z niej, zarówno w perspektywie krótko, jak też długoterminowej. Owe korzyści to oczywiście dobrze przygotowani i znający specyfikę oraz potrzeby firmy przyszli pracownicy, ale także możliwość stworzenia zaplecza szkoleniowego dla firmy, przydatnego dla doskonalenia umiejętności i kwalifikacji jej pracowników. Dotychczas polskie szkolnictwo zawodowe

praktycznie nie potrafi podjąć się tego zadania, pomimo tego, że dobrze funkcjonujące firmy są otwarte na podjęcie współdziałania ze szkołami pod warunkiem przedstawienia im atrakcyjnej merytorycznie i finansowo oferty szkoleniowej. W istniejącej sytuacji inicjatywa dotycząca nawiązania współpracy najczęściej musi wychodzić ze środowiska szkolnego. Trzeba także pamiętać, że jej brak powoduje absolutne ograniczenie możliwości sukcesu absolwentów na rynku pracy. W przypadku polskich szkół zawodowych skuteczność w nawiązywaniu kontaktów z pracodawcami ma szczególne znaczenie, w kontekście ich słabości w kształceniu zawodowym uczniów. Okazuje się, że swoistym paradoksem polskiego szkolnictwa zawodowego jest konieczność nauczania absolwentów niezbędnych w danym zawodzie umiejętności po podjęciu przez nich pracy. Przezwyciężenie tej słabości jest możliwe tylko poprzez urealnienie oferty edukacyjnej szkół, ale aby do tego doszło musi ona zostać podporządkowana rzeczywistym potrzebom rynku pracy. Trzeba pamiętać, że nie wystarczy, aby szkoła podjęła kształcenie w kierunku, na który istnieje zapotrzebowanie na rynku pracy; owo kształcenie musi być prowadzone zgodnie z wymogami obowiązującej w danej branży technologii, z wykorzystaniem narzędzi i metod stosowanych w działających na rynku firmach. W przypadku branży IT, w której mamy do czynienia z dynamicznie zmieniającymi się technologiami i wymaganiami, które muszą spełniać pracownicy bezpośrednia, bieżąca współpraca szkół z firmami jest wręcz koniecznością. Nie można także nie zauważyć, iż kłopoty szkół z właściwym przygotowaniem absolwentów wynikają nie tylko z ich braków w wyposażeniu w sprzęt i potrzebne pomoce dydaktyczne, ale często z nieznajomości realiów współczesnej gospodarki i rynku pracy przez nauczycieli. Większość z osób uczących przedmiotów zawodowych i prowadzących zajęcia z zakresu przedsiębiorczości czy edukacji dla rynku pracy nie ma żadnego doświadczenia związanego z wykonywaniem zadań zawodowych poza szkołą. Równocześnie trzeba pamiętać, iż w przypadku branży IT nawet najlepiej wykształceni nauczyciele bez bieżącego kontaktu z firmami bardzo szybko utracą szansę na skuteczną pracę. Stąd też wszelkie kontakty tych nauczycieli z firmami mają ogromne znaczenie dla jakości ich pracy. Niestety najczęściej mają one epizodyczny charakter, a najbardziej przydatne dla nauczycieli byłyby regularne staże odbywane przez nich w firmach, w których wykorzystywane są najnowsze technologie. Uczestniczący w owych stażach nauczyciele mogliby uzyskiwać wystawiane przez firmę certyfikaty umiejętności przydatnych w branży, w której działa owa firma. Równocześnie certyfikowani nauczyciele byliby pewnego rodzaju mężami zaufania firmy w szkole.

Elastyczność działania warunkiem sukcesu szkoły

Nie można nie zauważyć, iż na dynamicznie zmieniającym się rynku pracy szansa na sukces szkoły zawodowej wzrasta wraz ze stopniem i zakresem jej elastyczności w kształtowaniu swojej oferty edukacyjnej. Tymczasem prowadzone przez jednostki samorządu terytorialnego /JST/ polskie szkoły publiczne wtłoczone są w biurokratyczno – etatystyczny model działania, który

jest niezwykle mało podatny na zmiany. Owe zmiany oferty edukacyjnej bardzo często wymagają zmian wśród zatrudnionych w szkole nauczycieli, a ich przeprowadzenie jest niebywale utrudnione przez regulującą tę kwestię w prowadzonych przez samorząd szkołach publicznych Kartę nauczyciela. Dlatego też zdecydowanie bardziej przystają do gospodarki rynkowej szkoły zawodowe, w których zatrudnienie nauczycieli oparte jest na przepisach Kodeksu pracy, czyli szkoły publiczne prowadzone przez podmioty obywatelskie /stowarzyszenia, fundacje bądź osoby fizyczne/ oraz oczywiście szkoły niepubliczne. Pomimo niechęci nauczycieli i ich związków zawodowych właśnie one mogą okazać się w przyszłości głównym składnikiem oświaty publicznej. Zresztą już teraz samorządy mogą podejmować działania przekształcające prowadzone przez siebie placówki oświatowe w jednostki funkcjonujące w takiej formule. Aby tak się stało placówka samorządowa musi zostać przez JST zlikwidowana /po nowelizacji ustawy o systemie oświaty z 19.03.2009 r. nie jest już potrzebna zgoda kuratora oświaty na taką operację/, a następnie na jej bazie powołana publiczna placówka prowadzona przez podmiot obywatelski. Po takim przekształceniu placówka oświatowa /szkoła, centrum kształcenia praktycznego/ może działać, jak quasi rynkowa firma edukacyjna dopasowując swoją ofertę i działania do potrzeb uczniów, wynikających z sytuacji na rynku pracy.

Organizacja pracy szkoły

W organizacji współczesnej szkoły zawodowej istotne jest precyzyjne przyporządkowanie zadań związanych ze współpracą z pracodawcami, jak też z IRP. Za to zadanie może odpowiadać:

- bezpośrednio jeden z zastępców dyrektora /jest to sytuacja optymalna/
- pedagog szkolny /o ile jest do tego odpowiednio przygotowany/
- doradca zawodowy /jeżeli oczywiście pracuje w danej szkole/
- osoba odpowiedzialna w szkole za kształcenie zawodowe /np. przewodniczący zespołu kształcenia zawodowego/.

Optymalnym rozwiązaniem byłoby posiadanie przez taka osobę branżowej certyfikacji wystawionej przez firmę. Jeżeli ową osobą nie jest zastępca dyrektora to powinien on nadzorować wykonywanie tego zadania przez inną osobę. Ów zastępca dyrektora powinien mieć wykształcenie zgodne z profilem kształcenia szkoły oraz byłoby wskazanym, aby miał doświadczenie bezpośredniej pracy w firmie, zgodnie ze swoim przygotowaniem zawodowym. Do zadań osób odpowiedzialnych za to zadanie należeć powinny wszelkie sprawy związane z monitoringiem rynku pracy, jak też z losami absolwentów szkoły, szczególnie w kontekście radzenia sobie na rynku pracy oraz skuteczności w wykonywaniu zadań zawodowych. Efekty tego monitoringu winny mieć kluczowe znaczenie dla kreowania polityki rozwoju szkoły. To właśnie na ich podstawie powinny pojawiać się wszelkie propozycje zmian oferty kształcenia szkoły: nowe kierunki kształcenia, zaprzestanie kształcenia w prowadzonych kierunkach,

czy też zmiany w obrębie prowadzonych przez szkołę kierunków, związane ze zmianami wykorzystywanych w nich technologii, czy też metod. Analiza losu absolwentów winna także służyć do doskonalenia nauczania w zakresie kształtowania wśród uczniów kompetencji kluczowych.

Trzeba dodać, iż dla podejmowania koniecznych w jej ofercie zmian szkoła winna mieć w obszarze szkolnego zestawu programów nauczania konstrukcję warstwową, czyli taką w której jej programy nauczania można umieścić w trzech warstwach:

- podstawowej /stabilnej/ o charakterze ogólnokształcącym
- częściowo zmiennej: ogólnozawodowej oraz zawierającej przedmioty wspierające kształcenie zawodowe /w przypadku szkół kształcących w branży IT – przedmioty ścisłe, szczególnie matematyka i fizyka/
- w pełni elastycznej: kierunkowego kształcenia zawodowego oraz kształcenia praktycznego /zmieniającej się wraz z wprowadzaniem nowych kierunków kształcenia/.

Każda decyzja dotycząca zmian kierunków kształcenia powinna być poprzedzona nie tylko analizami bieżącej sytuacji na rynku pracy, ale także przewidywanymi zmianami na nim. Przewidywanie owych zmian należy do najtrudniejszych zadań specjalistów rynku pracy, jak też osób kierujących przedsiębiorstwami. Jednak brak tej umiejętności najczęściej kończy się katastrofą firmy. Warto zauważyć, że również w przypadku szkół ma smutne konsekwencje w postaci bezrobocia absolwentów, które winno skutkować spadkiem zainteresowania ofertą edukacyjną szkoły, a w konsekwencji /jeżeli nie podejmie ona działań restrukturyzacyjnych/ jej likwidacją.

Szkoła a informacje z rynku pracy

Dobrze zorganizowana i działająca szkoła będąca „blisko" pracodawców i IRP to placówka, w której na bieżąco dokonuje się wielowymiarowych analiz sytuacji na rynku pracy w kontekście wykonywanych przez nią zadań. Najlepszym miejscem do tego są spotkania zespołu kształcenia zawodowego, którego członkowie /z osobą odpowiedzialną za współpracę z pracodawcami i IRP/ wraz z przedstawicielami IRP i pracodawców winni czynić to przynajmniej dwa razy w semestrze. Oprócz tego przynajmniej raz w semestrze powinna na ten temat debatować cała rada pedagogiczna. Dla wykonywania przydatnych szkole analiz fundamentalne znaczenie ma pozyskiwanie z IRP i umiejętne korzystanie z informacji dotyczących rynku pracy. Trzeba jednak pamiętać, że jakość posiadanych przez konkretną IRP informacji może odbiegać od oczekiwań i potrzeb szkoły, stąd też powinna ona starać się współpracować z różnymi IRP. Optymalnie działająca szkoła powinna pozyskiwać informacje odnoszące się zarówno do lokalnego, jak też regionalnego i krajowego, ale także europejskiego rynku pracy. Tylko synteza takich

informacji pozwoli na kreowanie oferty szkoły w szerokim zakresie przygotowującej uczniów do przyszłej kariery zawodowej. Dla jakości i efektywności współpracy z IRP istotne znaczenie ma stopień zaangażowania tych instytucji we współdziałanie z placówkami oświatowymi. Owej współpracy dobrze służy wydzielenie w strukturze IRP osoby /osób, komórki/ zajmującej się kwestią korelowania kształcenia zawodowego z potrzebami rynku pracy.

Dla szkół bardzo ważne jest m.in. pozyskiwanie informacji o zarejestrowanych w danym urzędzie pracy bezrobotnych absolwentach. Stąd też IRP powinny starać się gromadzić informację o drodze edukacyjnej osób poszukujących zatrudnienia. Równocześnie urzędy pracy winny być zobowiązane do publicznego przedstawiania raportów o zakresie nieskuteczności kształcenia zawodowego poszczególnych szkół poprzez ukazywanie skali bezrobocia ich absolwentów. Taki materiał byłby przestrogą dla absolwentów gimnazjów przed podejmowaniem nauki w szkołach „specjalizujących się" w kształceniu bezrobotnych. Oczywiście lepszym rozwiązaniem byłyby raporty ukazujące skuteczność absolwentów w poszukiwaniu miejsc pracy oraz w wykonywaniu zadań zawodowych, jednak póki co nie mamy w Polsce dobrych narzędzi śledzenia losów absolwentów, natomiast posiadamy informację o zarejestrowanych bezrobotnych. W ramach współpracy szkół zawodowych z pracodawcami i IRP oraz z pomocą kuratoriów oświaty powinny zostać podjęte działania pozwalające stworzyć narzędzia śledzenia losów absolwentów.

Specjaliści z firm i IRP w szkole

Poszukując łączników z przedsiębiorstwami szkoła powinna starać się zatrudniać w niepełnym wymiarze pracujących w nich specjalistów, jako nauczycieli przedmiotów zawodowych, zarówno teoretycznych, jak też kształcenia praktycznego. Ich zajęcia z młodzieżą byłyby osadzone w kontekście rzeczywistych wymagań technologicznych oraz przesycone treściami i problemami mającymi odwzorowanie w autentycznie działającej firmie. Równocześnie stawaliby się oni ambasadorami szkoły w firmie, lobując na rzecz rozwijania współpracy. Z całą pewnością mogliby oni ułatwiać szkole organizację praktyk zawodowych na terenie firmy, gwarantując osobom za nią odpowiedzialnym posiadanie przez uczniów umiejętności pozwalających im na wykonywanie czynności przydatnych w pracy przedsiębiorstwa.

Dla szkoły ogromne znaczenie może mieć włączenie owych specjalistów do systemu wewnątrzszkolnego doskonalenia nauczycieli. W naszym systemie doskonalenia nauczycieli kwestie związane z kształceniem zawodowym mają marginalny charakter oraz niewielki związek z potrzebami. Stąd też pozyskanie do niego osób mogących wprowadzić nauczycieli w rzeczywiste problemy współczesnej firmy jest jednym z elementów kreowania dobrej szkoły zawodowej, szczególnie w kontekście bardzo częstego zatrudniania na stanowiskach

nauczycieli kształcenia zawodowego osób nie mających żadnego doświadczenia w działaniu na pozaoświatowym rynku pracy. Atutem takiej formy doskonalenia nauczycieli mogłoby być także prowadzenie zajęć dla nauczycieli w firmach, w oparciu o wykorzystywany w nich sprzęt. Zresztą współpraca jednostek oświatowych z przedsiębiorstwami i IRP w zakresie doskonalenia nauczycieli nie powinna ograniczać się tylko do szkół, ale winna dotyczyć także ośrodków doskonalenia nauczycieli. Może się również okazać, że w niektórych szkołach zawodowych powstaną ośrodki doskonalenia i doradztwa dla nauczycieli przedmiotów zawodowych i przedmiotów związanych z kreowaniem wśród uczniów postaw osób przedsiębiorczych. Trzeba zauważyć, iż obecnie w ofercie ośrodków doskonalenia nauczycieli /również tych prowadzonych przez samorząd województwa/ kształcenie zawodowe zajmuje marginalną pozycję, również niezwykle ważne dla szkół branży IT doskonalenie nauczycieli matematyki i fizyki w kontekście zastosowań tych przedmiotów w obszarze kształcenia informatycznego jest praktycznie w niej nieobecne.

W codziennej pracy szkoły przedstawiciele firm oraz IRP mogą także wesprzeć wychowawcę klasy w prowadzeniu zajęć w ramach tzw. godzin do dyspozycji wychowawcy. W ich trakcie mogliby oni przedstawiać młodzieży aktualne problemy rynku pracy, zarówno w wymiarze lokalnym, jak też krajowym, a także problemy branży. w której szkoła kształci. W ramach tych zajęć mogliby oni także poprowadzić zajęcia poświęcone etosowi pracy oraz miejscu młodego pracownika na rynku, z ukazaniem mu możliwości zrobienia w przyszłości pięknej kariery zawodowej.

Praktyczna nauka zawodu /pnz/

Ważnym elementem współdziałania szkół zawodowych z firmami jest obszar praktycznej nauki zawodu. Optymalnym rozwiązaniem jest sytuacja, w której szkoła staje się podwykonawcą dla firmy. Jednak dla jej zaistnienia konieczny jest wysoki stopień profesjonalizacji zadań wykonywanych w trakcie praktycznej nauki zawodu. Oczywiście uczniowie zaangażowani w ten typ zadań muszą mieć za sobą wstępne przygotowanie do wykonywania praktycznych zadań zawodowych, a ich nauczyciele muszą być autentycznymi specjalistami w danej dziedzinie, sprawnie radzącymi sobie z wyzwaniami stosowanych w danym zawodzie technologii. Ten typ współpracy może zaistnieć tylko w przypadku dużego zaufania osób odpowiedzialnych za firmę do kompetencji nauczycieli praktycznej nauki zawodu /dlatego optymalnym rozwiązaniem jest prowadzenie zajęć pnz przez osoby już pracujące w firmie/ oraz posiadania przez nich właściwych warunków do pracy. W przypadku zaistnienia i rozwoju doskonalenia nauczycieli kształcenia zawodowego poprzez staże w firmach, mogłyby one certyfikować tych nauczycieli do wykonywania z uczniami przydatnych dla nich zadań. Wydaje się to sposobem łatwiejszym niż pozyskiwanie pracowników firm do pracy w szkole, chociaż dotychczas nie występującym.

Jednak wszelkie podejmowane w tej materii próby /głównie w ramach projektów, takich jak Mistrz kształcenia zawodowego/ przynoszą bardzo zachęcające rezultaty.
Trzeba także wspomnieć o ważnej możliwości rozszerzania oferty kształcenia zawodowego w szkole o szkolenia pozwalające uczniom uzyskiwać dodatkowe uprawnienia zawodowe, co wzmacnia ich pozycję na rynku pracy. Owe szkolenia powinny być prowadzone przez specjalistów danej dziedziny zatrudnionych na co dzień w firmach współpracujących ze szkołą lub certyfikowanych przez firmę nauczycieli. Warto zauważyć, że rozwijając współpracę z firmami w tym zakresie szkoła w naturalny sposób może rozszerzyć swoją ofertę edukacyjną o kształcenie ustawiczne, co może poprawić jej pozycję na rynku edukacyjnym. Dodatkową, niezwykle ważną zaletą włączenia kształcenia ustawicznego do szkolnej oferty jest możliwość poprawy stanu szkolnych finansów, gdyż zajęcia dla dorosłych słuchaczy prowadzone są w formule komercyjnej. Równocześnie funkcjonują one na edukacyjnym rynku, wobec czego szczególnie istotna jest jakość prezentowanej przez szkołę oferty. Właśnie pozyskanie przez placówkę zatrudnionych na co dzień w firmie specjalistów lub posiadanie w gronie prowadzących zajęcia kształcenia zawodowego nauczycieli certyfikowanych przez firmy może dać szkole przewagę konkurencyjną nad innymi jednostkami.

Warto zauważyć, iż w przypadku zaistnienia takiej współpracy kolejnym krokiem może stać się podejmowanie okresowej pracy w firmie przez uczniów, co może być wstępem do ich zatrudnienia w przedsiębiorstwie po ukończeniu szkoły. Cennym rozwiązaniem może być także zakładanie przez uczniów pod opieką nauczyciela mikrofirm wykonujących zadania na rzecz przedsiębiorstwa. Pracując w takiej mikrofirmie, czy tez prowadząc ją młodzi ludzie uczą się działania w rzeczywistości gospodarczej, co stanowi najlepszą lekcję przedsiębiorczości. Oczywiście dla zaistnienia takiego przedsięwzięcia ogromne znaczenie ma inicjatywa nauczyciela i jego wsparcie uczniowskich wysiłków.

Równocześnie przy bliskiej współpracy firmy ze szkołą wręcz naturalne staje się przekazywanie przez nią do szkoły sprzętu wycofywanego z użytkowania w przedsiębiorstwie. Poza tym dużo łatwiejsze jest także korzystanie przez uczniów ze sprzętu posiadanego i użytkowanego przez firmę.

Wnioski

1. Staże nauczycieli, w uczestniczących w projekcie firmach, okazały się dla nich cennym doświadczeniem, szczególnie w kontekście uaktualnienia i rozszerzenia ich wiedzy specjalistycznej związanej z prowadzonymi przez nich w szkole zajęciami.
2. Staże pozwoliły ich uczestnikom zapoznać się z wymaganiami rynku pracy w branży IT, co dało im szansę na dokonanie analizy SWOT ich szkół w tym kontekście. Dzięki temu mogą oni

stać się agentami zmiany w swojej szkole, kreując niezbędne dla jej rozwoju modyfikacje w jej dotychczasowej pracy.

3. Mankamentem realizowanego projektu był krótki czas trwania staży, co uniemożliwiało ich uczestnikom dogłębne rozpoznanie ważnej dla ich rozwoju zawodowego problematyki.

4. Prekursorski charakter projektu sprawił, iż znaczną część przeznaczonych nań zasobów i środków pochłonęły działania wprowadzające go na rynek edukacyjny, ale także w środowisko firm branży IT. Było to także powodem trudności z pozyskaniem większej liczby firm zainteresowanych udziałem w nim.

5. Projekt wykazał, iż najefektywniej uczestniczyli w nim nauczyciele o najwyższych kwalifikacjach merytorycznych, najłatwiej wchodzili oni w zagadnienia i problematykę stażu w firmie.

6. Firmy branży IT w naturalny sposób stanowią część społeczeństwa informacyjnego, stąd też staże stworzyły nauczycielom ogromną szansę na dostrzeżenie konieczności przekształcania szkół, w których pracują z placówek przekazujących wiedzę w placówki zarządzające wiedzą.

7. Staże pozwoliły ich uczestnikom dostrzec znaczenie dla współczesnej firmy systemu ciągłego doskonalenia pracowników, kreowania kultury organizacji opartej na wiedzy, jak również stosowania w praktyce metod zarządzania przez cele.

8.Udział w projekcie dla wielu uczestników był zobrazowaniem różnicy pomiędzy doskonaleniem realnym a doskonaleniem fikcyjnym, z jakim wcześniej mieli do czynienia.

9. Dla sfinalizowania projektu niezbędne byłoby przeprowadzenie ewaluacji pracy nauczycieli w nim uczestniczących w celu zdiagnozowania jego wpływu na jakość ich pracy.

Rekomendacje

1. Należy podjąć pilne działania związane ze stworzeniem systemu doskonalenia dla nauczycieli przedmiotów zawodowych, szczególnie w ramach prowadzonych przez samorząd województwa publicznych, regionalnych placówek doskonalenia nauczycieli. W ich strukturze powinna zostać wydzielona część ds. tej problematyki. Równocześnie w ramach środków, którymi dysponuje marszałek województwa na doskonalenie nauczycieli powinna zostać wydzielona pula na doskonalenie nauczycieli przedmiotów zawodowych, ale również wspierających kształcenie zawodowe przedmiotów ścisłych. Dotychczas w działaniach tych ośrodków zdecydowanie dominują formy przeznaczone dla nauczycieli ogólnokształcących przedmiotów humanistycznych oraz działania o charakterze akcyjnym.

2. Staże w firmach winny stać się integralną, obowiązkową częścią doskonalenia nauczycieli pracujących w obszarze kształcenia zawodowego. Powinni być oni zobowiązani do okresowego odbywania takich staży. Za ich organizację winien odpowiadać regionalny ośrodek doskonalenia nauczycieli, ściśle współpracując w tej materii z regionalnym centrum kształcenia praktycznego. Środki na ten cel powinny pochodzić z puli przeznaczonej corocznie na doskonalenie nauczycieli.

3. Każdy staż powinien zostać poprzedzony analizą potrzeb potencjalnych uczestników, a następnie realizowany zgodnie z wynikającymi z tej analizy celami. W jego trakcie nauczyciel powinien otrzymać szansę pełnego zapoznania się ze specyfiką firmy, w której będzie odbywał ów staż.
4. Należy przygotować narzędzia do ewaluacji skuteczności udziału w stażu dla codziennej pracy nauczycieli oraz jakości pracy szkoły. W przypadku branży IT w ramach ewaluacji powinny zostać uwzględnione nie tylko wyniki egzaminów przygotowania zawodowego, ale dla techników również wyniki egzaminu maturalnego z fizyki i matematyki. Dyrektorzy szkół branży IT winni zadbać o korelację programową szkolnych programów nauczania przedmiotów ścisłych z przedmiotami kształcenia kierunkowego.
5. Należy przygotować program staży dla dyrektorów szkół zawodowych. Dla dyrektora szkoły zawodowej zapoznanie się ze specyfiką rynku pracy w branży, w której szkoła kształci mieć będzie niebagatelne znaczenie dla skuteczności jego pracy. Trzeba przypomnieć, iż optymalnym rozwiązaniem jest zgodne z daną branżą wykształcenie dyrektora szkoły. Równocześnie udział w stażu winien pomóc dyrektorowi w kreowaniu zintegrowanej programowo pracy szkoły.
6. Osoby odpowiedzialne w danej szkole za kształcenie zawodowe powinny legitymować się certyfikacją dokonaną przez współpracujące ze szkołą firmy.

dr Jerzy Lackowski

Branża IT w Polsce

I. Ogólna charakterystyka branży IT w Polsce

Technologie informacyjne to jeden z najbardziej rozwojowych i dynamicznych obszarów nauki. Pojawiają się stale nowe rozwiązania, obejmujące coraz więcej dziedzin życia społeczno-gospodarczego. Dynamika branży sprawia, że pojęcie zawodu informatyka i programisty poszerza się, a nawet zupełnie zmodyfikuje. Nie zmienia to jednak faktu, że zapotrzebowanie na specjalistów IT systematycznie wzrasta w każdej dziedzinie produkcji i usług. W chwili obecnej branża technologii informatycznych i telekomunikacji rozwija się najdynamiczniej, a zapotrzebowanie na pracowników zarówno w kraju, jak i za granicą, stale rośnie.

Branża wyróżnia się zatem największą dynamiką zatrudnienia w długim czasie i pewnością dalszego, szybkiego rozwoju produkcji, którego granice trudno dziś określić. Prognozowanie zmian w zakresie branży informatycznej jest trudne z uwagi na niezwykle dynamiczne zmiany zachodzące w tym sektorze, skutkujące powstawaniem nowych zawodów i znajdowaniem coraz to nowszych zastosowań technologii. Równocześnie branżę tą charakteryzują problemy z pozyskiwaniem kadr przez przedsiębiorców. Każde ożywienie koniunktury wywołuje na tyle istotny wzrost popytu na pracę z ich strony, że nie może być on zaspokojony poprzez istniejącą podaż. Występuje duże zainteresowanie młodzieży zdobywaniem zawodów związanych z informatyką i technologiami telekomunikacyjnymi, lecz wiedza i umiejętności potencjalnych pracowników nie zawsze zadowalają pracodawców. Niezwykle istotne jest zatem takie kształcenie zawodowe, które zaspokoi potrzeby pracodawców. Wiedza i umiejętności merytoryczne nauczycieli zawodów w branży technologii informatycznych i telekomunikacji oraz stałe ich poszerzanie i uzupełnianie, to jeden z najważniejszych elementów wpływających na przygotowanie przyszłych pracowników. Ponieważ popyt w tej branży dotyczy kadr wysoko wykształconych, to ich przygotowanie musi być prowadzone z dużym wyprzedzeniem i uwzględniać nie do końca dziś znane zastosowania technologii informacyjnych. To oczywiście wymaga odpowiednio wysokich jakościowo warunków kształcenia, ze szczególnym uwzględnieniem wiedzy merytorycznej nauczycieli.

W Polsce duży udział – co jest charakterystyczne dla krajów na niskim poziomie rozwoju – ma montaż i sprzedaż sprzętu informatycznego, natomiast usługi związane z oprogramowaniem pozostają słabiej rozwinięte. Można spodziewać się, że rozwój kraju nasili zmiany strukturalne, tzn. przesunięcia z usług hardware'owych do software'owych i zwiększenie udziału popytu na

prace wyższej klasy specjalistów, przy czym nie chodzi tu tylko o osoby z wyższym wykształceniem, ale też z odpowiednio wysokiej jakości wykształceniem na poziomie technikum, zdolne do twórczej pracy. Jedną z głównych słabości branży IT jest zatem kapitał ludzki, co powierdziły również dane Ministerstwa Nauki i Szkolnictwa Wyższego, które wskazało, że w perspektywie najbliższych 15 lat fachowcy z dziedziny informatyki będą najbardziej poszukiwaną grupą wśród zawodów technicznych. Również analiza ogłoszeń, przeprowadzona przez serwis rekrutacyjny otoPraca.pl, wskazała na wysokie zapotrzebowanie sektora IT na pracowników. Najliczniej reprezentowane zawody w opisywanej branży to: technik informatyk, specjalista zastosowań informatyki, pozostali informatycy nigdzie indziej niesklasyfikowani, inżynier systemów komputerowych i programista. Oferty pracy dla nich są stosunkowo łatwe do uzyskania, więc bezrobocie w tej grupie niemal nie występuje. Na zatrudnienie mają szanse zarówno serwisanci, jak administratorzy sieci i managerowie IT. Zdecydowanie najwięcej ofert jest jednak skierowanych do programistów - stanowiły one blisko połowę wszystkich ogłoszeń dotyczących branży informatycznej, umieszczonych w portalu otoPraca.pl. Największa ilość z tych ogłoszeń dotyczyła specjalistów posługujących się językami PHP oraz C/C++.

II.Oczekiwania pracodawców branży IT

Branża IT jest mocno wewnętrznie zróżnicowana. Większość stanowią tu małe firmy, ton jednak nadają firmy największe, stanowiące często filie międzynarodowych korporacji. Sektor informatyczny to w dużej mierze serwisowanie, „doglądanie" istniejących rozwiązań. Innym poddziałem jest w branży opracowywanie rozwiązań informatycznych na potrzeby różnych instytucji, kreowanie narzędzi, które wspierają działalność w różnych dziedzinach gospodarki. Poszukiwana jest kadra młoda i dobrze wykształcona, gdyż w każdego absolwenta trzeba dodatkowo inwestować, zanim stanie się samodzielnym pracownikiem. Cechy poszukiwane u nowego pracownika to przede wszystkim umiejętność logicznego myślenia, zdolność rozwiązywania abstrakcyjnych problemów, znajomość podstawowych narzędzi informatycznych, umiejętność pracy w zespole oraz umiejętności komunikacyjne. Kolejna grupa poszukiwanych umiejętności to gotowość do nauki, otwartość na rady bardziej doświadczonych kolegów oraz dyscyplina pracy – umiejętność systematycznej pracy. Ważne jest też rozumienie i akceptowanie kultury pracy firmy. Pozostałe umiejętności zdobywa się już w miejscu pracy. Każda firma informatyczna dysponuje bowiem charakterystycznym dla siebie know-how, który zdobyć można tylko w niej. W dużych korporacjach mówi się o istnieniu wewnętrznych uniwersytetów, do których są wysyłani zarówno młodzi, jak i bardziej doświadczeni pracownicy. W tej branży trzeba się bowiem stale i dynamicznie rozwijać, inaczej zostaje się w tyle. Podsumowując, wiedza wynoszona z uczelni i szkół jest tylko bazą, którą uzupełnia się konkretnymi narzędziami w miejscu pracy. Ale baza ta jest niezbędna. Wiedzę tą przekazuje w dużej mierze nauczyciel szkoły lub nauczyciel akademicki. Musi więc on stale ją

aktualizować, najlepiej w rzeczywistym środowisku pracy w branży.

III. Kształcenia zawodowe w technikach i szkołach policealnych dla potrzeb branży IT

Zaspokojenie potrzeb przedsiębiorców w zakresie wiedzy i umiejętności absolwentów techników i szkół policealnych nierozerwalnie związane jest z większym udziałem pracodawców w kształtowaniu programów nauczania w zawodach branży IT, w zwiększeniu czasu trwania zajęć praktycznych realizowanych w firmach informatycznych oraz w stałym doskonaleniu merytorycznym i metodycznym nauczycieli teoretycznych i praktycznych przedmiotów zawodowych. Stały kontakt nauczycieli z nowoczesną firmą branży IT, sprzętem, oprogramowaniem, organizacja pracy poprzez odbywanie staży i praktyk w tych przedsiębiorstwach lub wręcz zatrudnianie przez szkoły (na część etatu) pracowników pracujących w branży, jest nieodzownym warunkiem właściwego przygotowania absolwentów szkół do podjęcia pracy zawodowej. Równocześnie pracodawca, aby pozyskać pracownika na miarę swoich oczekiwań i stosowanej w firmie technologii, musi przyjąć także na siebie praktyczne uczenie zawodu w przedsiębiorstwie, na stanowisku pracy. Wśród zawodów zwianych z branżą IT najczęściej w technikach i szkołach policealnych kształci się technika informatyka i technika teleinformatyka.

Technik informatyk, zgodnie z obowiązującą obecnie podstawą programową kształcenia w tym zawodzie, powinien umieć między innymi: posługiwać się systemami operacyjnymi; pracować w różnych rodzajach sieci komputerowych; obsługiwać urządzenia wykorzystywane w sieciach komputerowych; projektować i wykonywać sieć komputerową; wykorzystywać wiedzę z zakresu budowy i działania systemów operacyjnych do pracy z różnymi komputerami, rodzajami systemów operacyjnych oraz sieci; posługiwać się typowym oprogramowaniem użytkowy i narzędziowym; dobierać konfiguracje sprzętu i oprogramowania dla podstawowych zastosowań zawodowych; posługiwać się językami obsługi wybranych rodzajów baz danych, w tym językiem SQL; zakładać i utrzymywać bazy danych; administrować bazami danych i nadzorować ich pracę; projektować bazy danych; dobierać oprogramowanie do obsługi baz danych; programować w wybranych językach (Pascal, C++, Java) w środowisku graficznym i tekstowym; stosować inny powszechnie używany język programowania; stosować metody programowania i doboru algorytmów. Technik teleinformatyk, zgodnie z obowiązującą obecnie podstawą programową kształcenia w tym zawodzie, powinien umieć między innymi: montować, konfigurować i eksploatować komputery i ich podzespoły; charakteryzować i instalować systemy operacyjne i oprogramowanie narzędziowe na stacjach roboczych i serwerach; instalować, konfigurować i diagnozować urządzenia wewnętrzne i peryferyjne komputera oraz serwerów; opisywać i konfigurować systemy operacyjne telefonów komórkowych; zabezpieczać komputery

i serwery oraz dane operacyjne w systemach teleinformatycznych; obsługiwać komputery i specjalistyczne oprogramowanie sterujące pracą urządzeń teletransmisyjnych; organizować pracę lokalnej sieci komputerowej w małych i średnich przedsiębiorstwach; uruchamiać sieć teleinformatyczną oraz tworzyć proste programy sterujące sieciami teleinformatycznymi; dobierać metody pomiarowe i przyrządy do pomiaru wielkości elektrycznych i nieelektrycznych, określających sprawność sprzętu komputerowego i sieci teleinformatycznych; standaryzować i integrować systemy teleinformatyczne; standaryzować stacje robocze i serwery; wykonywać przeglądy i naprawy urządzeń teleinformatycznych i sprzętu komputerowego; prowadzić dokumentację eksploatacyjną komputerów i sieci teleinformatycznych.

Najbardziej jednak istotną z punktu doskonalenia nauczycieli staje się nowa podstawa kształcenia w zawodach oraz konieczność stworzenia przez nauczycieli programów nauczania do zawodu. Kształcenie uczniów według nowej podstawy programowej rozpoczęło się w polskich szkołach 1 września 2011 r. Zatem efekty kształcenia, wspólne dla wszystkich zawodów oraz właściwe dla kształconego zawodu: technik informatyk, technik telekomunikacji, technik teleinformatyk, powinny stać się podstawą projektowania zakresu doskonalenia nauczycieli branży IT. Zgodnie z oczekiwaniami pracodawców, nauczyciele muszą wyposażyć absolwenta szkoły, bez względu na kształcony zawód, w umiejętności ogólno zawodowe - dotyczące kompetencji personalnych i społecznych, oraz organizacji małych zespołów.

Na kompetencje personalne i społeczne absolwenta składa się kreatywność i konsekwencja w realizacji zadań; przewidywanie skutków podejmowanych działań; współpraca w zespole; przestrzeganie zasad kultury oraz etyki; stała aktualizacja wiedzy i doskonalenie umiejętności zawodowych; otwartość na zmiany; umiejętność radzenia sobie ze stresem; przestrzeganie tajemnicy zawodowej.

Na kompetencje absolwenta związane z organizacją pracy małych zespołów składa się: umiejętność planowania prac zespołu w celu wykonania przydzielonych zadań; doboru osoby do wykonania przydzielonych zadań; kierowanie wykonaniem przydzielonych zadań; ocenianie jakości wykonania przydzielonych zadań; wprowadzanie rozwiązań technicznych i organizacyjnych, wpływających na poprawę warunków i jakość pracy; umiejętność komunikowania się ze współpracownikami.

Zakres kwalifikacji wspólnych dla zawodów technik informatyk i technik teleinformatyk obejmuje w nowej podstawie programowej kwalifikację E 13 „Projektowanie i administracja lokalnych sieci komputerowych”. Efekty kształcenia obejmują wiedzę i umiejętności absolwenta w obszarach: projektowania i wykonywania lokalnej sieci komputerowej, administrowania

sieciowymi systemami operacyjnymi, konfigurowania urządzeń sieciowych.

W ramach przygotowania projektowania i wykonywania lokalnej sieci komputerowej uczeń między innymi: rozpoznaje topologie lokalnych sieci komputerowych; wykonuje projekt sieci lokalnej; dobiera elementy sieci strukturalnej; dobiera urządzenia i oprogramowanie sieciowe; dobiera rodzaj okablowania do budowy lokalnych sieci komputerowych; wykonuje czynności związane z projektowaniem okablowania strukturalnego; dobiera przyrządy i urządzenia do montażu okablowania strukturalnego; montuje okablowanie sieciowe; wykonuje pomiary okablowania strukturalnego; montuje urządzenia sieciowe transmisji przewodowej i bezprzewodowej; projektuje strukturę adresów w sieci; tworzy podział sieci na podsieci wirtualne; wykonuje pomiary i testy sieci logicznej; modernizuje istniejącą lokalną sieć komputerową; opracowuje dokumentację powykonawczą lokalnej sieci komputerowej.

W ramach administrowania sieciowymi systemami operacyjnymi uczeń między innymi: instaluje sieciowe systemy operacyjne; konfiguruje interfejsy sieciowe; udostępnia zasoby sieci lokalnej; określa funkcje profili użytkowników i zasady grup; zarządza kontami użytkowników i grup; konfiguruje usługi katalogowe sieci lokalnej; zarządza centralnie stacjami roboczymi; monitoruje użytkowników sieci; stosuje zasady udostępniania i ochrony zasobów sieciowych; konfiguruje usługi odpowiedzialne za adresację hostów, system nazw, routing, firewall; podłącza sieć lokalną do Internetu; konfiguruje usługi serwerów internetowych; lokalizuje oraz usuwa przyczyny wadliwego działania systemów sieciowych; zabezpiecza komputery przed zawirusowaniem, niekontrolowanym przepływem informacji oraz utratą danych.

W ramach konfigurowania urządzeń sieciowych uczeń między innymi: rozpoznaje urządzenia sieciowe na podstawie opisu, symboli graficznych i wyglądu; dobiera podzespoły serwerów; modernizuje i rekonfiguruje serwery; dobiera urządzenia sieciowe, jak: przełącznik, ruter, firewall; konfiguruje przełączniki sieci lokalnych; konfiguruje sieci wirtualne w lokalnych sieciach komputerowych; konfiguruje routery i urządzenia zabezpieczające typu firewall; dobiera urządzenia sieciowe transmisji bezprzewodowej; konfiguruje urządzenia dostępu do lokalnej sieci bezprzewodowej; konfiguruje urządzenia telefonii internetowej; instaluje urządzenie transmisji danych, umożliwiające podłączenie do sieci komputerowej; monitoruje pracę urządzeń sieciowych; dobiera i stosuje narzędzia diagnostyczne; konfiguruje urządzenia zabezpieczające sieć lokalną; tworzy sieci wirtualne za pomocą połączeń internetowych; monitoruje pracę urządzeń lokalnych sieci komputerowych.
Dla zawodu technik informatyk nowa podstawa programowa określa dodatkowo dwie odrębne kwalifikacje zawodowe, niezbędne do wykonywania tego zawodu: E.12 „Montaż i eksploatacja komputerów osobistych oraz urządzeń peryferyjnych” i E.14 „Tworzenie i administracja stron internetowych i baz danych”.

Efekty kształcenia kwalifikacji „Montaż i eksploatacja komputerów osobistych oraz urządzeń peryferyjnych" obejmują wiedzę i umiejętności absolwenta w obszarach: przygotowanie stanowiska komputerowego do pracy, użytkowania urządzeń peryferyjnych komputera osobistego, naprawę komputera osobistego.

W ramach przygotowania stanowiska komputerowego do pracy uczeń między innymi: planuje konfigurację sprzętu i oprogramowania do realizacji określonych zadań; dobiera urządzenia techniki komputerowej do określonych warunków technicznych; dokonuje montażu komputera osobistego z podzespołów; modernizuje i rekonfiguruje komputery osobiste; określa przebieg pracy dotyczącej przygotowania komputera osobistego do pracy; stosuje polecenia systemów operacyjnych do zarządzania systemem; instaluje i aktualizuje systemy operacyjne i aplikacje; instaluje i konfiguruje sterowniki urządzeń; konfiguruje ustawienia personalne użytkownika w systemie operacyjnym; konfiguruje ustawienia dostępu użytkowników do systemu operacyjnego; stosuje oprogramowanie narzędziowe systemu operacyjnego; stosuje oprogramowanie zabezpieczające.
W ramach użytkowanie urządzeń peryferyjnych komputera osobistego uczeń między innymi: przygotowuje do pracy urządzenia peryferyjne; dobiera i wymienia materiały eksploatacyjne urządzeń peryferyjnych; wykonuje konserwację urządzeń peryferyjnych komputera; instaluje sterowniki urządzeń peryferyjnych; konfiguruje urządzenia peryferyjne; podłącza i konfiguruje urządzenia techniki komputerowej.
Natomiast w ramach naprawy komputera osobistego uczeń między innymi: posługuje się narzędziami do naprawy sprzętu komputerowego; rozpoznaje kody błędów uruchamiania komputera osobistego; lokalizuje oraz usuwa uszkodzenia sprzętowe podzespołów komputera osobistego; lokalizuje oraz usuwa usterki systemu operacyjnego i aplikacji; lokalizuje i usuwa uszkodzenia urządzeń peryferyjnych; dobiera oprogramowanie diagnostyczne i monitorujące pracę komputera; odzyskuje z komputera dane użytkownika; tworzy kopie bezpieczeństwa.
Efekty kształcenia kwalifikacji „Tworzenie i administracja stron internetowych i baz danych" obejmują wiedzę i umiejętności absolwenta w obszarach: tworzenia stron internetowych, tworzenia baz danych i administrowania bazami oraz tworzenie aplikacji internetowych.

W ramach tworzenia stron internetowych uczeń między innymi: posługuje się hipertekstowymi językami znaczników; tworzy strony internetowe za pomocą hipertekstowych języków znaczników; tworzy kaskadowe arkusze stylów; tworzy strony internetowe za pomocą edytorów WYSIWYG; projektuje strukturę witryny internetowej; wykonuje strony internetowe zgodnie

ze scenariuszami; testuje i publikuje witryny internetowe; stosuje zasady cyfrowego zapisu obrazu; tworzy grafikę statyczną i animacje jako elementy stron internetowych; przetwarza oraz przygotowuje elementy graficzne, obraz i dźwięk do publikacji w Internecie.

W ramach tworzenia baz danych i administrowania bazami uczeń między innymi: posługuje się strukturalnym językiem zapytań do obsługi baz danych; projektuje i tworzy relacyjne bazy danych; importuje dane do bazy danych; tworzy formularze, zapytania i raporty do przetwarzania danych; instaluje systemy baz danych i systemy zarządzania bazami danych; modyfikuje i rozbudowuje struktury istniejących baz danych; dobiera sposoby ustawiania zabezpieczeń dostępu do danych; zarządza bazą danych i jej bezpieczeństwem; udostępnia zasoby bazy danych w sieci; zarządza kopiami zapasowymi i odzyskiwaniem danych; dokonuje naprawy baz danych.

Natomiast w ramach tworzenia aplikacji internetowych uczeń między innymi: korzysta z wbudowanych typów danych; tworzy własne typy danych; stosuje zasady programowania; stosuje instrukcje, funkcje, procedury, obiekty, metody wybranych języków programowania; tworzy własne funkcje, procedury, obiekty, metody; wykorzystuje środowisko programistyczne: edytor, kompilator i debugger; kompiluje i uruchamia kody źródłowe; wykorzystuje języki programowania do tworzenia aplikacji internetowych, realizujących zadania po stronie serwera; stosuje skrypty wykonywane po stronie klienta przy tworzeniu aplikacji internetowych; stosuje frameworki do budowy własnych publikacji; pobiera i składuje dane aplikacji w bazach danych; testuje działanie tworzonej aplikacji i modyfikuje jej kod źródłowy.

Wyniki egzaminów potwierdzających kwalifikacje zawodowe dla technika informatyka i technika teleinformatyka w ostatnich latach nie przekraczały 40 % zdanych pozytywnie egzaminów. Kształtowanie wiedzy i umiejętności przyszłych techników - pracowników branży IT - wymaga poza dostępem do nowoczesnego sprzętu, oprogramowania, także stałej aktualizacji wiedzy merytorycznej nauczycieli teoretycznych i praktycznych przedmiotów zawodowych. Bez stałego kontaktu z przedsiębiorstwami branży IT, dysponującymi najnowszymi technologiami, sprzętem i oprogramowaniem, nauczyciel nie ma możliwości kształtowania u ucznia kwalifikacji opisanych w podstawach programowych kształcenia w zawodach. Odbywanie przez nauczycieli praktyk, staży w najlepszych przedsiębiorstwach branży dodatkowo pozwala zapoznać się z ich organizacją i kulturą pracy oraz oczekiwaniami pracodawców względem przyszłych pracowników. Jest to niezwykle ważne, aby nauczyciel mógł właściwie kształtować kompetencje społeczne i personalne ucznia oraz jego umiejętność pracy w zespole.

IV. Potrzeby w zakresie doskonalenia zawodowego nauczycieli teoretycznych przedmiotów zawodowych i praktycznej nauki zawodu branży IT.

Bardzo szybki postęp technologiczny w branży IT w zakresie oprogramowania użytkowego, baz danych, systemów operacyjnych i systemów teleinformatycznych, serwisowania sprzętu, komputerowego oraz komputeryzacja procesów projektowania i wykonania wyrobów praktycznie w każdej działalności przemysłowej i usługowej, wymagają stałego doskonalenia nauczycieli szkół przygotowujących przyszłych pracowników tej branży. Samo kształcenie nauczycieli, polegające na śledzeniu nowych rozwiązań w tym zakresie jest ważne, ale jedynie staże nauczycieli w nowoczesnych organizacjach świadczących usługi w branży IT oraz zajęcia z najlepszymi fachowcami w branży pozwola na bieżąco aktualizować ich wiedzę i umiejętności. Aktualizowanie wiedzy merytorycznej w zakresie najnowszych usług hardware'owych i software'owych jest niezwykle ważne, ale istotne dla pracodawców są również: organizacja pracy w przedsiębiorstwie, organizacja pracy nad dużymi projektami i umiejętności, jakie musi posiadać absolwent szkoły w tym zakresie. Każda firma informatyczna dysponuje bowiem charakterystycznym dla siebie know-how, który zdobyć można tylko w niej. Istotny staje się żywy kontakt nauczyciela z zakładem pracy, także w zakresie kultury pracy organizacji, ze szczególnym uwzględnieniem oczekiwanych od jego ucznia kompetencji personalnych i społecznych. Nauczyciele mają świadomość jak ważny jest stały kontakt z pracodawcami i możliwość odbywania staży w rzeczywistych warunkach pracy.
Na podstawie analizy podstaw programowych kształcenia w zawodach branży IT i zawodach wykorzystujących systemy i oprogramowanie komputerowe, można wskazać główne obszary doskonalenia zawodowego i metodycznego, tak teoretycznego, jak i praktycznego nauczycieli.

W zakresie wiedzy teoretycznej dotyczą one głównie:

- nowoczesnych rozwiązań informatycznych w poligrafii,
- zastosowań informatyki w fotografii,
- najnowszych rozwiązań w zakresie zarządzanie sieciami,
- poznania nowych języków programowania,
- nowych trendów w technologii internetowej,
- możliwości współczesnej grafiki komputerowej,
- zarządzania projektami informatycznymi,
- teoretycznych podstaw pracy zespołowej;
- metodologii i praktyki tworzenia aplikacji internetowych,
- administrowanie serwerami,
- programowanie, multimedia i grafika komputerowa,
- budowa i serwisowanie najnowszego sprzętu komputerowego,
- tworzenie statycznych i dynamicznych baz danych,
- teoretyczne podstawy programowania w językach PHP i C++,
- bezpieczeństwa sieci komputerowych,
- tworzenia grafiki komputerowej 3D,

- zaawansowane tworzenie stron www,
- administrowanie systemami linux,
- przetwarzanie w chmurze.

W zakresie wiedzy praktycznej dotyczą one głównie:

- obsługi programów komputerowych w fotografii,
- nabycia umiejętności tworzenia grafiki komputerowej,
- obsługi programów komputerowych w poligrafii,
- umiejętności zarządzania serwerem,
- tworzenia systemów i sieci komputerowych,
- administrowania sieciami,
- tworzenie serwisów internetowych,
- zawansowanej budowy stron www,
- administrowania serwerami,
- programowania we współczesnych językach programowania,
- serwisowania i konfigurowania najnowszego sprzętu komputerowego,
- obsługi zaawansowanych baz danych,
- budowy i konfiguracji sieci, administrowania serwerami,
- umiejętności ochrony sieci i danych.

Zagadnienia praktyczne (związane ze stażami nauczycieli u pracodawców) dotyczyły zatem przede wszystkim nabywania umiejętności w obsłudze najnowszego oprogramowania, tworzenia i administrowania systemami i sieciami komputerowymi, bazami danych, tworzenia grafiki komputerowej i stron www, budowy i konfiguracji serwerów, serwisowania sprzętu najnowszej generacji oraz umiejętności chronienia sieci i danych.
Poznanie najnowszego sprzętu i najnowszych technologii stosowanych w branży IT oraz organizacji i kultury pracy organizacji, pozwoli nauczycielom teoretycznych przedmiotów zawodowych i praktycznej nauki zawodu w tej branży stworzyć nowe lub unowocześnić istniejące programy nauczania w szkołach i dostosować je do współczesnych wymagań rynku pracy. Dobrze przygotowany merytorycznie nauczyciel, stale aktualizujący swoją wiedzę zawodową oraz znający w praktyce współczesne technologie informacyjne i organizację przedsiębiorstw branży IT, może wyposażyć absolwenta szkoły w wiedzę i umiejętności, jakich oczekuje pracodawca.

Przykładowe skrypty szkoleniowe

Kurs programowania C++ składał się z 10 części, które wprowadziły nauczycieli w świat programowania zorientowanego obiektowo (OOP), a także szereg pomocniczych narzędzi i bibliotek, takich jak STL, Boost, bez których używanie C++ byłoby bardzo trudne.

Historia

Język C++ został stworzony w latach osiemdziesiątych XX wieku jako obiektowe rozszerzenie języka C. Początkowo najważniejszą zmianą wprowadzoną w C++ w stosunku do C było programowanie obiektowe, później jednak zaimplementowano wiele innych ulepszeń, mających uczynić ten język wygodniejszym i bardziej elastycznym od swojego pierwowzoru. Nazwa języka odzwierciedla fakt, że język ten jest rozszerzeniem języka C.

Cechy języka

Język C++ jest obecnie najważniejszym, najczęściej stosowanym językiem programowania. Dzieje się tak, ponieważ:

- zawiera on niewielką ilość stosunkowo łatwych do opanowania słów kluczowych,
- programy źródłowe są stosunkowo łatwo przenoszone między różnymi platformami,
- dysponuje obszernymi bibliotekami na wszelkie okazje – Internet, grafika, nauka, inżynieria,
- jest bardzo uniwersalny.

No to zaczynamy

Nie tak prędko, aby rozpocząć swoją przygodę z C++ konieczny jest kompilator. Skąd go wziąć? Opisane jest na stronie Kompilatory C/C++ (http://pl.cpp.wikia.com/wiki/Kompilatory).
Aby móc Ci to wytłumaczyć, musimy przeanalizować w jaki sposób powstaje program w C++.

Tworzenie programu w C++ składa się z **4 etapów**. Pierwszy etap to edycja. W tym etapie wykorzystywany jest program edytora (np. Notatnik w Windows). Jest to moim zdaniem najważniejsza faza, gdyż tutaj wpisujesz instrukcje, które wykona komputer. Następnie plik z instrukcjami zapisywany jest na dysku. Kolejnym etapem jest kompilacja programu. Kompilacja polega na „przetłumaczeniu" instrukcji w C++ na język wewnętrzny procesora. Zanim nastąpi kompilacja wykonywany jest automatycznie program preprocesora. Preprocesor to taki program, który wykonuje instrukcje zwane dyrektywami preprocesora. Zadaniem tych instrukcji jest wykonanie pewnych czynności, zanim jeszcze dokonana zostanie kompilacja programu. Te czynności to np. dołączenie innych plików tekstowych. Preprocesor wywoływany jest przez kompilator, zanim program zostanie przekształcony na język maszynowy. Ostatnim etapem jest łączenie. Na czym to polega? Otóż programy w C++ składają się z funkcji lub obiektów zdefiniowanych w innych plikach (np. bibliotekach standardowych). Kod tworzony przez kompilator zawiera puste miejsca, które muszą zostać wypełnione przez funkcje z tych innych plików, tak aby stworzyć program w pełni wykonywalny.

Pierwszy program - "Hello world"

Działanie naszego programu będzie polegało na wyświetleniu linii tekstu.

```
// "Hello World" pierwszy program w C++.
#include <iostream>
#include <conio.h>

//using namespace std;
//przez powyższą funkce nie musisz wszędzie pisać std::

int main()
{
  std::cout << "Hello World!!!\n";
//można to też zapisać tak:
//std::cout << "Hello World!!!" << endl;
  getch();
  return 0;
}
```

Teraz przeanalizujmy program linijka po linijce, aby wszystko stało się jasne.

```
// "Hello World" pierwszy program w C++.
```

Znaki // oznaczają dla kompilatora, że po nich znajduje się komentarz, a nie instrukcje programu. Programiści używają komentarzy dla dokumentacji programu. Jest to szczególnie użyteczne w przypadku dużych projektów. Komentarze ułatwiają zrozumienie programu innym osobom

i nie powodują żadnych działań ze strony komputera, a podczas kompilacji są (przez kompilator) pomijane.

#include <iostream>
#include <conio.h>

Jest to dyrektywa preprocesora, czyli instrukcja dla programu preprocesora. Linie rozpoczynające się od znaku # są przetwarzane przez preprocesor zanim program zostanie skompilowany. Instrukcja include nakazuje preprocesorowi umieszczenie w programie zawartości pliku nagłówkowego iostream.h . Ten plik musi być dołączony, ponieważ nasz program wysyła informacje wyjściowe na ekran (io - input/output, stream - strumień).
using namespace std;
Gdy się to umieści na początku kodu, nie trzeba już pisać przy funkcjach std:: (np. zamiast std::cout wystarczy cout)

int main()

Każdy program w C++ składa się z jednej lub kilku funkcji, z których tylko jedna musi nazywać się main. Programy C++ zawsze rozpoczynają wykonywanie funkcji od main, nawet jeśli nie jest to pierwsza funkcja. Słowo int oznacza, że funkcja zwraca wartość będącą liczbą całkowitą. Lewy nawias klamrowy ‘ { ‘ pokazuje początek bloku funkcji, odpowiadający mu prawy nawias kończy ostatnio rozpoczęty i jeszcze nie zamknięty blok funkcji. Więcej o funkcjach będzie w 3 odcinku naszego kursu.

std::cout << “Hello World\n”;
//lub
std::cout << “Hello World!!!” << endl;

Linia ta to pojedyncza instrukcja.Nakazuke ona wyświetlenie napisu “Hello World” oraz przeniesienie kursora do nowej linii (co czyni instrukcja /n lub endl<. Składa się ona z obiektu strumienia standardowego wyjścia cout (ang. see out), przyłączonego normalnie do ekranu, operatora <<, zwanego operatorem wstawiania do strumienia, napisu “Hello World\n” oraz średnika ;. Każda instrukcja musi się kończyć średnikiem (jest to tzw. zakończenie instrukcji, ale o tym dalej). Brak średnika na końcu instrukcji powoduje wygenerowanie błędu składniowego. Operator << nazywany jest operatorem wstawiania do strumienia, natomiast napis “Hello World\n” to jego prawy operand (wartość znajdująca się na prawo od operatora). Backslash to znak specjalny, zmieniający znaczenie następnego znaku. Kiedy znak \ zostanie napotkany w napisie, to następny znak jest z nim łączony w celu stworzenia sekwencji o specjalnym

znaczeniu. Sekwencja “\n” i “endl” oznacza przejście do nowej linii.

getch();

To jest również pojedyncza instrukcja, która wymaga wciśnięcia dowolnego przycisku na klawiaturze, aby przejść dalej. By jej użyć trzeba użyć conio.h Można to przetłumaczyć jako “get char” - “weź znak”. Na jej końcu są dwa nawiasy ().
O ich roli dowiesz się później. Funkcja ta musi (!!!) być zakończona średnikiem (jak zresztą większość funkcji).

return 0;

Za pomocą słowa kluczowego return zwracana jest wartość z funkcji. Wartość 0 dla funkcji main oznacza, że program zakończył się pomyślnie. Więcej na temat zwracania różnych wartości przez funkcje będzie w 3 odcinku.

Zmienne i arytmetyka w C++

Na początek mały programik.

```
// Program, który prosi użytkownika o podanie
// 2 liczb, a następnie je mnoży
// i wyświetla wynik na ekranie.

#include <iostream>

using namespace std;

int main()
{
 // Do dobrych nawykow nalezy inicjowanie zmiennych
 // wartosciami
 int liczba1 = 0;
 int liczba2 = 0;
 int iloczyn = 0;

 cout<<”Wprowadz pierwsza liczbe calkowita\n”;
 cin>>liczba1;
```

```
 cout<<”Wprowadz druga liczbe calkowita\n”;
 cin>>liczba2;

 // Pobieramy znak nowej linii ktory
 // pozostal w strumieniu po poprzedniej operacji
 cin.get();

 iloczyn=liczba1*liczba2;
 cout<<” Iloczyn wynosi : “<<iloczyn<<endl;

 //Czekamy na wcisniecie klawisza Enter
 cout<<”Wcisnij Enter zeby zakonczyc”<<endl;
 cin.get();
 return 0;
}
```

Jeśli chodzi o linie:

```
// Program, który prosi użytkownika
// o podanie 2 liczb, a następnie je mnoży
// i wyświetla wynik na ekranie.

#include <iostream>

using namespace std;

int main()
{
```

patrz wyżej.

```
int liczba1,liczba2,iloczyn;
```

Linia ta to deklaracja trzech zmiennych całkowitych o nazwach liczba1, liczba2, iloczyn. Deklarację można również napisać w ten sposób:

```
int liczba1;
int liczba2;
int iloczyn;
```

Powyżej zmienna deklarowana jest w trzech oddzielnych instrukcjach. Poniżej deklarujemy zmienne w jednej instrukcji, ale w trzech liniach :

```
int liczba1,
liczba2,
iloczyn;
```

Nazwa zmiennej to dowolny dozwolony identyfikator. Identyfikator to układ znaków składający się z liter, cyfr i znaków podkreślenia(_). Nie może on rozpoczynać się cyfrą ani być słowem kluczowym. W C++ rozróżniane są duże i małe litery, więc Liczba1 i liczba1 to dwie różne zmienne. Deklaracje zmiennych mogą być umieszczane wszędzie w funkcji, pod warunkiem, że deklaracja pojawi się zanim zmienna zostanie użyta. Każda zmienna ma nazwę, typ, rozmiar i wartość. Rozmiar zmiennej określa ile komórek w pamięci zajmuje zmienna. Jeśli chodzi o typ zmiennej to określa go słowo kluczowe typu danych poprzedzające nazwę zmiennej (w naszym wypadku jest to słowo int wskazujące na to, że zmienna jest typu integer - liczbą całkowitą). Wartość zmiennej liczba1 jest nadawana instrukcją:

```
cin>>liczba1;
```

Instrukcja ta używa obiektu strumienia wejściowego cin(ang. see in) oraz operatora pobierania za strumienia >> do uzyskania liczby wpisywanej przez użytkownika. Cin i operator >> pobiera wejście ze standardowego strumienia wejściowego jakim zazwyczaj jest klawiatura i przypisuje ją zmiennej liczba1. Inaczej mówiąc cin przekazuje wartość wpisaną przez użytkownika do zmiennej liczba1. Tak naprawdę wartość ta jest przekazywana do komórek w pamięci komputera, którym kompilator przypisał nazwę zmiennej.

```
iloczyn=liczba1*liczba2;
```

Instrukcja oblicza iloczyn zmiennych liczba1 oraz liczba2 oraz przypisuje wynik do zmiennej iloczyn. Instrukcja ta używa operatora przypisania = oraz operatora * wskazującego na mnożenie. Są one nazywane operatorami dwuargumentowymi, ponieważ każdy z nich ma dwa operandy. W przypadku operatora * prawym operandem jest zmienna liczba2, a lewym operandem zmienna liczba1. Operandy operatora = to wartość wyrażenia liczba1*liczba2 oraz zmienna iloczyn.

Niektóre wbudowane typy danych w C++

Dane znakowe. Wartością danej typu znakowego char mogą być pojedyncze litery, cyfry oraz inne znaki. W pamięci z reguły zajmują 1 bajt.

Liczby całkowite.

short int , int, unsigned int, long int, unsigned long int

Liczba bajtów, zajmowanych przez typy zmiennych, zależna jest od systemu operacyjnego, a od tego zależy jaką maksymalną wartość może przechowywać zmienna danego typu. Przykładowo zmienna typu short int zajmować będzie najmniej miejsca w pamięci. Przechowywane w niej wartości będą obejmowały mały zakres (np. 0-255), ale to zależy już od systemu operacyjnego bądź kompilatora. Jak sprawdzić ile bajtów zajmuje dany typ pokażę w dalszej części kursu. Typ unsigned long int oraz unsigned int oznacza dodatnią liczbę całkowitą.

Liczby zmiennoprzecinkowe.

float, double, long double

Arytmetyka w C++

Poniżej przedstawiam operatory arytmetyczne oraz opisy ich działania:

OPERATOR ARYTMETYCZNY	WYRAŻENIE ALGEBRAICZNE	WYRAŻENIE W C++	WYNIK W C++	OPIS
+	2+5	2+5	7	Dodawanie
-	5-2	5-2	3	Odejmowanie
*	2x5	2*5	10	Mnożenie
/	7/2	7/2	3	Dzielenie całkowite. Wynik z dzielenia całkowitego jest zawsze liczbą całkowitą, część ułamkowa jest po prostu obcinana.
/	7/2	7.0/2	3,5	Dzielenie z liczbami zmienno-pozycyjnymi.
%	7 mod 2	7%2	1	Dzielenie modulo. Operator dzielenia modulo dostarcza resztę po dzieleniu całkowitym. Używać go można jedynie z liczbami całkowitymi.

Operatory relacyjne oraz wprowadzenie do if

Struktura if pozwala na podejmowanie decyzji opartych na prawdzie lub fałszu jakiś warunków.
Struktura ta ma postać:

```
if (warunek)
{
 instrukcja;
```

```
...
...
...
 instrukcja;
 instrukcja;
 //itd.
}
```

Jeśli warunek jest spełniony, instrukcje wewnątrz nawiasów klamrowych zostaną wykonane. Jeśli warunek nie będzie spełniony instrukcje te zostaną pominięte. Warunki mogą być formułowane za pomocą operatorów relacyjnych lub operatorów równości, które przedstawiam poniżej:

Operator	Przykład	Znaczenie
==	x==y	true jeśli zmienne x i y są sobie równe
!=	x!=y	true jeśli zmienne x i y nie są sobie równe
>	x>y	true jeśli x jest większy od y
<	x<y	true jeśli x jest mniejszy od y
>=	x>=y	true jeśli x jest większy lub równy y
<=	x<=y	true jeśli x jest mniejszy lub równy y

Pamiętaj, aby nie mylić operatora przypisania = z operatorem równości ==. Zasady pierwszeństwa operatorów w C++:

1. Operatory w wyrażeniach znajdujących się wewnątrz nawiasów są obliczane w pierwszej kolejności. W przypadku zagnieżdżonych nawiasów, operatory w najbardziej wewnętrznej parze obliczane są pierwsze.
2. Dzielenie, mnożenie i dzielenie modulo obliczane są w następnej kolejności.
3. Następnie wykonywane jest dodawanie i odejmowanie.
4. Dalej wykonywane są operatory relacyjne(<,>,<=,>=).
5. Potem operatory równości(== i !=).
6. Operator przypisania (=).

Spróbuj określić jaki będzie porządek wykonywania działań w poniższych wyrażeniach:

```
Z=w/x-y+r%q*c;
```

```
Z=a*b*c*d/e;
```

```
Z=y*x%f-g*c+g;
```

A teraz czas na zastosowanie tego, co już umiemy. Poniżej przedstawiam malutki programik do analizy.

```
// Program, który prosi użytkownika
// o podanie 1 liczby, a następnie sprawdza
// czy podana liczba jest parzysta.

#include <iostream>
#include<conio.h>

using namespace std;

int main()
{
 int liczba;
 cout<<”Wprowadz liczbe calkowita a”;
 cout<<” powiem Ci czy jest ona liczba parzysta czy nie ?\n”;
 cin>>liczba;
 if(liczba%2==0){
  cout<<”Ta liczba “<<liczba<<” jest parzysta. “;
 }
 if(liczba%2!=0){
  cout<<”Ta liczba “<<liczba<<” jest nieparzysta. “;
 }
 getch();
 return 0;
}
```

Wyjaśnienia wymaga linia:

```
cout<<”Ta liczba “<<liczba<<” jest parzysta. “;
```

Ta instrukcja używa kaskadowej operacji wstawiania do strumienia. Najpierw wstawiany jest napis “Ta liczba “, następnie wstawiana jest wartość zmiennej liczba, a na końcu napis “ jest parzysta. “. Dzięki temu nie musimy pisać:

```
cout<<”Ta liczba “;
cout<<liczba;
cout<<” jest parzysta. “;
```

Ćwiczenia

1. Napisz program, który wczytuje 2 liczby i podaje, która jest większa lub mniejsza albo czy są one równe.
2. Napisz program, który wczytuje 2 liczby i podaje iloraz, iloczyn, sumę i różnicę.
3. Napisz program, który wczytuje dwie liczby całkowite i oblicza pole prostokąta oraz jego obwód.
4. Napisz program, który odczytuje 2 liczby i określa czy pierwsza jest wielokrotnością drugiej.
5. Zapoznaj się dokładnie z dokumentacją dostarczoną wraz z kompilatorem. Przeczytaj dokładnie fragmenty dotyczące kompilacji oraz łączenia.
6. Napisz program, który wczytuje 3 liczby i mówi, która jest większa.

Kurs część 2

Algorytmy

Procedura służąca rozwiązaniu problemu, określona przez działania jakie należy wykonać oraz porządek w jakim te działania zostaną wykonane, nosi nazwę algorytmu. Przed napisaniem programu służącego rozwiązaniu określonego problemu, niezbędne jest dogłębne poznanie, zrozumienie i przemyślany plan jego rozwiązania. Rozważmy taki oto przykład (bardzo często pojawiający się w różnych podręcznikach programowania) “algorytm ubierania się” : (1) załóż majtki, (2) załóż skarpetki, (3) załóż podkoszulek, (4) załóż spodnie, (5) załóż sweter, (6) załóż kurtkę, (7) załóż buty. Mamy tu wymienione działania, jakie należy wykonać wyrażone w czasowniku “załóż” oraz porządek, w jakim mają one być wykonane wyrażony cyframi. Gdybyśmy nie trzymali się określonego porządku wykonania działań, moglibyśmy np. założyć skarpetki na buty albo majtki na spodnie. Dlatego bardzo ważny jest etap projektowania programu oraz konstruowania algorytmów. Jeśli na tym etapie popełnione zostaną błędy logiczne, Twój program nie będzie działał prawidłowo, mimo np. właściwej składni.

Struktury sterujące

Zazwyczaj instrukcje programu wykonywane są jedna po drugiej, w porządku w jakim zostały napisane. Jest to wykonanie sekwencyjne. Język C++ pozwala programiście określić inny sposób wykonania programu, tzn. kolejne wykonane wyrażenie będzie inne niż następne występujące w sekwencji. Jest to tzw. przekazywanie sterowania. Wszystkie programy mogą być napisane w oparciu o trzy struktury sterujące: sekwencji, wyboru, powtórzenia. Struktura sekwencji jest wbudowana w C++. Dopóki nie jest powiedziane inaczej, instrukcje wykonywane są jedna po drugiej. C++ daje programiście trzy struktury wyboru: instrukcja if, instrukcja if/else oraz switch. Są również trzy struktury powtórzenia: while, do/while, for. Daje to nam w sumie

siedem struktur sterujących. Słowa te są słowami kluczowymi i nie mogą być one użyte jako identyfikatory, a także dla nazw zmiennych.

Struktury wyboru

Struktura if

Instrukcja if ma następującą składnię:

```
if(warunek)
{
  instrukcja;
  instrukcja;
  instrukcja;
  ...;
}
```

Jeśli warunek jest prawdziwy - instrukcje wewnątrz nawiasów klamrowych są wykonywane. Jeśli warunek jest fałszywy - instrukcje te są pomijane. Jeśli po instrukcji if ma być wykonana tylko jedna instrukcja można pominąć nawiasy klamrowe, np.:

```
if(ocena>=3)
  cout<<”Zdales”;
```

Struktura wyboru if jest stosowana wtedy, gdy chcemy skorzystać z alternatywnych sekwencji wykonania programu. Załóżmy taki warunek:

Jeśli ocena jest równa lub większa niż 3
Wyświetl “Zdałeś”

Jeśli warunek jest prawdziwy (true), wyświetlone zostaje „Zdałeś”, a dopiero potem zostaje wykonane następne wyrażenie w programie. Jeśli natomiast warunek jest fałszywy (false), instrukcja „Wyświetl “Zdałeś”” jest pomijana i wykonywana jest kolejna instrukcja. Przełóżmy teraz to, co napisałem na język C++:

```
// ocena.cpp - struktura wyboru if.
#include <iostream>
#include<conio.h>
```

```
using namespace std;
int ocena;
int main()
{
 cout<<”Wprowadz ocene :”<<endl;
 cin>>ocena;

 if(ocena>2) cout<<”Zdales!!!”<<endl;
 //tutaj mogą następować inne instrukcje programu
 getch();
 return 0;
}
```

Struktura if/else

Struktura if/else ma następującą składnię:

```
if(warunek)
{
 instrukcja;
 instrukcja;
 instrukcja;
 ...;
}
else
{
 instrukcja;
 instrukcja;
 instrukcja;
 ...;
}
```

Jeśli warunek jest prawdziwy (true), instrukcje wewnątrz nawiasów klamrowych są wykonywane, jeśli warunek jest fałszywy (false) - wykonywane są instrukcje wewnątrz nawiasów klamrowych po instrukcji else, np.:

```
if(ocena>=3)
  cout<<”Zdales”;
else
  cout<<”Nie zdales”;
```

Jeśli do wykonania jest tylko jedna instrukcja, to można pominąć nawiasy klamrowe.
W przypadku, gdy warunek będzie prawdziwy, tzn. wartość zmiennej ocena będzie równa lub większa od trzech, zostanie wyświetlony napis “Zdales”, a dopiero wówczas zostaną wykonane kolejne instrukcje programu. Jeżeli warunek będzie fałszywy, tzn. wartość zmiennej ocena będzie mniejsza od trzech, wyświetlony będzie napis “Nie zdales”, po tym będą wykonywane kolejne instrukcje programu. Oprócz instrukcji if/else C++ dostarcza operator warunkowy (?:). Jest to jedyny operator trójargumentowy w C++. Pierwszy operand jest warunkiem, drugi jest wartością dla całego wyrażenia warunkowego jeśli warunek jest prawdziwy, a trzeci jest wartością dla całego wyrażenia, jeśli warunek jest fałszywy, np.:

```
  cout<<(ocena>=3 ? “Zdales” : “ Nie zdales”);
```

Struktura wyboru switch

Składnia instrukcji:

```
switch (wyrażenie sterujące)
{
  case etykieta:
    instrukcja;
    ...;
  break ;
  case etykieta:
    instrukcja ;
    ... ;
  break;
   ...
  default:
    instrukcja ;
    ... ;
}
```

Struktura switch składa się z serii przypadków case i opcjonalnego przypadku default. Po słowie kluczowym switch następuje wyrażenie sterujące (jest nim np. zmienna ujęta w nawiasy). Wartość tego wyrażenia jest porównywana z każdą z etykiet case. Jeśli wynik porównania jest prawdziwy, wykonane zostaną instrukcje po tym przypadku case. Instrukcja break powoduje, że wykonanie programu jest kontynuowane od pierwszej instrukcji po strukturze switch. Jeżeli nie byłoby nigdzie w strukturze switch instrukcji break, to po wystąpieniu dopasowania instrukcje wszystkich pozostałych przypadków case zostaną wykonane. Jeśli nie wystąpi dopasowanie w żadnym z przypadków case, przypadek domyślny default zostanie wykonany. Każdy przypadek case może zwierać jedną lub więcej instrukcji.
W przypadku case nie są wymagane nawiasy klamrowe, gdy wykonane ma być wiele instrukcji. Instrukcja switch może być stosowana tylko do testowania stałych wyrażeń całkowitych. Oto praktyczne zastosowanie struktury switch:

```
// ocena.cpp - struktura wyboru switch.

#include <iostream>

using namespace std;

int main()
{
 unsigned short int ocena;
 cout<<”Wprowadz ocene :\n”;
 cin>>ocena;
 switch(ocena)
 {
  case 1:
    cout<<”Pala\n”;
  break;
  case 2:
    cout<<”Bardzo slabo\n”;
  break;
  case 3:
    cout<<”Trojka to dobra ocena\n”;
  break;
  case 4:
    cout<<”Czworka to bardzo dobra ocena\n”;
  break;
```

```
  case 5:
    cout<<”Piatka to super ocena\n”;
  break;
  case 6:
    cout<<”Szostka to najfajniejsza ocena\n”;
  break;
  default:
    cout<<”To nie jest ocena\n”;
 }

 //tutaj mogą następować inne instrukcje programu

 return 0;
}
```

Po wprowadzeniu wartości zmiennej poprzez instrukcję cin>>ocena; struktura switch zostaje uruchomiona. Wartość wyrażenia sterującego, mającego postać (ocena) porównywana jest z każdą z etykiet case. Jeśli użytkownik wprowadził np. 4, wystąpi dopasowanie (case 4:) i instrukcje dla tego przypadku zostaną wykonane (cout<<”Czworka to bardzo dobra ocena\n”;); wyświetlony zostanie napis “Czworka to bardzo dobra ocena” i nastąpi wyjście ze struktury switch bezpośrednio po instrukcji break;

Struktury powtórzenia

Struktura powtórzenia while

Składnia instrukcji:

```
while(warunek)
{
 instrukcja;
 instrukcja;
 ...;
}
```

Podobnie jak w instrukcji if, jeśli po while ma być tylko jedna instrukcja, to można pominąć nawiasy klamrowe. Dopóki warunek będzie prawdziwy (true), instrukcje wewnątrz nawiasów klamrowych będą powtarzane. Jeśli warunek stanie się fałszywy (false) - powtarzanie kończy

się i wykonywana jest kolejna instrukcja programu. Do częstych błędów należy tworzenie nieskończonych pętli, tj. takich, w których wyrażenie warunkowe nigdy nie staje się fałszywe. Aby zobaczyć działanie struktury powtórzenia while w działaniu napiszemy teraz krótki program, służący obliczaniu średniej arytmetycznej klasy. Przed przystąpieniem do pisania programu musimy się zastanowić jakie instrukcje i w jakiej kolejności program ma wykonać. Algorytm będziemy formułowali niejako “od góry” czyli od ogółu do szczegółu. Napiszmy więc co program ma robić:

1. Oblicz średnią arytmetyczną klasy.

Niestety taki stopień uogólnienia nie sposób przełożyć na C++. Spróbujmy napisać trochę bardziej szczegółowy algorytm obliczania średniej:

1. Inicjuj zmienne.
2. Wprowadź i zsumuj stopnie.
3. Oblicz i wyświetl średnią.

Niestety, aby w łatwy sposób napisać program w C++, musimy algorytm obliczania średniej jeszcze bardziej uszczegółowić:

1. Ustaw średnią na zero.
2. Ustaw liczbę uczniów.
3. Ustaw licznik na jeden.
4. Ustaw wynik na zero.
5. Jeśli licznik jest mniejszy lub równy liczbie uczniów:
    1. Wprowadź następny stopień.
    2. Dodaj stopień do wyniku.
    3. Dodaj jeden do licznika.
6. Ustaw średnią klasy na wynik podzielony przez liczbę uczniów.
7. Wyświetl średnią klasy.

Z takim stopniem szczegółowości możemy przystąpić do pisania programu w języku C++. Oto “przełożony” na język C++ algorytm obliczania średniej:

```
// srednia.cpp struktura while .

#include <iostream>
```

```
using namespace std;

int main()
{
 double srednia=0;
 int stopien, licznikPetli, wynik, liczbaUczniow;

 stopien=0;
 licznikPetli=1;
 wynik=0;
 liczbaUczniow = 0;

 cout<<”Ilu jest uczniow w Twojej klasie?”<<endl;
 cin>>liczbaUczniow;

 while(licznikPetli<=liczbaUczniow)
 {
  cout<<”Wprowadz stopien “<<licznikPetli<<” ucznia”<<endl;
  cin>>stopien; // pojedynczy stopień
  wynik=wynik+stopien;
  licznikPetli=licznikPetli+1;
 }

 srednia=static_cast<double>(wynik)/liczbaUczniow;
 cout<<”Srednia w Twojej klasie wynosi\t”<<srednia;

 return 0;
}
```

Zgodnie z naszym planem program najpierw inicjuje zmienne. Zmienna średnia została zadeklarowana jako double, czyli liczba z miejscami dziesiętnymi. Zmienne wykorzystywane do przechowywania wyników, powinny być zwykle inicjowane wartością zero przed ich użyciem. Jeśli zmienna nie zostanie zainicjowana przed jej użyciem, będzie zawierać poprzednią wartość przechowywaną w komórkach pamięci. Zmienne licznika są z reguły inicjowane wartością zero lub jeden w zależności od ich użycia. Niezainicjowane zmienne przechowują błędne wartości (tzw. niezdefiniowaną wartość) ostatnio przechowywaną pod adresem zarezerwowanym dla tej zmiennej. Według konwencji przyjętej przez wielu programistów, zmienne inicjowane są w ten sposób, że każda zmienna jest inicjowana w oddzielnej linii. Poprawia to czytelność

programu. Dobrze jest, gdy przyjmiemy pewne reguły, co do zasad nazywania zmiennych i ściśle się ich trzymamy. Ułatwia to późniejsze testowanie i wykrywanie błędów w programie. Warto nadawać zmiennym nazwy zgodne z ich przeznaczeniem. Ja preferuję rozpoczynanie nazw zmiennych z małej litery. Po inicjacji zmiennych program prosi użytkownika o podanie liczby uczniów jego klasie. Wartość wpisana przez użytkownika jest przypisywana do zmiennej "liczbaUczniow". W ten sposób zmienna "liczbaUczniow" jest inicjowana wartością wpisywaną przez użytkownika. Następnie program rozpoczyna pętlę while. Dopóki warunek jest prawdziwy, tzn. "licznikPetli" jest mniejszy lub równy zmiennej "liczbaUczniow", instrukcje znajdujące się w nawiasach klamrowych są wykonywane. Program prosi użytkownika o podanie stopnia ucznia poprzez kaskadową instrukcję cout<<"Wprowadz stopien "<<licznikPetli<<" ucznia\t";. Używa on zmiennej "licznikPetli" do wyświetlania numeru ucznia, którego stopień chcemy wprowadzić. Następnie instrukcja wynik=wynik+stopien; dodaje do zmiennej wynik wartość zmiennej stopien, wówczas wartość tego działania przypisywana jest do zmiennej wynik. Po tym zmienna "licznikPetli" jest zwiększana o jeden. Po tym warunek pętli while jest ponownie sprawdzany. Jeśli jest on fałszywy, będzie wykonywana następna instrukcja po nawiasach klamrowych. Jak wiemy z poprzedniego odcinka kursu, wynik z dzielenia całkowitego i jest liczbą całkowitą, średnia tymczasem nie zawsze jest całkowita. Chcąc wytworzyć zmiennoprzecinkowe obliczenie z wartościami całkowitymi, musimy utworzyć tymczasowe wartości, które są liczbami zmiennoprzecinkowymi (tzn. z miejscami dziesiętnymi). C++ dostarcza jednoargumentowy operator rzutowania, abyśmy mogli to wykonać (static_cast<typ danych>(zmienna)).

Użycie operatora rzutowania jest nazywane jawną konwersją. Wartość przechowywana w zmiennej wynik jest nadal liczbą całkowitą. Natomiast obliczenie z wartości zmiennoprzecinkowej wartości (tymczasowej wersji zmiennej wynik) podzielonej przez całkowitą wartość zmiennej "liczbaUczniow". Kompilator C++ zna informację tylko o sposobie obliczania wyrażeń, w których typy danych operandów są identyczne. Aby móc przeprowadzić wyrażenie z mieszanymi operandami, przeprowadzana jest promocja (zwana konwersją niejawną) na wybranych operandach, np. jeżeli mamy jeden operand typu double, a drugi typu int operandy typu int są promowane do double. Dokładne reguły promocji zostaną omówione w następnych odcinkach kursu. Po przeprowadzeniu jawnej konwersji zmiennej wynik i niejawnej konwersji zmiennej "liczbaUczniow" przeprowadzane jest działanie dzielenia na tymczasowych zmiennoprzecinkowych wartościach tych zmiennych i wynik tego działania przypisywany jest do zmiennej średnia. Po tym program wyprowadza wartość średniej na ekran.

Struktura powtórzenia for

Struktura ta ma składnię:

```
for (inicjacja zmiennej; warunek kontynuacji pętli; inkrementacja/dekrementacja zmiennej)
{
 instrukcja;
 instrukcja;
 ...;
}
```

Struktura powtórzenia for jest doskonałym narzędziem wszędzie tam, gdzie zachodzi potrzeba powtarzania kontrolowanego licznikiem. W strukturze for wymagane są dwa średniki. Najlepiej będzie, gdy wyjaśnię na przykładzie jak działa ta struktura.

```
// for.cpp Powtarzanie kontrolowane licznikiem

#include <iostream>

using namespace std;
int main()
{
  for(int licznik=1;licznik<=100;licznik++)
    cout<<licznik<<”\n”;

  return 0;
}
```

Gdy struktura for rozpoczyna wykonywanie najpierw deklarowana jest (jako typ int) i inicjowana (wartością 1) zmienna licznik. Po tym sprawdzany jest warunek kontynuacji pętli. Ponieważ wartość zmiennej licznik jest 1, warunek jest prawdziwy, więc wyświetlane jest 1. Następnie zmienna licznik jest inkrementowana (licznik++). Więcej o operatorach inkrementacji dowiecie się w dalszej części kursu. Cały ten proces jest kontynuowany, aż zmienna licznik osiągnie wartość 100. Dla struktury for nie ma znaczenia czy użyjesz pre- czy postinkrementacji. Wartość licznika zwiększana jest dopiero, gdy ciało for zostanie wykonane.

Struktura powtórzenia do/while

Struktura do/while jest podobna do struktury while. W strukturze while warunek kontynuacji pętli jest sprawdzany, zanim jej ciało zostanie wykonane, natomiast w do/while sprawdza warunek kontynuacji pętli po tym, jak ciało pętli zostaje wykonane. Wynika z tego, że ciało zostanie wykonane przynajmniej jeden raz. Składnia instrukcji:

```
do
{
 instrukcja;
 instrukcja;
 ...;
}
while(warunek);
```

Kiedy warunek jest fałszywy wykonanie programu jest kontynuowane od pierwszej instrukcji po while.

Instrukcje break i continue

Instrukcja break, gdy jest wykonywana w strukturze while, for, do/while, switch, powoduje bezpośrednie wyjście z tej struktury.

```
//break.cpp przykład użycia break
#include <iostream>

using namespace std;

int main()
{

 for(int a=1;a<=20;a++)
 {
  if(a==10)
    break;

  cout<<a<<" ";
 }
```

```
 return 0;
}
```

Gdy wartość zmiennej a osiągnie wartość 10 pętla for zostanie przerwana.
Wyrażenie continue, gdy jest wykonywane w strukturze while, for, do/while, switch. pomija pozostałe wyrażenia w ciele tej struktury i kontynuuje następną iteracją.

```
//continue.cpp przykład użycia break
#include <iostream>

using namespace std;

int main()
{

 for(int a=1;a<=20;a++)
 {
  if(a==10)
   continue;

  cout<<"a="<<a<<"\t";
 }

 return 0;
}
```

Gdy wartość zmiennej a będzie się równała 10 dalsze wyrażenia w pętli for zostaną pominięte, tzn. program wyświetli wszystkie wartości zmiennej a od 1 do 20 z pominięciem wyświetlenia wartości 10.

Operatory przypisania

C++ zwiera różne operatory, których celem jest skrócenie wyrażeń przypisania, np.:

```
a=a*3;
```

może zostać napisane jako :

```
a*=3;
```

Możliwości graficzne programu może zostać przekształcone do zmienna operator=wyrażenie; przy czym operator musi być jednym z operatorów dwuargumentowych +, -, *, /, %.

Operatory inkrementacji i dekrementacji

W C++ wbudowane są też jednoargumentowe operatory inkrementacji i dekrementacji.
Wyrażenie w postaci ++a inkrementuje zmienną a o 1, a następnie używa nowej wartości a w wyrażeniu w którym jest a. Wyrażenie w postaci a++ najpierw używa a w wyrażeniu, w którym a występuje, a następnie zwiększa a o jeden.
Wyrażenie w postaci --a zmniejsza a o jeden, a następnie używa nowej wartości a w wyrażeniu, w którym a występuje. Wyrażenie w postaci a-- najpierw wykorzystuje a w wyrażeniu, w którym a występuje, a potem zmniejsza a o jeden.

Operatory logiczne

C++ dostarcza również operatorów logicznych, aby umożliwić nam formułowanie bardziej złożonych warunków. C++ sprawdza pod względem prawdy lub fałszu wszystkie wyrażenia, które zawierają operatory relacyjne, równości oraz operatory logiczne.

Operator iloczynu logicznego(AND)

Operator iloczynu logicznego zapisujemy: &&. Tabela prawdy dla tego operatora:

Wyrażenie1	Wyrażenie2	Wyrażenie1 & Wyrażenie2
false	false	False
false	true	False
true	false	False
true	true	True

Czyli np. mamy wyrażenie:

```
if((a<10) && (b==5))
```

Jeśli wartość pierwszego wyrażenia i drugiego wyrażenia będzie miało wartość true, wartość całego wyrażenia będzie miało wartość true. Jeśli którekolwiek z tych wyrażeń będzie miało wartość false, wartość całego wyrażenia będzie miało wartość false. Operator iloczynu logicznego stosujemy wtedy, gdy chcemy się upewnić, że dwa warunki są prawdziwe.

Operator sumy logicznej (OR)

Operator sumy logicznej zapisujemy: ||. Tabela prawdy dla tego operatora pokazuje wszystkie możliwe kombinacje:

Wyrażenie1	Wyrażenie2	Wyrażenie1 || Wyrażenie2
false	false	False
false	true	True
true	false	True
true	true	True

Logiczne OR stosujemy wtedy, gdy chcemy się upewnić, że jeden lub dwa warunki są prawdziwe.

Operator negacji logicznej

Operator negacji logicznej ! jest, w przeciwieństwie do operatorów && i ||, operatorem jednoargumentowym. Operator ten ma tylko jeden warunek jako wyrażenie. Jest on umieszczany przed warunkiem, np. :

```
if( !(wyrażnie==wyrażenie))
```

Możemy to zapisać również za pomocą operatora relacyjnego != :

```
 if(wyrażenie!=wyrażenie)
```

Tabela prawdy dla tego operatora:

Wyrażenie	!Wyrażenie
true	false
false	true

Ćwiczenia

1. Napisz program, który obliczy średnie zużycie paliwa. Program powinien pobierać liczbę przejechanych kilometrów i zatankowanych litrów przy każdym tankowaniu. Program powinien obliczyć i wyświetlić zużycie paliwa przy każdym tankowaniu, a także dla wszystkich tankowań.
2. Napisz program wyświetlający potęgi liczby 2 (2,4,8,16 ..).#
3. Napisz program odczytujący promień koła i obliczający oraz wyświetlający średnicę, obwód oraz pole.

4. Napisz program odczytujący 3 niezerowe wartości zmiennoprzecinkowe (float) oraz określający i wyświetlający informację, czy mogą one stanowić długość boków trójkąta.
5. Trójkąt prostokątny może mieć boki, których długości są liczbami całkowitymi. Trójka pitagorejska musi spełniać jeden warunek: suma kwadratów dwóch boków musi być równa kwadratowi przeciwprostokątnej. Znajdź wszystkie trójki pitagorejskie dla wartości nie dłuższych niż 500.

Kurs część 3

Funkcje

Wstęp

Programy komputerowe są tworami bardzo złożonymi. Umieszczenie wszystkich instrukcji w funkcji main() byłoby niewygodne z kilku powodów: po pierwsze wykrycie ewentualnych błędów byłoby utrudnione, po drugie instrukcje, które powtarzałyby się, trzeba byłoby wpisywać ponownie. Funkcje pozwalają programiście na budowanie programu z mniejszych części oraz ponowne wykorzystywanie tych komponentów. Programista może pisać funkcje do wykonywania określonych zadań. Mogą one być wykonywane w wielu miejscach w programie. Nic nie stoi na przeszkodzie, aby funkcje były wykonywane w różnych programach. Funkcje są wywoływane. Wywołanie funkcji określa jej nazwę oraz argumenty przekazywane do funkcji.

Prototypy funkcji

Programy z poprzednich części kursu zawierały wywołania do funkcji z biblioteki standardowej. Teraz nauczymy się pisania własnych funkcji, które będą mogły być wykorzystywane w naszych programach. Zanim wywołamy funkcję musimy przekazać kompilatorowi informację o nazwie funkcji, argumentach przez nią pobieranych oraz o zwracanej wartości. Kompilator używa tej informacji do sprawdzania wywołań funkcji, tzn. czy argumenty jakie przekazujemy do funkcji są odpowiednie itp. Pierwsze wersje kompilatorów nie przeprowadzały tego typu sprawdzenia, co prowadziło do częstych błędów.

Mały przykład:

```
#include <iostream>
using namespace std;
//prototyp funkcji max
int max(int,int,int );
```

```
int main()
{
  int x, y, z;
  cout<<”Wprowadz trzy liczby calkowite :”<<endl;
  cin>>x>>y>>z;
  cout<<”Najwieksza z tych liczb to :”<<max(x,y,z)<<endl;

return 0;
}

//definicja funkcji max
int max(int a,int b,int c)
{
 int max=a;
 if(b>max)
  max=b;
 if(c>max)
  max=c;

 return max;
}
```

Powyższy program pobiera trzy liczby całkowite od użytkownika i określa, która z nich jest największa. Linia:

```
//prototyp funkcji max
int max(int,int,int );
```

zawiera prototyp funkcji max. Pierwszy wyraz z lewej - int - określa typ wartości zwracany przez funkcję, w tym wypadku mamy liczbę całkowitą. Kolejny wyraz jest identyfikatorem (nazwą) funkcji. Wyrazy umieszczone w nawiasach określają typ argumentów pobieranych przez funkcję - są to trzy liczby całkowite. Prototyp mógłby również wyglądać tak:

```
//prototyp funkcji max
int max(int a, int b, int c );
```

przy czym identyfikatory argumentów w prototypach są pomijane przez kompilator. Można umieszczać je tam dla celów dokumentacyjnych. Kompilator odwołuje się do prototypu funkcji,

aby sprawdzić czy wywołanie funkcji max zawiera poprawny zwracany typ, poprawną liczbę argumentów, poprawne ich typy i porządek. Prototyp funkcji nie jest wymagany, jeżeli funkcja pojawiła się przed pierwszym wywołaniem, tzn. jeżeli ciało funkcji max() umieścilibyśmy przed main(), prototyp funkcji max() można pominąć. Część prototypu funkcji, która zawiera nazwę funkcji i typy jej argumentów, nazywana jest sygnaturą. Gdybyśmy prototyp funkcji określili jako:

```
void max(int, int, int );
```

kompilator wygenerowałby błąd, ponieważ zawracany typ void w prototypie nie zgadzałby się z tym w nagłówku. Ważną cechą prototypów jest koercja argumentów. Polega to na tym, że wartości argumentów, które nie odpowiadają dokładnie typom parametrów w prototypie funkcji, są przekształcane do właściwego typu zanim funkcja zostanie wywołana. Przekształcenia czasami mogą prowadzić do błędów, jeżeli reguły promocji nie zostaną zachowane. Reguły te określają jak typy danych mogą zostać przekształcone do innych typów bez utraty danych, np. gdybyśmy do funkcji max() przekazali double jako argument ułamkowa, część double byłaby obcięta. Reguły promocji stosuje się też do wyrażeń mieszanych, czyli takich, które zwierają dwa lub więcej typów danych. Oto tabela hierarchii promocji:

long double
double
float
unsigned long
long
unsigned int
int
unsigned short
short
unsigned short
short
char

Przekształcanie z wyższego typu danych w hierarchii do niższego, może prowadzić do błędów.

Definicja funkcji

Funkcja max jest wywoływana w funkcji main w linii:

```
cout<<”Najwieksza z tych liczb to :”<<max(x,y,z)<<endl;
```

Funkcja ta otrzymuje kopie wartości x, y, z w parametrach a, b, c. Następnie porównuje ona ze sobą argumenty i zwraca największy z nich. Wynik jest przesyłany do main(), gdzie funkcja była wywołana. Definicja funkcji max() jest w liniach:

```
 //definicja funkcji max
int max(int a,int b,int c)
{
  int max=a;
  if(b>max)
   max=b;
  if(c>max)
   max=c;
  return max;
}
```

Linia:

```
int max(int a,int b,int c)
```

Wyraz int najbardziej z lewej strony oznacza, że funkcja zwraca wynik całkowity (integer). Po nim następuje identyfikator oznaczający nazwę funkcji, w nawiasie podane są parametry, jakich oczekuje funkcja oraz ich nazwy. Definicja funkcji ma więc postać:

```
typ_zwracanej_wartości nazwa_funkcji (lista_parametrów)
{
ciało funkcji;
}
```

typ_zwracanej_wartości jest typem danych zwracanych przez funkcję. Typ void oznacza, że funkcja nie zwraca wartości. W C++ typ wartości zwracanej musi być zawsze określony (nawet jeśli funkcja nic nie zwraca, wówczas musimy zasygnalizować to kompilatorowi słowem kluczowym void). nazwa_funkcji jest dowolnym dozwolonym identyfikatorem. Lista_ parametrów jest to lista oddzielonych przecinkami deklaracji parametrów otrzymywanych przez funkcję. Jeśli funkcja nie otrzymuje żadnych argumentów lista_parametrów jest typu void lub jest pusta, np.:

```
void max(void)
void max()
```

Typ parametru musi być wyraźnie napisany przed każdym parametrem. Po nawiasach () nie umieszczamy średnika. Są trzy sposoby na opuszczenie ciała funkcji:
a. jeśli funkcja nie zwraca wartości, sterowanie jest zwracane wtedy, gdy osiągnięty zostanie prawy nawias kończący funkcję,
b. jeśli funkcja nie zwraca wartości, sterowanie jest zwracane przez wykonanie wyrażenia return,
c. jeśli funkcja zwraca wartość to wyrażenie return wyrażenie.

Argumenty domyślne

Programista może określić domyślną wartość argumentu. Kiedy argument jest pominięty w wywołaniu, jego wartość jest wstawiana przez kompilator i przekazywana w wywołaniu. Argumenty te powinny być położone jak najbardziej z prawej strony. Argumenty domyślne powinny być określone wraz z pierwszym wystąpieniem nazwy funkcji, z reguły będzie to prototyp.

```
#include <iostream>

using namespace std;
//prototyp funkcji poleProstokata

int poleProstokata( int=1,int=1 );

int main()
{
 int x, y;
 cout<<"Wprowadz dwie liczby calkowite :"<<endl;
 cin>>x>>y;
 cout<<"Pole prostokata wynosi :"<<poleProstokata(x,y)<<endl;
 cout<<"Pole prostokata z argumentami domyslnymi :"<<poleProstokata()<<'\n'
   <<"Pole prostokata z jednym argumentem domyslnym:"<<poleProstokata(x)<<endl;

 return 0;
}

//definicja funkcji poleProstokata
int poleProstokata(int a,int b)
{
```

```
 return a*b;
}
```

Klasy pamięci

Każdy identyfikator ma atrybuty obejmujące : klasę pamięci, zasięg i połączenia. Do określenia klasy pamięci w C++ służą specyfikatory klas pamięci. C++ zawiera cztery specyfikatory klas pamięci: auto, register, extern, static. Klasa pamięci identyfikatora określa czas, w którym identyfikator znajduje się w pamięci. Niektóre identyfikatory istnieją bardzo krótko, a inne przez cały czas wykonywania programu. Zasięg identyfikatora określa z jakiego miejsca w programie można się odwołać do identyfikatora. Do niektórych identyfikatorów można się odwołać z każdego miejsca w programie (np. zmienne globalne), a do innych tylko z niektórych części. Połączenia identyfikatorów określają czy w programach złożonych z wielu plików źródłowych identyfikator jest znany tylko w bieżącym pliku źródłowym, czy w każdym. Specyfikatory klas pamięci mogą być podzielone na dwie klasy:

1. statyczną,
2. automatyczną.

Tylko zmienne mogą być elementami automatycznych klas pamięci. Zmienne lokalne i parametry funkcji są zwykle elementami automatycznych klas pamięci. Specyfikator auto wyraźnie deklaruje zmienne automatycznej klasy pamięci. Zmienne tego typu są tworzone podczas wykonywania bloku, w którym są deklarowane i są niszczone, gdy następuje wyjście z niego. Zmienne lokalne są domyślnie automatycznej klasy pamięci, więc słowo kluczowe auto jest rzadko używane. Specyfikator klasy pamięci register może być umieszczony przed deklaracją zmiennej automatycznej. Specyfikator ten oznacza, że kompilator powinien umieszczać tą zmienną raczej w rejestrze procesora. Kompilator może ignorować deklaracje register.

Słowa kluczowe extern i static są używane do deklarowania zmiennych i funkcji statycznej klasy pamięci. Zmienne takie istnieją od momentu, w którym program rozpoczyna wykonywanie. Pamięć jest im przydzielana i inicjowana tylko raz. Globalne zmienne i funkcje mają domyślnie klasę pamięci extern. Są one tworzone poprzez umieszczenie ich poza jakąkolwiek funkcją. Do zmiennych tych i funkcji może się odwoływać każda funkcja w pliku. Zmienne używane tylko w określonej funkcji powinny być deklarowane jako zmienne lokalne. Deklarowanie zmiennych jako globalne utrudnia wykrywanie błędów... Zmienne lokalne zadeklarowane za słowa static są nadal znane tylko w funkcji, w której zostały zadeklarowane, inaczej od zmiennych automatycznych zachowują swoje wartości po wyjściu z funkcji. Następnym razem przy wywołaniu funkcji zmienne te mają taką wartość, jaką miały przy wyjściu z funkcji.

```
#include <iostream>

using namespace std;
//prototyp fukncji zmienneStatyczne
int zmienneStatyczne( );
int main()
{
 for(int i=0;i<10;i++)
 cout<<i<<". wywolanie funkcji zmienneStatyczne() "<<zmienneStatyczne()<<'\n';
 return 0;
}
//definicja funkcji zmienneStatyczne
int zmienneStatyczne( )
{
 static int a=0;
 a++;
 return a;
}
```

Reguły zasięgu

Reguły zasięgu określają z jakiego miejsca w pliku można się odwołać do danego identyfikatora. Pięcioma zasięgami dla identyfikatora są: zasięg funkcji, zasięg pliku, zasięg bloku, zasięg prototypu funkcji oraz zasięg klasy.
Identyfikator zadeklarowany poza jakąkolwiek funkcją ma zasięg pliku. Można do niego odwoływać się od miejsca, w którym został on zadeklarowany, aż do końca pliku. Etykiety są identyfikatorami mającymi zasięg funkcji. Mogą one być używane gdziekolwiek w funkcji, w której się pojawiają, ale nie można się do nich odwołać spoza ciała funkcji. Etykiety są używane w poleceniach switch i goto. Identyfikatory zadeklarowane wewnątrz bloku mają zasięg bloku. Zasięg bloku rozpoczyna się w miejscu deklaracji identyfikatora i kończy się po osiągnięciu nawiasu klamrowego - } - . Zasięg bloku mają zmienne lokalne zdefiniowane na początku funkcji, a także parametry funkcji.

Jedynymi identyfikatorami o zasięgu prototypu funkcji są identyfikatory użyte na liście jej parametrów.

```
#include <iostream>
using namespace std;
```

```
//zmienna globalna
int a=1;
void funkcja();
int main()
{
 //zmienna lokalna w main
 int a=5;
 cout<<"Zmienna lokalna w main :"<<a<<'\n';
 //nowy blok
 {
  int a=10;
  cout<<"Zmienna lokalna w bloku :"<<a<<'\n';
 }
 funkcja();
 cout<<"Zmienna lokalna w main :"<<a<<'\n';
 return 0;
}
//definicja funkcji
void funkcja( )
{
 cout<<"Zmienna globalna to :"<<a<<'\n';
 return;
}
```

Rekurencja

Funkcja rekurencyjna jest to funkcja, która wywołuje bezpośrednio sama siebie lub pośrednio przez inną funkcję. Funkcja rekurencyjna jest wywoływana do rozwiązania określonego problemui z reguły wie, jak rozwiązać najprostszy przypadek. Jeśli wywoływana jest do problemu złożonego, dzieli go na dwa elementy: ten, który potrafi rozwiązać i ten, którego nie potrafi rozwiązać. Ten drugi element jest nieznacznie upraszczany. Funkcja wywołuje swoją kopię do pracy z tym uproszczonym elementem. Nazywane jest to krokiem rekurencji. Gdy problem zostaje uproszczony do przypadku podstawowego, wywołania rekurencyjne kończą się i funkcja zwraca wartość. Typowym problemem dla rekurencji może być silnia oraz szereg Fibonacciego. Silnia nieujemnej liczby całkowitej pisana jest n! (wymawiana jako “n silnia”). Jest to iloczyn:

n*(n-1)*(n-2)*...*1

przy czym 1! jest równe 1 i 0! jest równe 1. Dla przykładu 3!=3*2*1. Rekurencyjna silnia ma postać:

n!=n*(n-1)!

np.:

3!=3*(2!)
2!=2*(1!)
1!=1*(0!)
0!=1

Oto program obliczający silnię z liczby podanej przez użytkownika:

```
#include <iostream>
using namespace std;
//prototyp funkcji
long silniaRekurencyjnie(long);

int main()
{
 long a;
 cout<<”Wprowadz liczbe calkowita \n”;
 cin>>a;
 cout<<”Silnia liczby “<<a<<” wynosi :”<<silniaRekurencyjnie(a)<<endl;
 return 0;
}
//definicja funkcji
long silniaRekurencyjnie(long liczba)
{
 //przypadek podstawowy
 if(liczba<=1)
  return 1;
 else //przypadek złożony upraszczamy nieznacznie i funkcja wywołuje samą siebie
 return liczba*silniaRekurencyjnie(liczba-1);
}
```

Funkcja silniaRekurencyjnie() została zadeklarowana jako funkcja oczekująca parametru

typu long oraz zwracająca wynik typu long. Proponuję Wam, abyście zmodyfikowali tak ten program, aby było widać, jakie argumenty funkcja otrzymuje oraz jaką wartość zwraca.
Szereg Fibonacciego rozpoczyna się od 0 i 1 i charakteryzuje się tym, że kolejna liczba jest sumą dwóch poprzednich. Stosunek kolejnych liczb Fibonacciego jest równy w przybliżeniu 1.618. Liczba ta nazywana jest złotym podziałem. Szereg Fibonacciego może być przedstawiony rekurencyjnie:

fibonacci(0)=0
fibonacci(1)=1
fibonacci(n)=fibonacci(n-1)+fibonacci(n-2)

Oto program obliczający szereg Fibonacciego rekurencyjnie:

```
#include <iostream>

using namespace std;

//prototyp funkcji
long fibonacci(long);

int main()
{
 long a;
 cout<<”Wprowadz liczbe calkowita \n”;
 cin>>a;
 cout<<”fibonacci (“<<a<<”) wynosi :”<<fibonacci(a)<<endl;
 return 0;
}

//definicja funkcji
long fibonacci(long liczba)
{
 //przypadek podstawowy
 if(liczba<=0)
  return 0;
 if(liczba==1)
  return 1;
 else //przypadek złożony dzielimy na dwie części
```

```
 return fibonacci(liczba-1)+fibonacci(liczba-2);
}
```

Rekurencja ma wiele wad. Obliczenie 20 liczby Fibonacciego wymagać będzie 2^20 wywołań (ok. miliona wywołań !). Będzie to oczywiście znacznie obciążać zasoby komputera w efekcie może to doprowadzić do przepełnienia stosu. Dlatego rekurencje należy stosować wówczas, gdy liczba wywołań nie będzie zbyt wielka. Rekurencja ma też zalety, podstawową zaletą jest niejako “naturalny” sposób rozwiązywania problemów. W przypadku, gdy złożoność obliczeniowa naszego rekurencyjnego algorytmu jest zbyt duża, należy upraszczać rekurencję do wyrażenia nie będącego rekurencyjnym.
Drobny komentarz: Ciągu Fibonacciego nie da się obliczyć rekurencyjnie, gdyż razem ze wzrostem wartości tego ciągu dokładność będzie malała. Jedyną dokładną metodą jest metoda iteracyjna.

Referencje i parametry referencji

Funkcję można wywołać na dwa sposoby: wywołanie przez wartość (ang. call by value) oraz wywołanie przez referencje. Do tej pory funkcje wywoływaliśmy poprzez wartość. Gdy wywołujemy funkcję poprzez wartość, tworzona jest kopia argumentu, który przekazujemy w wywołaniu. Kopia ta jest przekazywana do funkcji. Wszystkie działania w ciele funkcji są wykonywane na tej kopii i nie oddziaływają na oryginalną wartość zmiennej. Dodatnią stroną tego typu wywołania jest to, że zapobiega to ujemnym skutkom ubocznym. Ujemną stroną jest to, że w przypadku dużych struktur danych kopiowanie zabiera dużo czasu i pamięci.
Dzięki wywołaniu przez referencje funkcja wywołująca przekazuje funkcji wywoływanej zdolność do bezpośredniego dostępu do danych. Nie jest tworzona kopia, a wszelkie modyfikacje są dokonywane na zmiennej bezpośrednio. Jeśli parametr ma być przekazywany do funkcji przez referencje po typie parametru, a przed identyfikatorem umieść znak ampersandu (&). Oto program demonstrujący różnicę między tymi dwoma wywołaniami:

```
#include <iostream>
using namespace std;

//prototypy funkcji
int wywolaniePrzezWartosc (int);
void wywolaniePrzezReferencje(int &);

int main()
{
```

```
 int a;
 cout<<”Wprowadz liczbe calkowita \n”;
 cin>>a;
 cout<<”a przed wywołaniem funkcji przez wartość :”<<a<<’\n’;
 cout<<”Wartość zwracana przez funkcję
 wywołaniePrzezWartość:”<<wywolaniePrzezWartosc(a)<<’\n’;
 cout<<”a po wywołaniu funkcji przez wartość :”<<a<<’\n’;
 cout<<”a przed wywołaniem funkcji przez referencje :”<<a<<’\n’;
 cout<<”Wywołanie funkcji przez referencje \n”;
 wywolaniePrzezReferencje(a);
 cout<<”a po wywołaniu funkcji przez referencje :”<<a<<endl;
 return 0;
}
//definicja funkcji
int wywolaniePrzezWartosc(int liczba)
{
 liczba=liczba*5;
 return liczba;
}

void wywolaniePrzezReferencje(int &liczba1)
{
 liczba1=liczba1*5;
}
```

Przeciążanie funkcji

W C++ można definiować kilka funkcji o tej samej nazwie, przy czym muszą się one różnić parametrami. Jest to nazywane przeciążaniem funkcji. Na przykład: mamy funkcję obliczającą pole powierzchni prostokąta. Możemy zdefiniować kilka funkcji polePowierzchni. Będą one pobierały jako argumenty różne typy (np. jedna int, druga double itp.). Kiedy funkcja przeciążona jest wywoływana kompilator C++ wybiera właściwą poprzez sprawdzenie liczby typów i porządku argumentów w tym wywołaniu. Przeciążanie funkcji jest stosowane tam, gdzie przeprowadzane są takie same obliczenia na różnych typach danych. Oto podany przykład:

```
#include <iostream>
```

```
using namespace std;

//prototypy funkcji
int polePowierzchni(int bokX,int bokY);
double polePowierzchni(double bokX,double bokY);

int main()
{
 int a=5,b=4;
 double c=5.8,d=3.2;
 cout<<"Pole powierzchni z argumentami całkowitymi "<<polePowierzchni(a,b)<<'\n';
 cout<<"Pole powierzchni z argumentami rzeczywistymi "<<polePowierzchni(c,d)<<endl;
 return 0;
}
//definicja funkcji
int polePowierzchni(int bokX,int bokY)
{
 return bokX*bokY;
}

double polePowierzchni(double bokX,double bokY)
{
 return bokX*bokY;
}
```

Ćwiczenia

1. Jaki jest zasięg zmiennej x w main.
2. Napisz funkcję określającą dla pary liczb całkowitych, czy pierwsza jest wielokrotnością drugiej.
3. Napisz funkcję main zwracającą najmniejszą z trzech liczb całkowitych.
4. Liczba jest liczbą pierwszą jeżeli dzieli się tylko przez 1 i przez samą siebie. Napisz funkcję, która określi czy dana liczba jest liczbą pierwszą.
5. Czy main() może być wywołana rekurencyjnie? Sprawdź to.
6. Największy wspólny dzielnik x i y jest największą liczbą całkowitą przez którą x i y dzielą się bez reszty. Napisz funkcję, która będzie pobierała dwa argumenty i zwracała NWD.

Kurs część 4

Tablice

Tablica jest strukturą danych. Tablica to ciągła grupa komórek w pamięci. Grupa ta posiada wspólną nazwę. Komórki pogrupowane są w typ danych, który określamy przy definicji tablicy. Do poszczególnych elementów tablicy odwołujemy się poprzez określenie nazwy tablicy i numeru pozycji odpowiedniego elementu. Elementy tablicy numerujemy od zera (pierwszy jest element zerowy). Chcąc zdefiniować tablicę 20 liczb całkowitych, piszemy:

```
int tablicaInt[20];
```

Poszczególne elementy będziemy numerować od 0 do 19. Chcąc się odwołać do pierwszego elementu tablicy, piszemy:

```
tablicaInt[0];
```

Numer elementu ujmowany w nawiasy kwadratowe nazywany jest indeksem. Indeks musi być liczbą całkowitą lub wyrażeniem całkowitym.

Deklaracja tablic

Tablice zajmują miejsce w pamięci. Przy deklaracji tablicy należy określić typ elementów tablicy oraz ich ilość. Posiadając tą informację kompilator może zarezerwować odpowiednią ilość komórek pamięci. Deklaracja tablicy ma postać:

```
typ_elementów nazwa_Tablicy [ ilość_elementów ];
```

Kilka tablic tego samego typu może być deklarowanych w tej samej linii:

```
int tablicaInt1 [20], tablicaInt [50];
```

W linii tej deklarujemy dwie tablice typu int: jedną składającą się z 20 elementów, a drugą z 50 elementów.

Inicjowanie tablic

Elementy tablicy mogą być inicjowane w deklaracji tablicy przez umieszczenie po niej znaku równości i rozdzielonej przecinkami listy wartości inicjujących ujętej w nawiasy klamrowe.

```
int tablicaInt[5]={1,2,3,4,5};
```

Jeśli wartości inicjujących będzie mniej niż elementów tablicy, pozostałe elementy będą zainicjowane wartością zero.

```
int tablicaInt[5]={5};
```

Deklaruje tablicę typu int, która ma pięć elementów. Pierwszy element tej tablicy jest jawnie inicjowany wartością 5. Pozostałe elementy tej tablicy są inicjowane niejawnie wartością zero. Do inicjacji tablic możemy również wykorzystać pętle for.

```
#include <iostream>

using namespace std;
int main()
{
 const int wTab=100;
 int tablicaInt[wTab];

 for(int i=0; i<wTab; i++)
  tablicaInt[i]=i;

 cout<<"Tablica została zainicjowana liczbami :\n";

 for(int i=0; i<wTab; i++)
  cout<<"Numer elementu :"<<i<<" wartosc elementu "<<tablicaInt[i]<<'\n';

 return 0;
}
```

Linia:

```
const int wTab=100;
```

stosuje kwalifikator const w celu zadeklarowania stałej symbolicznej wTab, której wartość wynosi 100. Stałe takie muszą być inicjowane wyrażeniem stałym i nie mogą być później modyfikowane. Stałe symboliczne nazywane są również zmiennymi tylko do odczytu. Określając liczbę elementów tablicy kompilator oczekuje wyrażenia całkowitego i stałego, dlatego możemy użyć stałej symbolicznej. Wykorzystywanie stałych symbolicznych do określenia wielkości tablic, daje programom większą skalowalność. Nic nie stoi na przeszkodzie,

aby zamiast liczby 100 elementów typu int, inicjowane było 1000 elementów poprzez zmianę stałej symbolicznej. Gdybyśmy nie używali stałej symbolicznej, musielibyśmy zmienić program w 3 miejscach.

Wykorzystywanie tablic

Zostałeś poproszony o napisanie programu, który zliczałby odpowiedzi na ankietę zamieszczoną w pewnym piśmie. Oto pytania, jakie zadano czytelnikom:

Co Twoim zdaniem powinno się znaleźć w naszym piśmie ?
1. recenzje gier,
2. opisy sprzętu,
3. kursy programowania,
4. nowości ze świata komputerów.

A oto program realizujący to zadanie:

```
#include <iostream>

using namespace std;

int main()
{
 // inicjalizacja zmiennych
 // najpierw inicjalizujemy stałą symboliczną określającą wielkość tablicy
 // potem inicjalizujemy tablicę liczb typu int o nazwie tabOdp
 // tablica ma 4 elementy

 const int wTab=4;
 int tabOdp[wTab]={0}, licz;

 // wyświetlamy informacje dla użytkownika

 cout<<"Co Twoim zdaniem powinno się znalezc w naszym pismie ?\n"
                <<"1. recenzje gier\n2. opisy sprzętu,\n3. kursy programowania,\n"
                <<"4. nowości ze świata komputerów.\nWprowadz -1 aby zakończyć.";

 cin>>licz;
```

```
// pętla odpowiedzialna za zliczanie odpowiedzi
// 1. sprawdzamy jaka wartość została wprowadzona przez użytkownika
// -jeśli użytkownik wprowadził mniej niż 1 lub więcej niż 4 to przerywamy pętlę

while(licz>0 && licz<5)
{
  cout<<"Co Twoim zdaniem powinno się znalezc w naszym pismie ?\n"
  <<"1. recenzje gier\n2. opisy sprzętu,\n3. kursy programowania,\n"
  <<"4. nowości ze świata komputerów.\nWprowadz -1 aby zakończyć."<<endl;

  // zliczenie odpowiedzi indeks tablicy jest obliczany na przez wyrażenie licz-1
  // zwiększenie elementu tablicy przez operator ++
  // wprowadzenie kolejnej odpowiedzi

  ++tabOdp[licz-1];
  cin>>licz;
}

// wyświetlamy wyniki

cout<<"Wynik ankiety :\n"
                <<"1. recenzje gier :"<<tabOdp[0]<<"\n2. opisy sprzętu :"<<tabOdp[1]
                <<"\n3. kursy programowania :"<<tabOdp[2]
                <<"\n4. nowości ze świata komputerów :"<<tabOdp[3]<<endl;
 return 0;
}
```

Generowanie liczb losowych

Do generowania liczb losowych służy funkcja rand(). Funkcja ta generuje liczbę całkowitą między 0 i RAND_MAX (stałą symboliczną zdefiniowaną w pliku nagłówkowym cstdlib). Jeśli funkcja ta wybiera liczby naprawdę losowo, to każda z nich ma taką samą szansę bycia trafioną. Zakres wartości wytwarzanych przez funkcję rand() jest bardzo duży. Zwykle nie potrzebujemy aż tylu wartości. Na przykład program, który będzie symulował rzuty kostką będzie potrzebował sześć wartości. Chcąc wytworzyć liczby całkowite w zakresie od 0 do 5 używamy dzielenia modulo (%) w połączeniu z funkcją rand:

```
1+rand()%6
```

Nazywamy to skalowaniem. Liczba 6 jest czynnikiem skalowania. A liczba 1 jest wartością przesunięcia. Wartość przesunięcia to liczba równa pierwszej liczbie w pożądanym zakresie kolejnych liczb całkowitych.

Oto program symulujący rzut kostką 20 razy:

```
#include <iostream>
#include <cstdlib>

using namespace std;

int main()
{
        for(int i=1; i<20; i++)
                cout<<(1+rand()%6)<<'\n';

        cout<<endl;

        return 0;
}
```

Jeśli uruchomisz ten program kilka razy zobaczysz, że wyświetlana jest taka sama sekwencja wartości. Jest to spowodowane tym, że funkcja rand() generuje liczby pseudolosowe. Konieczne w tym wypadku jest losowanie, które jest realizowane przez funkcje srand(). Funkcja srand() pobiera jako argument liczbę unsigned int i wywołuje funkcję rand() do rozpoczęcia wytwarzania różnych sekwencji liczb losowych. Jeżeli za każdym razem chcemy losować bez konieczności wprowadzania liczby bazowej do generowania liczb losowych, możemy użyć wyrażenia:

```
srand(time(0));
```

Wyrażenie to odczytuje zegar, aby otrzymać wartość bazową dla generowania liczb losowych. Funkcja time z argumentem 0 zwraca czas jaki upłynął w sekundach od 1 stycznia 1970 roku. Oto program zliczający rzut kostką 5000 razy. Wykorzystujący w tym celu tablicę:

```
#include <iostream>
#include <cstdlib>
```

```
using namespace std;

int main()
{
        const int wTab=7;
        int czestosc[wTab]={0};
        srand(time(0));
        //ignoruj element zero w tablicy
        for(int rzut=1; rzut<=5000; rzut++)
                ++czestosc[1+rand()%6];

        cout<<"Scianka : "<<" czestosc wystepowania \n";
        for(int scianka=1;scianka<wTab;scianka++)
                cout<<scianka<<" "<<czestosc[scianka]<<endl;

        cin.get();
        return 0;
}
```

Przekazywanie tablic do funkcji

Chcąc przekazać tablicę do funkcji, piszemy nazwę tablicy bez nawiasów, np.:

```
#include <iostream>

void wyswietlTablice (int [], int);

using namespace std;

int main()
{

        const int wTab=10;
        int tab[ wTab ]={1,4,7,5,100,150,9,34};

        cout<<"Wywolanie funkcji wyswietlTablice"<<endl;

        wyswietlTablice(tab, wTab);
        cin.get();
```

```
        return 0;
}

void wyswietlTablice (int tablica [], int wielkosc)
{
 for(int i=0; i<wielkosc; i++)
   cout<<"Element tablicy : "<<(i+1)
     <<" Wartosc elementu :"<<tablica[i]<<endl;
}
```

Przekazując tablicę do funkcji, przekazuj jednocześnie jej rozmiar. Dzięki temu funkcja może przetworzyć odpowiednią ilość elementów. W przeciwnym wypadku musielibyśmy informacje o wielkości tablicy umieścić wewnątrz funkcji lub określić jako zmienną globalną, co jest niezgodne z zasadą najmniejszego przywileju.
C++ automatycznie przekazuje tablice do funkcji przez referencję (a jak się przekonamy później wskaźnik), w związku z tym funkcja może modyfikować elementy tablicy bezpośrednio. Jest to spowodowane względami wydajnościowymi. Jeśli tablicę przekazywalibyśmy przez wartość, kopiowanie wszystkich elementów tablicy zabierałoby dużo czasu i potrzebowałoby dużo pamięci. Poszczególne elementy tablicy są przekazywane są przez wartość, w związku z tym funkcja nie modyfikuje elementu bezpośrednio, a operuje na kopii wartości bezpośrednio.

Poniższy program demonstruje przekazywanie tablic i poszczególnych elementów do tablic:

```
 #include <iostream>

void wyswietlTablice (int [], int);
void modyfikujTablice (int [], int);
void modyfikujElement (int );

using namespace std;

int main()
{

  const int wTab=10;
  int tab[ wTab ]={1,4,7,5,100,150,9,34};

  cout<<"Wywolanie funkcji wyswietlTablice przed modyfikacja\n";
  wyswietlTablice(tab, wTab);
```

```
 modyfikujTablice(tab,wTab);

 cout<<”Tablica po modyfikacji :\n”;
 wyswietlTablice(tab, wTab);

 cout<<”Element 3 tablicy przed wywolaniem funcji modyfikujElement :”<<tab[3]<<endl;
 modyfikujElement(tab[3]);

 cout<<”Element 3 tablicy po wywolaniu funcji modyfikujElement : “<<tab[3]<<endl;
 cin.get();
  return 0;
}

void wyswietlTablice (int tablica [], int wielkosc)
{
        for(int i=0; i<wielkosc; i++)
                cout<<”Element tablicy : “<<(i+1)<<” Wartosc elementu :
“<<tablica[i]<<endl;
}
void modyfikujTablice (int tablica [], int wielkosc )
{
        for(int i=0; i<wielkosc; i++)
                tablica[i]=tablica[i]+10;
}

void modyfikujElement (int a)
{
        a*=2;
        cout<<”Wewnatrz funkcji modyfikujElement :”<<a<<endl;
}
```

Sortowanie tablic

Sortowanie danych jest jednym z najważniejszych zadań obliczeniowych. Powstało wiele wydajnych algorytmów, których zadaniem jest sortowanie tablic. Tutaj przedstawię dwa najprostsze algorytmy sortowania: bąbelkowe i przez selskcję.

Sortowanie bąbelkowe

Sortowanie bąbelkowe wymaga kilku przejść przez całą tablicę. W każdym przejściu tablicy kolejne pary są porównywane. Jeżeli para znajduje się w porządku rosnącym lub wartości są identyczne, ich wartości są pozostawione. Jeśli para znajduje się w porządku malejącym, wartości zamieniane są miejscami.

```
#include <iostream>

void wyswietlTablice (int [], int);
void sortBab(int [], int);

using namespace std;
int main ()
{
        const int wTab=10;
        int tab[ wTab ]={1,4,7,5,100,150,9,34,-1,120};

        cout<<”Tablica przed posortowaniem :\n”;
        wyswietlTablice(tab,wTab);
        sortBab(tab,wTab);
        cout<<”Tablica po posortowaniu :\n”;
        wyswietlTablice(tab,wTab);
        cin.get();
        return 0;
}

void wyswietlTablice (int tablica [], int wielkosc)
{
        for(int i=0; i<wielkosc; i++)
                cout<<”Element tablicy : “<<(i+1)<<” Wartosc elementu :
“<<tablica[i]<<endl;
}

void sortBab (int tablica [], int wielkosc )
{
        int temp;
```

```
        for (int i=0; i<wielkosc-1; i++)
        {
                for (int j=0; j<wielkosc-i; j++)
                {
                        if(tablica[j]>tablica[j+1])
                        {
                                temp=tablica[j];
                                tablica[j]=tablica[j+1];
                                tablica[j+1]=temp;
                        }
                } // for(j=0..
        } // for(i=0..
}
```

Sortowanie przez selekcję

Algorytm ten jest następujący:
1. należy znaleźć najmniejszy element i zmienić go miejscem z pierwszym,
2. znaleźć drugi element w kolejności i zamienić go miejscem w drugim,
3. kontynuować krok drugi, aż do posortowania.

```
#include <iostream>

void wyswietlTablice (int [], int);
void sortSel(int [], int);

using namespace std;

int main ()
{

        const int wTab=10;
        int tab[ wTab ]={1,4,7,5,100,150,9,34,-1,120};

        cout<<”Tablica przed posortowaniem :\n”;
        wyswietlTablice(tab,wTab);
        sortSel(tab,wTab);
        cout<<”Tablica po posortowaniu :\n”;
        wyswietlTablice(tab,wTab);
        cout<<”Nacisnij ENTER”<<endl;
```

```
        cin.get();
        return 0;
}

void wyswietlTablice (int tablica [], int wielkosc)
{
        for(int i=0; i<wielkosc; i++)
        cout<<"Element tablicy : "<<(i+1)<<" Wartosc elementu : "<<tablica[i]<<endl;
}
void sortSel (int tablica [], int wielkosc )
{
 void zmien( int [], int, int);
 int min;

 for (int i=0; i<wielkosc; i++)
        {
        min=i;
         for(int j=i+1; j<wielkosc; j++)
                {
                if(tablica[j]<tablica[min])
                        {
                                min=j;
                        }
                }
         zmien(tablica,i,min);
 }
}

void zmien (int tablica [], int indeks, int indeks1)
{
        int temp;
        temp=tablica[indeks];
        tablica[indeks]=tablica[indeks1];
        tablica[indeks1]=temp;
}
```

Tablice wielowymiarowe

W C++ tablice mogą być wielowymiarowe. Typowym ich zastosowaniem jest przedstawienie informacji pogrupowanych w wierszach i kolumnach. W tablicach wielowymiarowych, chcąc wskazać element tablicy, musimy określić dwa indeksy: pierwszy oznacza wiersz, a drugi kolumnę.
Tablice wymagające dwóch indeksów do zidentyfikowania określonego elementu są nazywane tablicami dwuwymiarowymi. Mogą mieć one więcej niż dwa indeksy. Kompilatory C++ dopuszczają 12 indeksów tablicy. Wielowymiarowa tablica może być zainicjowana w podobny sposób do tablicy jednowymiarowej. Chcąc utworzyć dwuwymiarową tablicę typu int, mającą dwa wiersze i dwie kolumny, napiszemy (jednocześnie incjując):

```
int tabInt[2][2]={{1,1},{2,2}};
```

Wartości w nawiasach klamrowych zgrupowane są według wierszy. Chcąc przekazać tablicę wielowymiarową do funkcji, musimy określić wielkość drugiego i ewentualnie następnych indeksów. Kompilator używa tych wielkości do określenia położenia w pamięci elementów wielowymiarowej tablicy. Przykładowo: chcąc przekazać do funkcji wyswTab() tablicę tabInt, musielibyśmy tę funkcję zadeklarować:

```
void wyswTab(int [][2],int );
```

A pierwsza linia definicji funkcji wyglądałaby tak:

```
void wyswTab(int tablica[][2], int wielkosc)
```

Ćwiczenia

1. Sortowanie bąbelkowe jest mało wydajne - dopracuj ten algorytm. Aby poprawić wydajność (przecież dane w tablicy mogą być już w całości lub w części posortowane), zmodyfikuj sortowanie tak, aby na końcu każdego przebiegu sprawdzano czy zostały dokonane jakieś zmiany, jeżeli nie - dane są już uporządkowane i należy zakończyć sortowanie.
2. Pewne przedsiębiorstwo płaci swoim sprzedawcom wynagrodzenie zależnie od wartości sprzedaży. Sprzedawcy otrzymują 690 zł + 10% swojej sprzedaży brutto w tym miesiącu. Napisz program obliczający pensję sprzedawcy w zależnośći od sprzedaży.
3. Napisz algorytm sortowania przez selekcję w sposób rekurencyjny.
4. Napisz program symulujący rzuty dwiema kostkami 100 razy. Powinien on wykorzystywać funkcję rand() w połączeniu z funkcją srand(). Suma tych 36 (od 2 do 12) kombinacji winna być

wyświetlona w wierszach i kolumnach jak poniżej.

Źródła:

- Andrzej Stasiewicz, C++ Ćwiczenia praktyczne, Helion, Gliwice, 2010.
- http://cpp0x.pl/
- http://www.ithelpdesk.pl/
- http://pl.wikibooks.org/wiki/C++

Delphi

Delphi – język programowania, którego można używać w środowiskach firmy Borland, Embarcadero, Microsoft (Delphi Prism) oraz w środowisku Lazarus. Narzędzia te są zintegrowanymi środowiskami programistycznymi typu RAD.

Dawniej język Delphi był nazywany Object Pascalem, lecz w świadomości społecznej programistów na całym świecie Delphi zaczęło być postrzegane z czasem jako osobny język programowania. Przy okazji premiery Delphi 6 w roku 2002 r., w oficjalnej dokumentacji programu, została użyta po raz pierwszy nazwa „Delphi language". Standard języka Delphi obejmuje wiele bogatych funkcjonalnie klas, których nie ma w standardzie Object Pascala, a ponadto umożliwia programowanie wizualne, czyli Object-Oriented Design. Object Pascal jest rozszerzeniem języka Pascal.

Programy tworzone w Delphi dla Win32 muszą zostać skompilowane do postaci kodu binarnego przed pierwszym wykonaniem. Niektóre komponenty wizualne działają już podczas tworzenia projektu, umożliwiając oglądanie efektów pracy. Delphi dla Win32 zapisuje informacje o właściwościach obiektów, udostępniając je programiście. Informacje te umożliwiają zmianę ich wartości przez programistę bez pisania kodu programu oraz są używane podczas pracy programu – technika ta zwana jest RTTI. Tworzone programy pracują na zasadzie obsługi zdarzeń, każde polecenie (np. kliknięcie myszką) generuje zdarzenie, które poprzez wewnętrzne mechanizmy programu są przesyłane do odpowiedniego komponentu, a rolą programisty jest tylko dołączenie odpowiedniego kodu umożliwiającego obsługę tego zdarzenia. Natywne programy tworzone w Delphi pod Win32 umożliwiają w prosty sposób stworzenie wydajnej aplikacji, sam język jest przyjazny użytkownikowi i podobny do języka C# dla platformy .NET

Odpowiednik Delphi produkcji firmy Borland dla Linuksa nosił nazwę Kylix. Jego produkcja została zaniechana, gdyż charakteryzował się niską jakością. Obecnie rozwijane jest na licencji GNU GPL zintegrowane środowisko programistyczne (IDE) o nazwie Lazarus, wzorowane na Delphi. Lazarus jest środowiskiem wieloplatformowym, dostępnym dla różnych systemów operacyjnych.

W przeciwieństwie do języka Pascal, Delphi nie było tworzone dla celów edukacyjnych, lecz biznesowych, jako język, który miał połączyć prostotę i przejrzystość języka Pascal z łatwym i wygodnym tworzeniem aplikacji. Delphi umożliwia również niskopoziomowe programowanie poprzez możliwość wstawiania części kodu napisanego w języku assembler. Jest językiem obiektowym.

Cechy i funkcjonalność:

- wspomaganie dla obsługi relacyjnych systemów bazodanowych,
- obsługa standardowych mechanizmów windowsowych,
- szeroki zestaw gotowych do użycia komponentów,
- rozszerzalność środowiska (zarówno palety komponentów, jak i samego IDE),
- budowa wizualnej części aplikacji za pomocą techniki drag and drop,

• zawiera wiele elementów mających na celu uproszczenie tworzenia aplikacji związanych z Internetem,
• szybki, efektywny kompilator,
• zawiera wiele dodatkowych narzędzi wspomagających programistów.

Środowisko Delphi wraz z dołączonymi narzędziami może być uznane za język czwartej generacji.

Środowisko użytkowników

Delphi cieszy się w Polsce stosunkowo dużą popularnością, w głównej mierze ze względu na prostotę i powszechność różnego rodzaju poradników dla początkujących.

Ważne uwagi

1. Podczas pisania programu źródłowego w Delphi wiele linii kodu generuje samo środowisko Delphi.
2. W systemie Turbo Pascal plik źródłowy ma rozszerzenie *.pas.
3. Program napisany w języku Delphi jest nazywany projektem. Środowisko Delphi dla każdego projektu tworzy wiele plików. Nazwa pliku składa się z dwóch elementów: nazwy nadanej projektowi i jego modułom oraz predefiniowanego rozszerzenia stosowanego przez Delphi.
4. Pliki źródłowe napisane w języku Delphi mają różne rozszerzenia (nie tylko *.pas).
5. Podczas tworzenia nowego projektu w trybie okienkowym Delphi zakłada 6 - 9 plików, m. in.:
- plik źródłowy projektu, który zawiera kod wykonywany podczas uruchamiania aplikacji,
- moduł formularza głównego, zawierający deklarację i definicję klasy formularza głównego,
- plik zasobów formularza głównego,
- plik zasobów projektu.

Pojęcie projektu

Projekt jest to zestaw plików, których przetworzenie przez Delphi daje w efekcie gotowy program (plik wykonywalny). Skompilowanie i połączenie plików projektu może dać w efekcie jeszcze inne obiekty, np. bibliotekę DLL. Najważniejszymi elementami projektu Delphi (okienkowego) są pliki źródłowe modułów, będących głównymi składnikami funkcjonalnymi programu (np. Unit1.pas oraz Unit1.dfm).

Start systemu Delphi7

1. Wybrać: Start | Programy | Borland Delphi 7 | Delphi 7
2. Wejść do katalogu C:\Program Files \ Borland \ Delphi7 \ Bin i wybrać plik delphi32.exe
3. Na pulpicie utworzyć ikonę skrótu do programu.

Okno programu Delphi

Po uruchomieniu programu na ekranie pojawia się zestaw okien i innych elementów tworzących tzw. zintegrowane środowisko robocze – IDE (Integrated Development Environment). Układ okien, charakterystyczny dla narzędzi RAD firmy Borland, odzwierciedla samą metodę tworzenia aplikacji – zamiast okna edytora, w którym wpisuje się tekst programu, centralne miejsce na ekranie zajmuje formularz, na którym ustawia się komponenty.

Elementy IDE:

• okno zatytułowane Form1 – główne pole robocze – formularz główny (main form) programu (aplikacji), reprezentuje okno aplikacji i stanowi „podłoże", na którym umieszcza się komponenty, wybierane z palety komponentów i umieszczane na formularzu poprzez klikanie myszą; proste programy zawierają zwykle pojedynczy formularz (formularz główny),
• paleta komponentów, zawierająca kilka kart obiektowych, np. Standard, Additional, Win32, System, Dialogs, komponenty przeznaczone do obsługi baz danych, komponenty internetowe, itd.,
• okno hierarchii komponentów Object TreeView – prezentuje ono w formie graficznej wszystkie komponenty zawarte na formularzu oraz zależności między nimi,
• okno inspektora obiektów Object Inspector – zawierające dwie karty: kartę Properties (lista właściwości – ich nazwy i wartości) i kartę Events (lista zdarzeń i procedur ich obsługi),
• okno edytora kodu źródłowego o nazwie Unit1.pas zawierające kod źródłowy modułu o nazwie Unit1, który to moduł jest częścią składową całego projektu (treść programu),
• pasek tytułu – „Delphi 7 – Project1",
• pasek menu (menu główne),
• paleta narzędzi – przyciski realizujące najczęściej wykonywane operacje:

górny rząd

New items - utwórz nowy projekt
Open - otwórz plik
Save - zapisz plik
Save all - zapisz wszystkie pliki
Open Project - otwórz projekt
Add file to Project - dodaj plik do projektu
Remove file from Project - usuń plik z projektu
Help contents - pomoc

dolny rząd:

View Unit - wyświetl listę modułów
View Form - wyświetl listę formularzy
Toggle Form / Unit - przełącz tekst/formularz

New Form - utwórz nowy formularz
Run - uruchom program
Pause - wstrzymaj program
Trace into - wykonaj następną instrukcję
Step over - wykonaj następną procedurę

Pojęcie komponentu

Komponent jest to element programu, realizujący konkretne zadanie (umożliwiający wprowadzanie lub wyprowadzanie danych, obsługujący połączenie z serwerem WWW, odmierzający czas, wyświetlający zawartość folderu, itd.).

Zapisywanie plików w trybie okienkowym

1. IDE proponuje różne katalogi w zależności od sposobu uruchomienia (Start | Programy | … czy ikona skrótu):
C:\Program Files \ Borland \ Delphi7 \ Projects lub
C:\Program Files \ Borland \ Delphi7 \ Bin
2. Delphi nie zapamiętuje nazwy katalogu użytego do zapisania ostatniego projektu; każdorazowo trzeba wskazywać katalog docelowy.
3. Zapisywanie projektów Delphi odbywa się dwuetapowo:
Save Unit1 As
Save Project1 As
w sumie min. 6 plików:
Unit1.pas – tekst programu w Pascalu,
Unit1.dfm – opisuje układ i właściwości formularza i wszystkich zawartych w nim obiektów
pliki zawierające informacje o ustawieniach projektu i tzw. zasobach

Project1.dof
Project1.dpr
Project1.cfg
Project1.res

Najważniejszymi elementami projektu są pliki źródłowe modułów, będących głównymi składnikami funkcjonalnymi programu.

Pasek menu (menu główne)

File (plik) – manipulowanie plikami i projektami
New
Application – utworzenie nowego programu
……..

Other – przejście do aplikacji konsolowej
Open – otwórz plik
Open Project – otwórz projekt
Reopen - lista ostatnio otwieranych plików
Save - zapisz plik
Save As - zapisz jako
Save Project As - zapisz projekt jako
Save All – zapisz wszystkie otwarte pliki
Close
Close All
Use Unit
Print
Exit

Edit (edycja) - typowe funkcje służące do edycji tekstu programu i zawartości formularza
Undelete – cofnięcie operacji
Redo – powtórzenie operacji
Cut - wycinanie
Copy - kopiowanie
Paste - wklejanie
Delete -- usuwanie
Select All – wybieranie elementów
polecenia zawarte w dolnej części menu służą do manipulowania komponentami umieszczonymi na formularzu
Align to Grid
……
Lock Controls

Search (szukaj)
Find – wyszukiwanie tekstu
Find in Files – przeszukiwanie kilku wybranych plików
Replace – zamiana tekstu
Incremental Search – wyszukiwanie tekstu pasującego do kolejno wpisywanych znaków

View (widok) – zarządzanie oknami wchodzącymi w skład IDE
Project Manager - wyświetla zawartość projektu
Object Inspector
Object TreeView

Alignment Palete – umożliwia manipulowanie komponentami na formularzu
Component List
Window List – wyświetla listę otwartych okien
Additional Message Info
Debug Windows – udostępnia obszerne menu okien podsystemu uruchomieniowego (debuggera)
Desktops
Toggle Form / Unit – przełącza pomiędzy oknami formularza i edytora kodu
Units – wyświetla listę modułów zawartych w projekcie
Forms – wyświetla listę formularzy zawartych w projekcie
New Edit Window
Toolbars – konfiguruje zwartość okna głównego

Project (projekt) – polecenia służące do zarządzania projektami
Add to Project – dodaj wybrany moduł do projektu
Remove from Project– usuń wybrany moduł z projektu
Import Type Library
Add to Repository
View Source
polecenia umożliwiające tworzenie tzw. grup składających się z kilku powiązanych ze sobą projektów
Add New Project Add Existing Project
polecenia pozwalające przetworzyć tekst źródłowy programu na postać wykonywalną
(polecenia kompilacji)
Compile Project1
Build Project1
Syntax check Project1 – sprawdzanie poprawności tekstu źródłowego polecenia kompilacji wszystkich projektów wchodzących w skład grupy
Compile All Project
Build All Project
Options – okno umożliwiające zmianę ustawień projektu

Run (uruchomienie)
Run – uruchom program
Parameters
Step Over - wykonaj następną procedurę
Trace Into - wykonaj następną instrukcję
Trace to Next Source Line

Run to Cursor
Run Until Return
Show Execution Point
Program Pause – wstrzymuje pracę programu
Program Reset – przerywa wykonanie programu
Evaluate / Modify
Add Watch
Add Breakpoint

Component (komponent) - zarządzanie komponentami
New Component
Install Component
Import ActiveX Control
Create Component Template
Install Packages
Configure Palette

Tools (narzędzia) – udostępnia funkcje konfiguracyjne oraz programy narzędziowe
Environment Options – zmiana ustawień całego IDE
Editor Options – zmiana ustawień samego edytora kodu
Debugger Options
Repository
Configure Tools
Image Editor

Window (okno) – spis aktualnie otwartych okien

Help (pomoc)

System Delphi 7 - aplikacje konsolowe - możliwość tworzenia tradycyjnych programów pracujących w trybie tekstowym:

Opis postępowania
1. Uruchomić program Delphi
np. Start | Programy | Borland Delphi 7 | Delphi 7.
2. Pojawia się standardowe okno programu Delphi.
Przejść do trybu tekstowego (aplikacji konsolowej)
3. Wybrać polecenie File | New | Other ….

Na ekranie pojawi się okno New Items, reprezentujące tzw. składnicę obiektów (Object Repository).
Składnica obiektów jest w Delphi magazynem zawierającym kreatory i szablony programów oraz ich składników (formularzy, bibliotek, okien, modułów danych oraz innych elementów związanych z bazami danych i Internetem). Służy ona jako punkt wyjścia do tworzenia projektów i zawiera kilka kart (New, Project1, Forms, Dialogs, Projects).
4. Przejść na kartę New, skupiającą najpopularniejsze elementy aplikacji i narzędzia. Interesuje nas tzw. aplikacja tekstowa lub konsolowa (console application).
5. Aby utworzyć odpowiedni projekt należy na karcie New kliknąć dwukrotnie ikonę Console Application.
6. Utworzony w ten sposób projekt nie zawiera formularza, posiada tylko okno edytora, a okna Object TreeView i Object Inspector są puste. Na pasku tytułu pojawia się nazwa Project2.dpr. W oknie edytora pojawia się następująca treść programu:

```
program Project2;

{$APPTYPE CONSOLE}
        (dyrektywa określająca rodzaj aplikacji - tekstowa lub okienkowa)

uses SysUtils;                  (lista modułów wykorzystywanych przez program)


begin
  { TODO –oUser –cConsole Main : Insert code here }
end.
```

Powyższe 10 linijek to kompletny program w Pascalu, który co prawda nie robi nic, ale daje się uruchomić (np. klawiszem F9) – wyświetla przez chwilę puste okno.
Komentarz zaczynający się od słowa TODO (do zrobienia) można usunąć.
Pomiędzy słowa kluczowe begin i end wpisać instrukcje w języku Pascal (program źródłowy w języku Pascal).
7. Zapisać projekt w swoim katalogu:
polecenie File | Save Project As ….
pod wybraną, opisową nazwą.
8. Skompilować i uruchomić program:
- polecenie Run | Run,
- klawisz F9,
- przycisk uruchomienia - zielona strzałka na palecie narzędzi (dolny pasek narzędziowy).

9. Kompilację i uruchomienie programu można przeprowadzić dwuetapowo:
- polecenie Project | Compile nazwa_projektu, <Ctrl + F9>
- polecenie Run | Run <F9>
10. Zamknąć aplikację
- polecenie File | Exit,
- przycisk zamknięcia głównego okna Delphi „X”,
- klawisze Alt+F4
11. Otwarcie istniejącego pliku *.dpr przygotowanego w trybie tekstowym powoduje automatyczne przejście programu Delphi do trybu aplikacji konsolowej.

W aplikacji konsolowej (tekstowej) najważniejszym plikiem jest plik *.dpr, zawierający kod źródłowy programu Pascal.

W systemie Turbo Pascal standardowo tworzony był jeden plik (*.pas), zawierający kod źródłowy.

Podczas tworzenia nowego projektu w trybie tekstowym Delphi zakłada 4 pliki:
1. Plik źródłowy projektu (*.dpr), który zawiera kod wykonywany podczas uruchamiania aplikacji.
2. Wykonywalny program wynikowy (*.exe).
3. Plik konfiguracyjny projektu (*.cfg), w którym zawarte są główne ustawienia kompilatora i konsolidatora.
4. Plik opcji projektu (*.dof), w którym zawarte są bieżące ustawienia wszystkich opcji projektu.

Zapisywanie plików w trybie tekstowym

1. IDE proponuje różne katalogi w zależności od sposobu uruchomienia (Start | Programy | … czy ikona skrótu):

C:\Program Files \ Borland \ Delphi7 \ Projects lub
C:\Program Files \ Borland \ Delphi7 \ Bin

2. Delphi nie zapamiętuje nazwy katalogu użytego do zapisania ostatniego projektu; każdorazowo trzeba wskazywać katalog docelowy.
3. Zapisywanie projektu - polecenie:

File | Save Project As

Przykładowe materiały do wykorzystania na lekcjach z przedmiotu „Programowanie strukturalne i obiektowe" w technikum informatycznym. Materiały zostały opracowane na przykładzie języka programowania Delphi.

Przedmiot: Programowanie strukturalne i obiektowe
Temat: Przyciski opcji
Czas: 2 jednostki lekcyjne
Cele lekcji:
- zapoznanie z nowymi komponentami,
- zapamiętanie informacji o ich cechach, przeznaczeniu i możliwościach wykorzystania,
- ćwiczenie umiejętności samodzielnego tworzenia programów z wykorzystaniem poznanych komponentów,
- wyrabianie i kształtowanie umiejętności samodzielnego uczenia się.

Przycisk opcji (ang. radio button) jest to widżet, element graficznego interfejsu użytkownika programów komputerowych.
Przycisk opcji posiada dwa stany: włączony i wyłączony, a „naciśnięcie" go lewym klawiszem myszy zawsze spowoduje jego włączenie. W momencie aktywacji deaktywuje on pozostałe przyciski radiowe z grupy - widżet ten to grupa przycisków wzajemnie wykluczających się.
Zdarzenia generowane są spod każdego przycisku; zawsze generowane jest zdarzenie aktywacji przycisku, bywa że jest też generowane zdarzenie deaktywacji wcześniej aktywnego przycisku. Jeśli biblioteka jest wyposażona w mechanizm sygnałów i slotów, to zwykle podłącza się wszystkie sygnały aktywacji każdego przycisku z grupy do jednego slotu, gdyż wszystkie przyciski zwykle są stowarzyszone z jednym stanem w używającym je oknie.
Widżet [wym. łidżet] to podstawowy element graficznego interfejsu użytkownika (GUI), np. okno, pole edycji, suwak lub przycisk. Termin ten jest szczególnie popularny wśród użytkowników systemów operacyjnych z rodziny UNIX, natomiast użytkownicy systemów MS Windows używają w tym kontekście terminu kontrolka lub element kontrolny. W pewnych kontekstach synonimem widżetu jest okno.
Graficzny interfejs użytkownika (ang. Graphical User Interface, GUI) często nazywany też środowiskiem graficznym – ogólne określenie sposobu prezentacji informacji przez komputer oraz interakcji z użytkownikiem, polegającego na rysowaniu i obsługiwaniu widgetów.

Zadanie 1
Przeczytaj uważnie powyższe definicje. Zastanów się, czy spotkałeś się już z przyciskami opcji w programach komputerowych. Zanotuj w zeszycie temat lekcji oraz sporządź notatkę z definicją przycisku opcji. (10 minut)

RadioButton – przycisk opcji w Delphi

Przyciski budowane za pomocą komponentu RadioButton służą do wyboru jednego spośród kilku elementów. Możemy je umieszczać na formie, aczkolwiek najlepiej to zrobić w oknie komponentu GroupBox.

Uwaga! Przy korzystaniu z komponentu RadioButton zwróć uwagę na nową właściwość Checked. Przybiera ona wartość True (gdy komponent jest zaznaczony) lub False.

RadioGroup – grupa przycisków opcji

Komponent RadioGroup pozwala na generowanie przycisków opcji. Nie musimy ich umieszczać na formie w sposób ręczny. Do tego zadania służy narzędzie String List Editor, które uaktywnia się po kliknięciu pola właściwości Items.

Przy obsłudze korzystamy ze zdarzenia OnClick dla RadioGroup i wykorzystujemy właściwość ItemIndex, podającą numer wybranego przycisku. Kolejne przyciski w grupie są numerowane od 0, a więc pierwszy element ma numer 0, drugi ma numer 1 itd. Zapisując operacje związane z wyborem poszczególnych opcji, możemy wykorzystać instrukcję case. W danej chwili może być wybrana co najwyżej jedna opcja. Gdy żadna opcja nie jest wybrana, to ItemIndex = -1. Komponent nie reaguje na kliknięcie, jeśli wybieramy opcję już zaznaczoną. Aby móc ponownie wybrać tę samą opcję, możemy programowo zmienić wartość ItemIndex: RadioGroup1. ItemIndex:= -1; // usuwa zaznaczenie opcji.

Zadanie 2

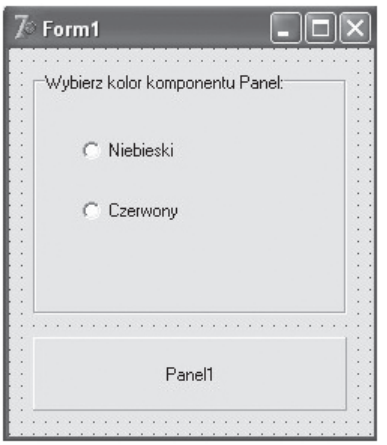
Przeczytaj powyższy tekst. Odszukaj opisane komponenty i wstaw je na formularzu. Przejrzyj ich właściwości, starając się zwrócić szczególną uwagę na wyżej opisane (plus columns) oraz te, które dotyczyły komponentów, które już znasz. Uruchom utworzony program i przetestuj zachowanie się przycisków opcji. Uzupełnij notatkę o definicje wymienionych komponentów oraz listę istotnych ich cech. (15 minut)

Zadanie 3

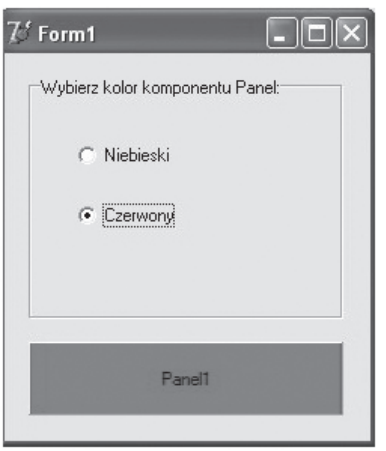
Utwórz program, który zawiera dwa przyciski opcji umieszczone w komponencie GroupBox oraz komponent Panel na zakładce Standard. Przyciski opisz dowolnymi nazwami kolorów (np. niebieski i czarny) - po wyborze jednego z przycisków kolor komponentu Panel będzie się zmieniał. Zadanie możesz wykonać samodzielnie lub skorzystać z podanego rozwiązania. Jeżeli uznasz to za słuszne uzupełnij notatkę w zeszycie. Staraj się dobrze zrozumieć zasadę działania programu oraz wykorzystanych komponentów i instrukcji. (15 minut)

1. Umieść na formie komponent GroupBox oraz dwa przyciski opcji RadioButton.
2. Zmień napis na przyciskach opcji na nazwy wybranych przez siebie kolorów.

3. Umieść na formie komponent Panel. Wszystkie komponenty umieść tak, jak na rysunku:



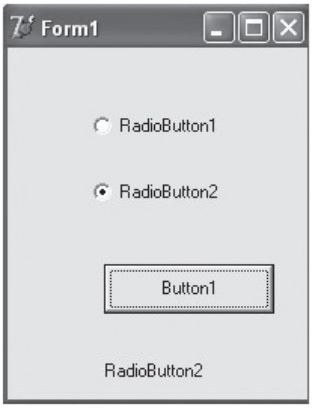
4. Podepnij pod przyciski opcji instrukcje według wzoru:
Panel1.Color:=ClBlue; {Po znaku równości wpisz nazwę koloru}
5. Sprawdź działanie aplikacji (Rysunek 2)



Zadanie 4

Napisz aplikację, która będzie zawierać dwa przyciski opcji, komponent przycisku oraz etykietę. Po naciśnięciu na etykiecie będzie wyświetlać się nazwa zaznaczonego przycisku opcji. Zadanie możesz wykonać samodzielnie lub skorzystać z podanego rozwiązania. Jeżeli uznasz to za słuszne, uzupełnij notatkę w zeszycie. (10 minut)

1. Wstaw na formę dwa przyciski opcji, przycisk oraz komponent Label (etykietę).
2. Ustaw właściwość Checked komponentu RadioButton1 na True.
3. Podepnij pod przycisk instrukcję warunkową:




```
if RadioButton1.Checked then
label1.Caption:=RadioButton1.name
else {przed tą instrukcją nie wpisujemy znaku średnika}
label1.Caption:=RadioButton2.name;
```

Powyższą instrukcję określa się mianem instrukcji warunkowej (if ... then ... else ...).

4. Sprawdź działanie programu.

Zadanie 5

Napisz aplikację, która będzie zawierać pole RichEdit (karta Win32) oraz grupę przycisków opcji opisanych nazwami 5 kolorów. Przyciskami opcji wybieramy kolor czcionki w oknie RichEdit. Nazwy wybranych kolorów wyświetlane są w etykiecie.
Podpowiedź: Wykorzystaj cechę Items do wyświetlania nazw kolorów (Radiogroup1. Items[RadioGroup1.Itemindex]) (15 minut)

Zadanie 6

W sklepie z wyposażeniem wnętrz sprzedawany jest parkiet z różnego rodzaju drewna: sosna, buk, dąb, w dwóch gatunkach: pierwszym i drugim. Cena parkietu w drugim gatunku stanowi 70% ceny gatunku pierwszego. Ustal ceny dla gatunku pierwszego. Zbuduj grupy przycisków opcji do wyboru rodzaju i gatunku parkietu. Przycisk Wyświetl cenę uruchamia procedurę wyświetlającą w oknie komunikatu cenę jednego metra wybranego parkietu. (15 minut)

Zadanie 7

Zapisz wszystkie wykonane przez siebie zadania w katalogu nazwanym swoim nazwiskiem i imieniem oraz skopiuj go na serwer (sbs2005) do folderu Programowanie 3Ti. (5 minut)

Po lekcji powinieneś umieć:

- scharakteryzować poznane komponenty,
- wykorzystać poznane komponenty w programach komputerowych,
- samodzielnie tworzyć notatki oraz budować programy komputerowe w oparciu o nowe wiadomości.

Przedmiot: Programowanie strukturalne i obiektowe
Temat: Przyciski wyboru
Czas: 1 jednostka lekcyjna
Cele lekcji:

- zapoznanie z nowymi komponentami,
- zapamiętanie informacji o ich cechach, przeznaczeniu i możliwościach wykorzystania,
- ćwiczenie umiejętności samodzielnego tworzenia programów z wykorzystaniem poznanych komponentów,
- wyrabianie i kształtowanie umiejętności samodzielnego uczenia się.

Przycisk wyboru (ang. check box) jest w działaniu bardzo podobny do przycisku przełączania, z tym tylko, że znacznie różni się od niego wyglądem.
Przycisk zwykle jest skojarzony z odpowiednią etykietą (niektóre biblioteki, np. MS Windows API, i tak implementują napis jako stan każdego widżetu), natomiast jeśli chodzi o wygląd to jest to zwykle mały prostokącik, wielkości mniej więcej jednego-dwóch znaków, który w zależności od stanu albo jest „czysty", albo jest na nim narysowany znaczek „tick" (v z wydłużonym prawym ramieniem), ewentualnie niektóre style stosują krzyżyk (zdarza się też mały prostokącik, który wygląda na wypukły lub wklęsły). Po prawej lub lewej stronie tego prostokąciku znajduje się etykieta przycisku (innych sposobów umieszczania etykiety się raczej nie spotyka).
Przycisk wyboru jest zwykle trójstanowy (oczywiście tylko od konfiguracji zależy, czy będzie on się zachowywał jak dwustanowy lub trójstanowy). W „trzecim stanie" przycisk zaznaczania wygląda zazwyczaj tak jak w stanie „zaznaczony", tyle tylko że rysunek „tick" -
a jest wyszarzony.
Przycisk ten można tylko kliknąć lewym klawiszem myszy, po czym zmienia on stan na „następny" (w przypadku przycisków dwustanowych jest to odwrócenie stanu, w przypadku trójstanowych jest to cykliczne przechodzenie w inne stany).
Zdarzenia są niemal identyczne, jak w przypadku przycisku - generowanie zdarzenia następuje w momencie zwolnienia przycisku myszy, pod warunkiem oczywiście, że zwolnienie nastąpi w momencie, gdy kursor myszy znajduje się nad prostokątem lub etykietą przycisku.

Zadanie 1

Przeczytaj uważnie powyższą definicję. Zastanów się, czy spotkałeś się już z przyciskami wyboru w programach komputerowych. Zanotuj w zeszycie temat lekcji oraz sporządź notatkę z definicją przycisku wyboru. (10 minut)

CheckBox – przycisk wyboru w Delphi
Przyciski wyboru tworzy się za pomocą komponentu CheckBox. Służą one do zaznaczania elementów grupy. W odróżnieniu od przycisków RadioButton, jednocześnie może być tu wybrany więcej niż jeden składnik.

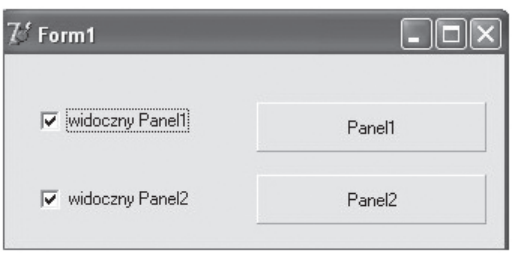
Zadanie 2

Przeczytaj powyższy tekst. Odszukaj opisany komponent i wstaw go na formularzu. Przejrzyj jego właściwości. Uruchom utworzony program i przetestuj zachowanie się przycisków wyboru. Uzupełnij notatkę o definicję tego komponentu oraz listę istotnych jego cech. (5 minut)

Zadanie 3

Utwórz program, który za pomocą komponentów CheckBox będzie uwidaczniał komponenty typu Panel. Zadanie możesz wykonać samodzielnie lub skorzystać z podanego rozwiązania. Jeżeli uznasz to za słuszne uzupełniaj notatkę w zeszycie. (10 minut)

1. Wstaw na formę dwa komponenty CheckBox oraz Panel.
2. Zmień etykiety komponentów CheckBox na widoczny Panel1 oraz widoczny Panel2.
3. Zmień właściwość Visible komponentów Panel1 oraz Panel2 na False.
4. Podepnij pod przyciski wyboru odpowiednie instrukcje:
• Dla pierwszego komponentu CheckBox:
if panel1.Visible=false then panel1.Visible:=true else panel1.Visible:=false;
• Dla drugiego komponentu CheckBox:
if panel2.Visible=false then panel2.Visible:=true else panel2.Visible:=false;
5. Sprawdź działanie programu.



Zadanie 4

Napisz aplikację, która będzie zawierać trzy przyciski wyboru oraz panel. Przyciski wyboru opisz nazwami kolorów: czerwony, zielony, niebieski. Po wyborze przez użytkownika kolorów oraz naciśnięciu przycisku „Pokoloruj panel", nastąpi malowanie panelu kolorem wygenerowanym przez makro RGB. (15 minut)

Podpowiedź: Makro RGB (r, g, b: byte) generuje kolor o podanych składowych r, g, b (red, green, blue). Wartości parametrów od 0 do 255 oznaczają intensywność określonej składowej w kolorze wynikowym. Powinieneś zadeklarować zmienne r, g i b w programie i w zależności od tego czy użytkownik zaznaczy wybrany kolor zmiennym, tym przypisywać wartość 0 lub 255. Później należy przemalować panel instrukcją: Panel1.Color:=RGB (r, g, b);

Zadanie 5

Zapisz wszystkie wykonane przez siebie zadania w katalogu nazwanym swoim nazwiskiem i imieniem oraz skopiuj go na serwer (sbs2005) do folderu Programowanie 3Ti. (5 minut)

Po lekcji powinieneś umieć:

- scharakteryzować poznane komponenty,
- wykorzystać poznane komponenty w programach komputerowych,
- samodzielnie tworzyć notatki oraz budować programy komputerowe w oparciu o nowe wiadomości.

Przedmiot: Programowanie strukturalne i obiektowe
Temat: Przyciski wyboru - ćwiczenia
Czas: 1 jednostka lekcyjna
Cele lekcji:

- zapoznanie z nowymi komponentami,
- zapamiętanie informacji o ich cechach, przeznaczeniu i możliwościach wykorzystania,
- ćwiczenie umiejętności samodzielnego tworzenia programów z wykorzystaniem poznanych komponentów,
- wyrabianie i kształtowanie umiejętności samodzielnego uczenia się.

Zadanie 1

Przypomnij sobie ostatnio poznane komponenty związane z przyciskami opcji i wyboru. Jak się one nazywają w Delhi? Jaka jest różnica w ich działaniu? Odpowiedź wraz z tematem zapisz w zeszycie. (5 minut)

Zadanie 2

Utwórz program, który za pomocą komponentów CheckBox pozwoli zmieniać styl czcionki w polu RichEdit. Dostępne style czcionek to: pogrubiona, pochylona, podkreślona i przekreślona. Po zaznaczeniu odpowiednich przycisków wyboru nastąpi zmiana stylu czcionki w polu RichEdit. (15 minut)

Podpowiedź: Dostępne style czcionki dla pola RichEdit to: fsBold – pogrubiona, fsItalic – podkreślona, fsUnderline – podkreślona, fsStrikeout – przekreślona, np.:

- dodanie do istniejącego stylu czcionki pogrubienia: RichEdit1.Font.Style:= RichEdit1.Font.Style+[fsBold];
- usunięcie z istniejącego stylu czcionki pochylenia: RichEdit1.Font.Style:= RichEdit1.Font.Style-[fsItalic];

Pamiętaj, że nazwy poszczególnych stylów należy podawać w nawiasach kwadratowych.

Zadanie 3

W oparciu o poprzedni program, dodaj do aplikacji dwie grupy przycisków opcji, ustalające w polu RichEdit kolor wyświetlanego tekstu i kolor tła. Użyj właściwości RichEdit1.Font.Color oraz RichEdit1.Color oraz stałych opisujących wybrane kolory. (15 minut)

Zadanie 4

Zapisz wszystkie wykonane przez siebie zadania w katalogu nazwanym swoim nazwiskiem i imieniem oraz skopiuj go na serwer (sbs2005) do folderu Programowanie 3Ti. (5 minut)

Po lekcji powinieneś umieć:

- scharakteryzować poznane komponenty,
- wykorzystać poznane komponenty w programach komputerowych,
- samodzielnie tworzyć notatki oraz budować programy komputerowe w oparciu o nowe wiadomości.

Scenariusze zajęć

Właściwości obiektów wektorowych

1. Cele lekcji

a) Wiadomości

Uczeń:

- zna nazwy przykładowych programów graficznych,
- wie, czym różni się grafika wektorowa od rastrowej,
- zna synonimy (grafika rastrowa = bitmapowa),
- wie, jakie są ograniczenia podczas automatycznej wektoryzacji,
- wie, jak wygląda wektoryzowane zdjęcie,
- wie, na czym polega różnica pomiędzy fontem a krzywą,
- zna podstawowe formaty zapisu plików wektorowych.

b) Umiejętności

Uczeń potrafi:

- zamienić wektor na bitmapę,
- zamieniać fonty w krzywe,
- wykorzystywać narzędzie trasowania do tworzenia obiektów wektorowych.

2. Metoda i forma pracy

Wykład, zajęcia praktyczne, praca grupowa.

3. Środki dydaktyczne

Pracownia komputerowa wraz z oprogramowaniem CorelDraw.

4. Czas trwania lekcji

45 minut

5. Przebieg lekcji

Wyjaśnienie uczniom pojęcia: „program do grafiki wektorowej”, „grafika wektorowa”, „grafika rastrowa”.

Programy graficzne generalnie możemy podzielić na:

a) programy do grafiki wektorowej (CorelDraw, Inkscape),

b) programy do grafiki rastrowej, bitmapowej (Gimp, Photoshop),

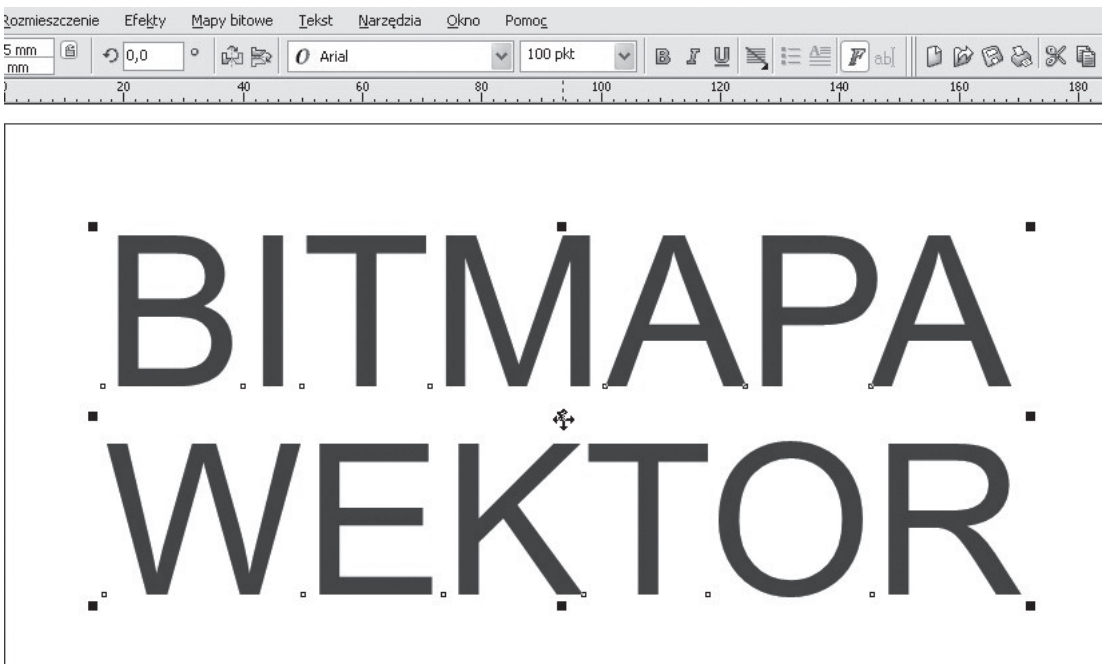
c) programy do grafiki 3D (Blender, 3DStudio MAX).

Program do grafiki wektorowej służy do pracy nad obiektami skalowalnymi, czyli takimi, których jakość nie będzie zależeć od stopnia przybliżenia. Obiekty wektorowe to krzywe: nie składają się np. z pikseli, ale są w pewnym sensie obrazowaniem pewnych matematycznych zależności, które rządzą kształtem i proporcjami obiektu. Co to oznacza w praktyce? Obiektem

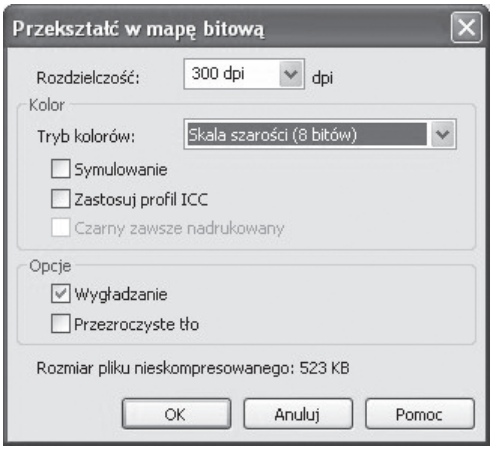
wektorowym może być np. rysunek stworzony w programie Inkscape lub CorelDraw, może to być również mapa bitowa (zdjęcie), ale poddana procesowi trasowania (wektoryzacji – będzie o tym poniżej). Ważne jest, że rysunek musi być stworzony właśnie w programie wektorowym, jeśli stworzymy go np. w Gimpie (bo jest taka możliwość) będzie rysunkiem, ale zrastrowanym (w formie bitmapy) i fizycznie nie będzie się różnił od zdjęcia.

To, co potocznie nazywamy zdjęciem, w kategoriach grafiki jest określone mianem grafiki rastrowej lub bitmapowej. Taki obraz składa się z pikseli, które mają stałą wielkość na danym monitorze, więc po powiększeniu są interpolowane (rozciągane) do większych „kwadratów". Najlepszym sposobem na zrozumienie tych zjawisk będzie zaobserwowanie ich w programie wektorowym, np. w CorelDraw. Można sobie zadać pytanie: dlaczego do porównywania używać akurat programu do grafiki wektorowej, a nie rastrowej? Odpowiedź jest taka, że nie ma w tym momencie programu przeznaczonego do obróbki zdjęć, który jednocześnie zachowywałby się, jak program wektorowy, dlatego łatwiej jest to pokazać wykorzystując program wektorowy, który ma możliwość zamiany obiektu na bitmapę lub zaimportowanie bitmapy.

Po otwarciu programu możemy wpisać 2 słowa fontem Arial 100p, powinno to wyglądać tak:



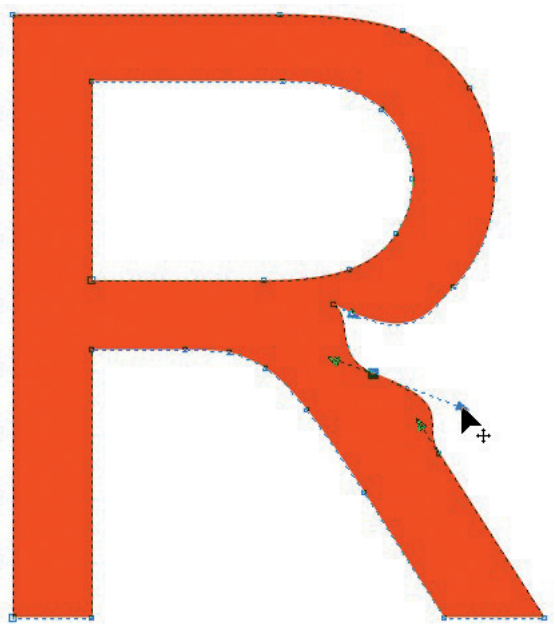
Teraz zaznaczamy słowo „BITMAPA" i otwieramy okno dialogowe: Mapy bitowe > Przekształć w mapę bitową i wybieramy ustawienia, np. takie jak poniżej (te elementy zostaną omówione przy okazji zajęć „Właściwości obrazów rastrowych").



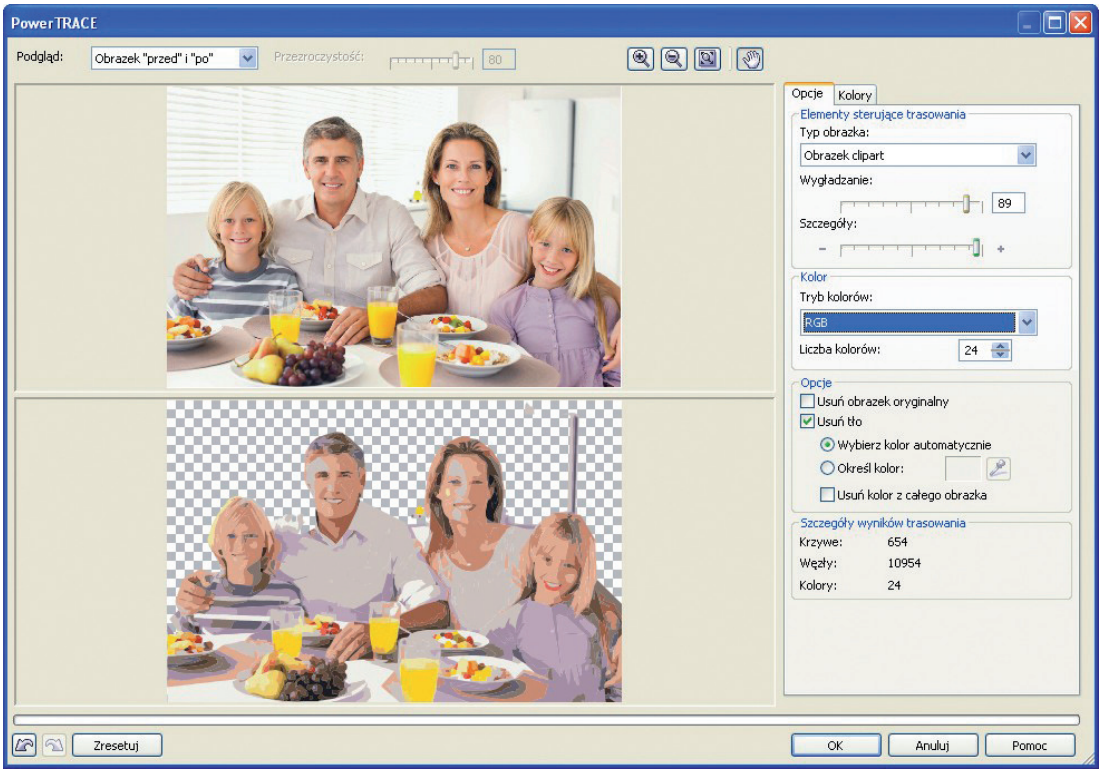
Dajemy „OK”. Na pierwszy rzut oka oba teksty wyglądają podobnie, sytuacja się zmienia przy powiększeniu. Dla lepszej widoczności można nadać kolor czerwony dla obiektu wektorowego.

Na górze bitmapa, pod spodem wektor. Oprócz wyraźnej pikselizacji, której pozbawiony jest obiekt wektorowy, widać jeszcze artefakty (np. pionowa linia) spowodowane niedokładnością wyświetlania ekranu (konflikt „optyczny” pomiędzy rozdzielczością obiektu i ekranu).

Jak widać obiekt wektorowy składa się z gładkich linii krzywych, choć w tym konkretnym wypadku (fontu) warto jeszcze uściślić pojęcie „zamiany na krzywe”.
Każdy edytowalny krój pisma w programie wektorowych zachowuje się jak „krzywa”, ale pozostaje nadal fontem, więc w przypadku nietypowego kroju może dojść do sytuacji, że użytkownik nie posiadający go w systemie, nie będzie mógł go otworzyć na swoim komputerze. W celu uniknięcia tego zjawiska, można zamienić font na rysunek. Robi się to za pomocą opcji: Rozmieszczenie > Przekształć w krzywe. Wygląd się nie zmienia, dochodzą natomiast 2 ważne cechy:
a) font przestaje być edytowalny, a więc jest „nieczuły” na ewentualny brak pliku fontu w czyimś systemie,
b) posiada teraz dodatkowe właściwości – można go modyfikować jak obrazek.

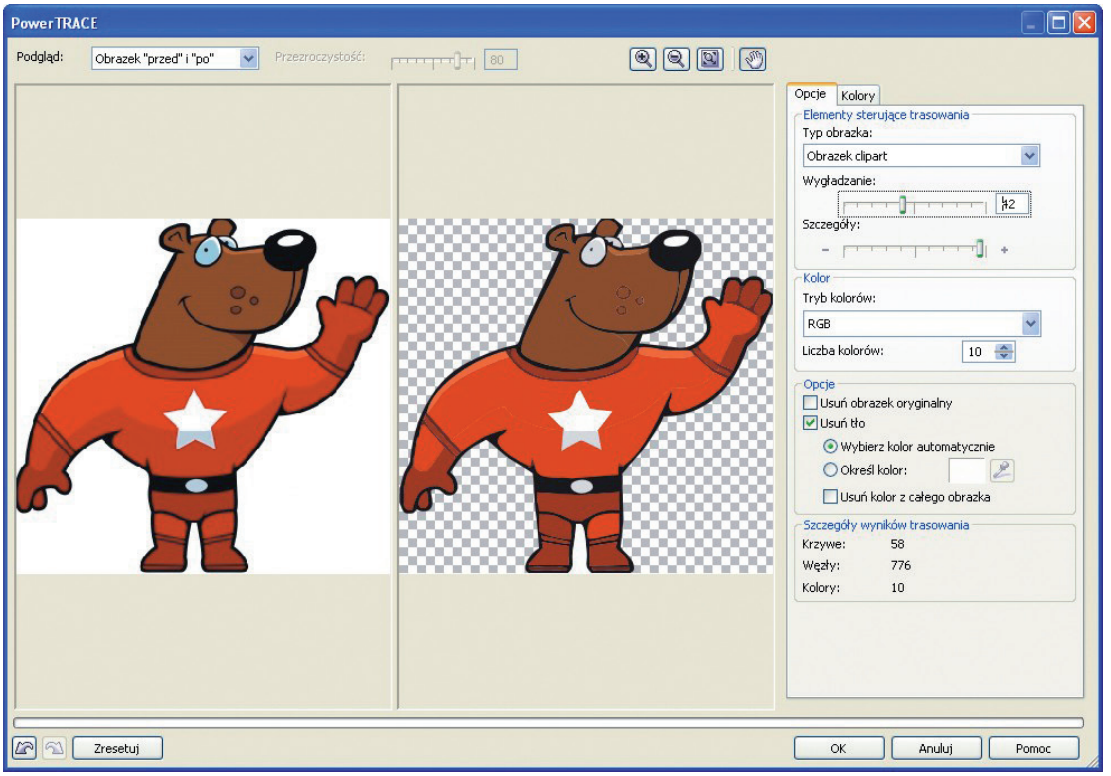


Ciekawą opcją jest trasowanie mapy bitowej, czyli zamiana zdjęcia na rysunek. Wybieramy menu Mapy bitowe > trasuj mapę bitową > obrazek clipart. Możemy uzyskać np. taki efekt:



Od razu widać, że pomimo prawie maksymalnych ustawień dokładności, wynik różni się znacznie. Wektoryzacja ma duże problemy z interpretowaniem przejść tonalnych, a to właśnie przejścia tonalne (gradienty kolorów) są istotnym składnikiem zdjęć. Zostają one zamienione na plamy jednolitego koloru, gdybyśmy chcieli utworzyć obraz z gradientami kolorów, musimy te gradienty sami stworzyć w CorelDraw, w innym wypadku będziemy mieć do czynienia z symulacją przejścia tonalnego.

Znacznie lepiej wygląda trasowanie obiektu, który nie ma skomplikowanych tonacji i nie posiada dużej ilości szczegółów.



Przykład powyżej pokazuje jaki typ grafiki poddaje się łatwej wektoryzacji. Oczywiście mówimy cały czas o wektoryzacji automatycznej, istnieją bardziej szczegółowe i skomplikowane procedury, ale wymagają znacznego doświadczenia zawodowego.

Plik po trasowaniu staje się zbiorem niezależnych obiektów, którymi możemy dowolnie manipulować.

Warto wypróbować różne ustawienia z okna dialogowego Power TRACE, aby uzyskać ciekawe efekty. Trasowanie przyda nam się również do nauki rysunku, ponieważ pozwoli nam łatwiej przyjrzeć się proporcjom lub wykorzystać je jako elementy wyjściowe do naszych prac, oczywiście w granicach wyznaczonych przez prawa autorskie.

Na koniec warto jeszcze wspomnieć o formatach plików związanych z grafiką wektorową. Jeśli zapiszemy plik w postaci źródłowej, np. w CorelDraw, otrzyma on rozszerzenie „.cdr", znacznie częściej używa się jednak plików źródłowych w formie plików Adobe Illustratora, a więc „.ai". Rozwiązaniem, które uniezależni nas od oprogramowania, może być zapisanie pliku, czy w kontekście CorelDraw – wyeksportowanie pliku do formatu EPS, SVG lub PDF. Pliki PDF są niezależne od systemu operacyjnego i oprogramowania, ale tylko kiedy mamy na myśli ich czytanie, przeglądanie, wyświetlanie. Należy jednak pamiętać, że możemy mieć problem z zaimportowaniem pliku PDF stworzonego w Illustratorze, np. do Corela lub pliku wygenerowanego dzięki PDF Creator do Illustratora itd.

Właściwości obiektów bitmapowych

1. Cele lekcji

a) Wiadomości

Uczeń:

- dowiaduje się o programie GIMP,
- dowiaduje się o tzw. „wolnej licencji”,
- wie, jakie są podstawowe parametry obrazu rastrowego,
- zna pojęcie ppi, dpi,
- zna pojęcie interpolacji,
- wie, jak wygląda obraz po wydruku w jakości 72 dpi,
- wie, jak wygląda obraz po nadmiernym obniżeniu jakości JPG.

b) Umiejętności

Uczeń potrafi:

- zmienić wielkość i „wagę” plików,
- zapisywać w formatach bezstratnych,
- rozpoznawać różne rodzaje interpolacji.

2. Metoda i forma pracy

Wykład, zajęcia praktyczne, praca grupowa.

3. Środki dydaktyczne

Pracownia komputerowa wraz z oprogramowaniem Gimp.

4. Czas trwania lekcji

45 minut

5. Przebieg lekcji

Na początek wprowadzenie dotyczące samego programu Gimp.

Gimp to darmowy program do grafiki rastrowej, czyli do obróbki zdjęć. Jest to program dostępny na tzw. otwartej licencji (Open Source). Jest programem oferującym podobne funkcje, jak jego komercyjny odpowiednik Adobe Photoshop.

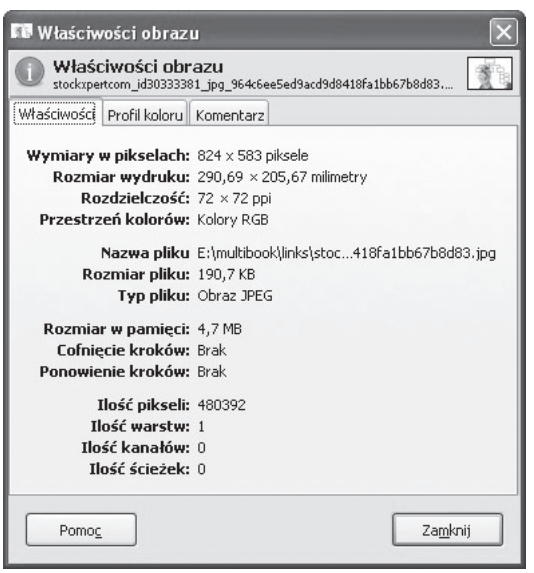
Na stronie gnu.org można znaleźć dokładne wyjaśnienie pojęcia „wolna licencja”, możemy na niej przeczytać m.in.

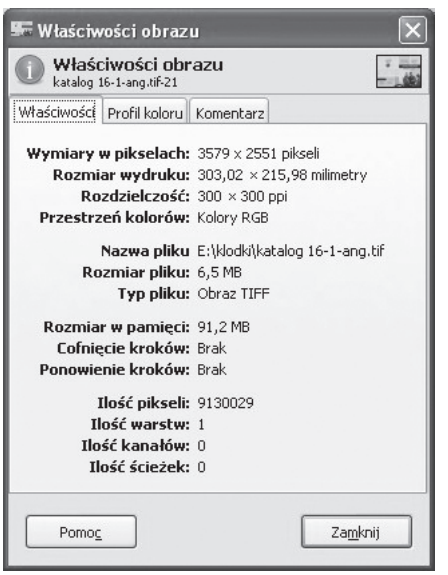
„Co to znaczy „GPL”?

„GPL” oznacza „General Public License”, „Powszechna Licencja Publiczna”. Najbardziej rozpowszechnioną licencją tego rodzaju jest Powszechna Licencja Publiczna GNU lub krótko: GNU GPL. Można również używac samego skrotu „GPL”.

Dlaczego GPL pozwala użytkownikom na publikację własnych zmodyfikowanych wersji?
Decydującym aspektem wolnych programów jest swoboda współpracy użytkowników. Pozwolenie użytkownikom, którzy chcą pomagać innym, na dzielenie się własnymi poprawkami błędów i udoskonaleniami, jest absolutnie kluczowe.
Niektórzy proponowali alternatywy dla GPL, które wymagały, aby zmienione wersje przechodziły przez pierwotnego autora. W praktyce, póki dbałby on o potrzeby konserwacji programu, mogłoby to działać dobrze. Jednak jeśli autor przestaje, mniej czy bardziej, zajmować się konserwacją, by robić coś innego, lub gdy nie zwraca uwagi na wszystkie potrzeby użytkowników, ten plan zawodzi. Pomijając problemy praktyczne, taka metoda nie pozwala użytkownikom na wzajemną pomoc.
Czasem proponuje się kontrolę nad zmodyfikowanymi wersjami jako środek zapobiegający kłopotliwemu pomieszaniu między rozmaitymi wersjami utworzonymi przez użytkowników. Wedle naszego doświadczenia, to pomieszanie nie stanowi większego kłopotu. Wiele wersji Emacsa wykonano poza Projektem GNU, ale użytkownicy potrafią je odróżnić. GPL wymaga umieszczania w wersji nazwiska producenta, by odróżnić ją od innych wersji i chronić reputację pozostałych opiekunów."
Z pewnością wielokrotnie spotkacie się z pojęciem „wolnej licencji", „open source", „GNU", choćby w wypadku Mozilla Firefox czy Wordpress.

Praca nad grafiką bitmapową powinna się zacząć od zrozumienia właściwości i parametrów obiektu na którym pracujemy.
Po otwarciu zdjęcia w Gimpie możemy wybrać opcję Obraz > Właściwości obrazu, w zależności od zdjęcia, które otwieramy pokażą się różne wartości jego parametrów.
Poniżej znajdują się 2 różne zrzuty ekranu przedstawiające różne wartości parametrów





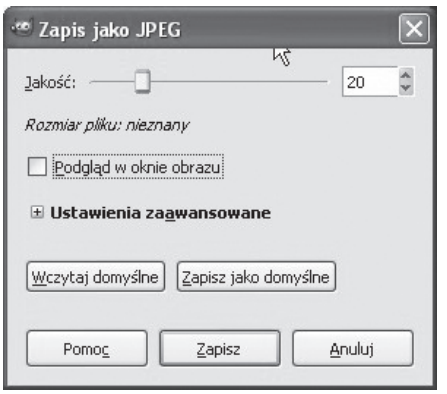
zdjęć. Pierwszy obraz nadaje się głównie do zastosowań Internetowych, drugi do zastosowań drukarskich (pomijając fakt, że przestrzeń barwna powinna być CMYK, a nie RGB, niestety na dzień dzisiejszy tylko dodatkowe plug-iny umożliwiają zmianę przestrzeni barwnej na CMYK w Gimpie).

Pierwszy obraz posiada wymiary w pikselach 824x583, co daje 480392 pikseli, drugi posiada wymiary ponad 3000x2000 i łącznie zawiera 9 130 029 pikseli, czyli więcej niż np. lustrzanka Nikon D60. Błędem byłoby jednak nazywanie tego rozdzielczością. Tu chodzi o wielkość obrazu, nie rozdzielczość. Za rozdzielczość odpowiada inny parametr – ppi. W pierwszym przypadku mamy 72, w drugim 300. Tak naprawdę ppi = dpi. W poligrafii nie mówimy pixels per inch (piksele na cal), lecz dots per inch (punkty na cal), ale w praktyce to wartość analogiczna.

Zbliżamy się do ciekawego punktu zagadnienia – rozmiar wydruku. Proszę zauważyć, że te obrazy posiadają podobne wartości w parametrze „Rozmiar wydruku", ale tylko ten drugi wydrukuje się „ostro", pierwszy ma zbyt małą wartość ppi, co w efekcie doprowadzi do pikselizacji.

To, co wynika jeszcze z porównania tych dwóch okien dialogowych, to format pliku. JPG jest to rodzaj kompresji obrazu i jest na tyle dobry, że nie musimy się obawiać nadmiernego spadku jakości. W większości przypadków spadek jakości będzie niezauważalny dla oka, ale może dojść do sytuacji, gdy będziemy musieli zredukować wielkość pliku (np. wymogi mailingu grupowego) bez redukcji jego wielkości. Pozostaje wtedy zapisanie pliku z pomniejszoną wartością „Jakość".

Kiedy wejdziemy do okna zapisu, możemy wybrać format JPG, po kliknięciu na OK pojawi się okno dialogowe, w którym możemy ustawić jakość, obserwując jednocześnie spadek wielkości (ilości kilobajtów) pliku. Możemy wybrać, np. mała wartość 20, aby zaobserwować wyraźny spadek „ciężaru" pliku, ale również wyraźne pogorszenie jakości.



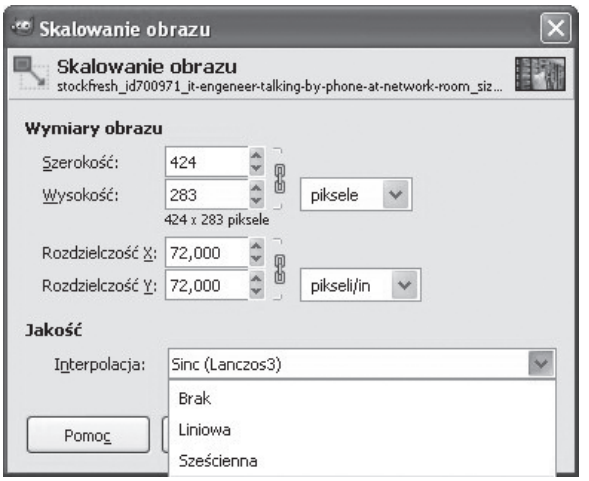
Poniżej efekt – warto porównać pierwotne zdjęcie i po zapisaniu w JPG z ustawieniami 20: jakość uległa wyraźnemu pogorszeniu, ale rozmiar pliku zmniejszył się aż 10 krotnie!

Jeżeli pracujemy nad jakimś projektem i zależy nam na zapisie bezstratnym, możemy użyć formatu tiff, z ustawieniami jak poniżej (format tiff wybieramy z okna dialogowego „Zapisz jako").
Całkiem dobrym i poprawnym w poligrafii rozwiązaniem będzie opcja wyboru LZW, ponieważ zapewnia ona wysoką jakość i dosyć spory ubytek „wagi" pliku.

Na koniec warto omówić jeszcze jedno pojęcie – interpolacja. Jest to rodzaj „dobudowywania" pikseli, czyli gdy skalujemy obraz, który ma wielkość np. 100x100 pikseli do obrazu 400x400 pikseli, powstanie ogromna ilość pikseli, które muszą zostać utworzone.

Gimp rozróżnia 3 podstawowe sposoby interpolacji:
a) liniowa: piksel tworzony jest w oparciu o 4 piksele: góra, dół, lewy, prawy,
b) sześcienna: dokładniejsza, oblicza wartość brakującego piksela w oparciu o 8 sąsiadujących,
c) sinc (nieobecny w wersjach poniżej 2.4) - uwzględnia średnie wartości z kwadratów 4x4, 6x6 lub 8x8, co daje rezultaty podobne do tych otrzymywanych w Photoshopie.

Warto na pewno samemu poeksperymentować zarówno z ustawieniami kompresji (Jakości), jak i z ustawieniami różnych sposobów interpolacji obrazu. Wtedy będzie można wybrać najbardziej optymalny sposób na zachowanie dużej jakości i niewielkich rozmiarów pliku.


Skalowanie obrazu
Skalowanie obrazu
stockfresh_id700971_it-engeneer-talking-by-phone-at-network-room_siz...
Wymiary obrazu
Szerokość:
424
Wysokość:
283
piksele
424 x 283 piksele
Rozdzielczość X:
72,000
Rozdzielczość Y:
72,000
pikseli/in
Jakość
Interpolacja:
Sinc (Lanczos3)
Brak
Liniowa
Sześcienna
Pomoc

Techniki rozmieszczania i wyrównywania obiektów wektorowych (CorelDraw)

1. Cele lekcji

a) Wiadomości

Uczeń:

- dowiaduje się o standardach w zarządzaniu rozmieszczeniem obiektów,
- wie, jakie są sposoby wyrównywania i rozkładu obiektów,
- zna pojęcie „prowadnicy”,
- zna przydatny skrót klawiaturowy „P”,
- zna sposoby na przesuwanie obiektów za pomocą kombinacji klawiszy.

b) Umiejętności

Uczeń potrafi:

- rozmieścić równomiernie na stronie obiekty,
- wyrównać obiekty do środka strony,
- wyrównać obiekty względem siebie,
- ustawić precyzyjnie skoki przesunięcia obiektów,
- pracować z prowadnicami.

2. Metoda i forma pracy

Wykład, zajęcia praktyczne, praca grupowa.

3. Środki dydaktyczne

Pracownia komputerowa wraz z oprogramowaniem CorelDraw.

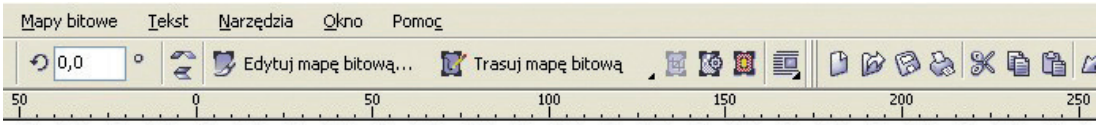
4. Czas trwania lekcji

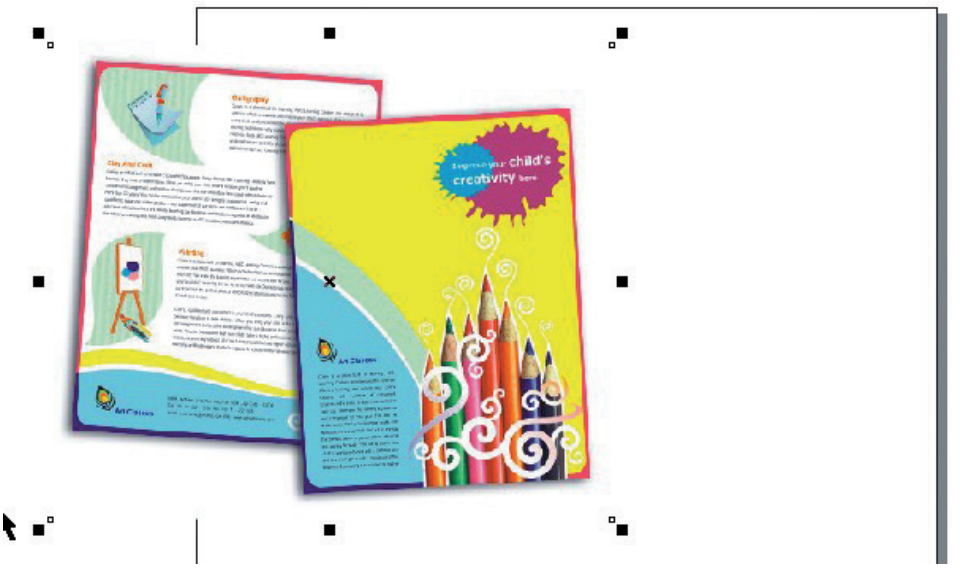
45 minut

5. Przebieg lekcji

Wprowadzenie na temat uniwersalności zasad rozmieszczania obiektów.

Programy do grafiki wektorowej i grafiki 3D pozwalają operować dużą ilością elementów na obszarze roboczym. W celu uporządkowania zarządzania obiektami wprowadza się różne techniki, głównie polegające na automatycznym wyrównywaniu obiektów względem siebie, czy równomiernym rozmieszczaniu ich na obszarze roboczym. Z tego powodu takie pojęcia jak „align” (wyrównanie) czy „pivot point” (punkt obrotu) będą często spotykanymi komendami.

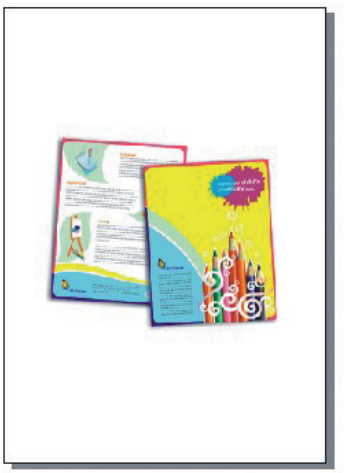




Dla ułatwienia czynności stosuje się też skróty klawiaturowe, które w częstej pracy okazują się bardzo przydatne. Osoby używające np. CorelDraw prędzej czy później docenią wartość litery „P" na klawiaturze.

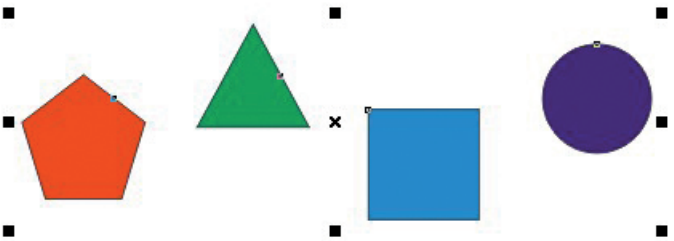
Zaimportujmy np. jakiś obiekt do Corela (Plik > importuj...), możemy oczywiście sami stworzyć dowolny obiekt – ważne, żeby jego położenie było dowolne, w jakimkolwiek miejscu na obszarze roboczym.

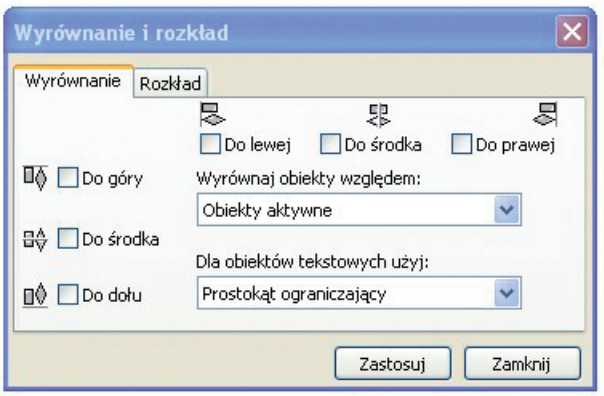
Zaznaczmy obiekt i wciśnijmy klawisz „P". Obiekt ustawi się na środku arkusza roboczego, zaoszczędzając nam czasu na tworzenie prowadnic, czy obliczanie ręczne środka arkusza.



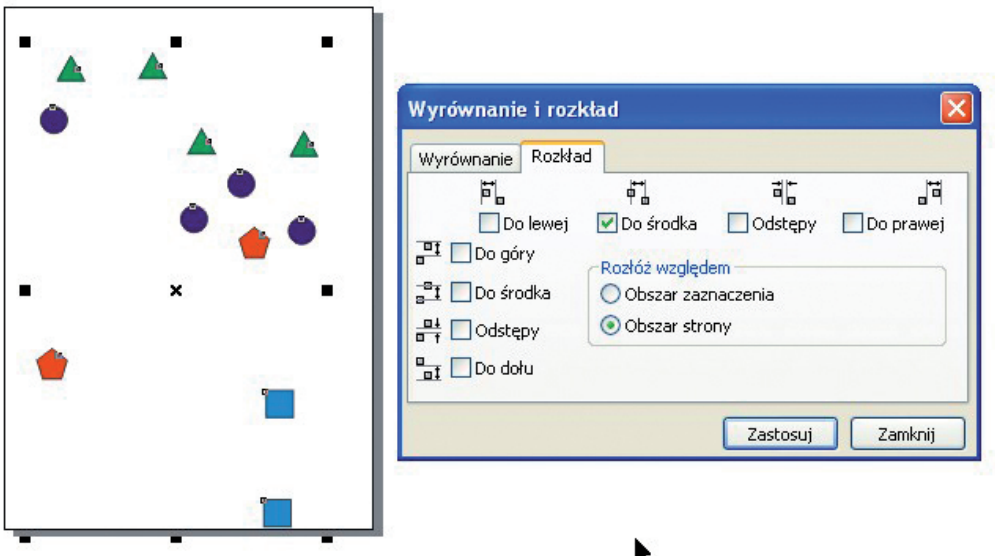
(prowadnica to linia pomocnicza, wyprowadza się ją - nie tylko w CorelDraw - przeciągając myszką górną lub boczną linijkę z wciśniętym lewym klawiszem myszy).

Podstawowym sposobem na wyrównywanie w pionie i poziomie jest polecenie „Wyrównanie i rozkład" (menu Rozmieszczenie > Wyrównanie i rozkład > Wyrównanie i rozkład... – na samym dole)
Ta funkcja ma sens tylko w wypadku więcej niż jednego obiektu. Stwórzmy więc kilka obiektów, zaznaczmy je, a następnie wejdźmy do okna dialogowego, kierując się ścieżką określoną powyżej.

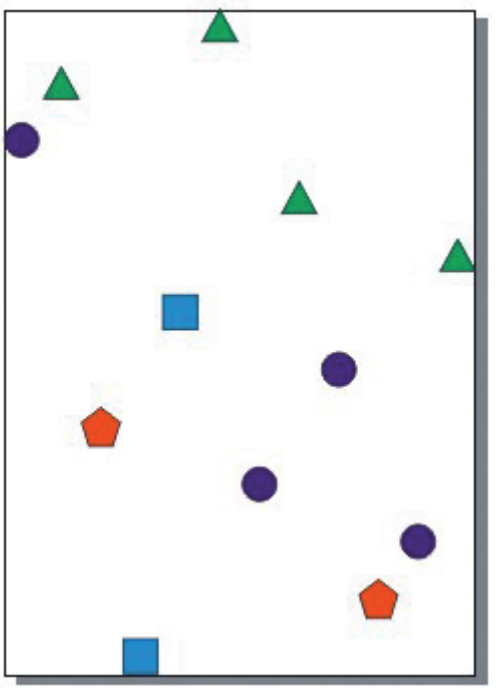




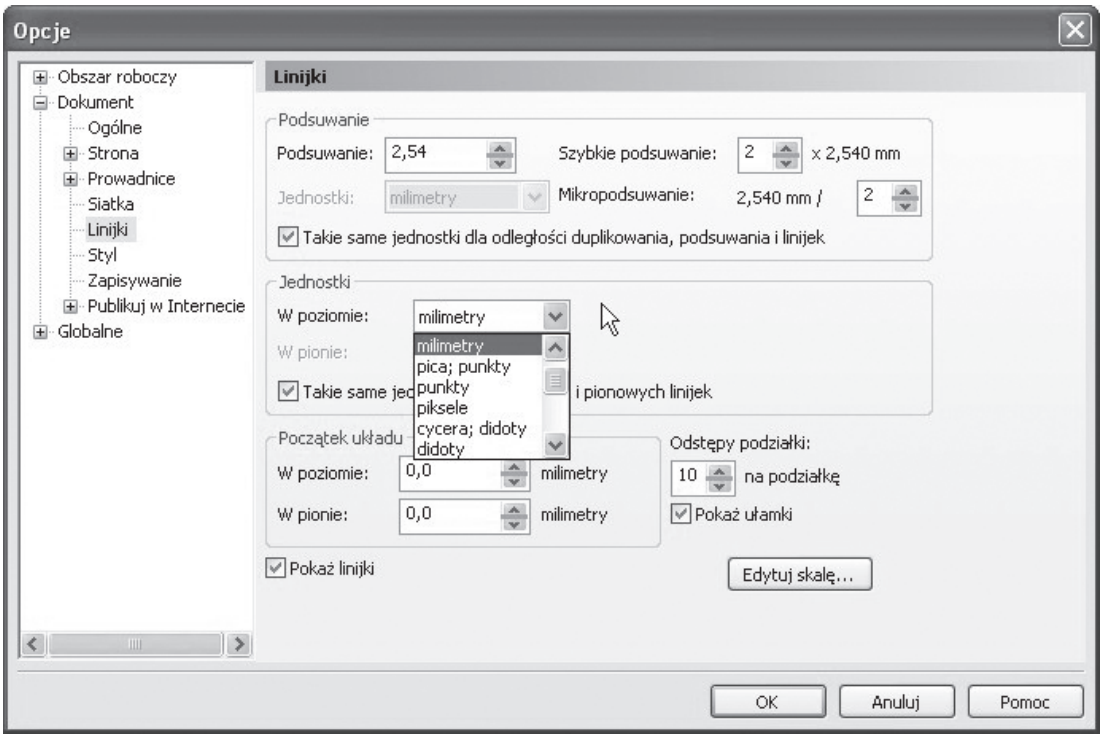
Okno dialogowe wyrównania jest tak pomyślane, że mamy opisane i przedstawione graficznie sposoby wyrównania. Warto poeksperymentować, ponieważ najlepiej można będzie zrozumieć te operacje, dostając konkretne wyniki wyrównania.
Ciekawych rozwiązań dostarcza nam druga zakładka w tym oknie dialogowym, czyli „Rozkład". Rozmieśćmy chaotycznie kilka figur, zaznaczmy je i wejdźmy do zakładki „Rozkład". W tej zakładce zaznaczmy opcje tak, jak na obrazku poniżej.



Kliknijmy „zastosuj”. Figury powinny się wyrównać w obszarze całej strony.

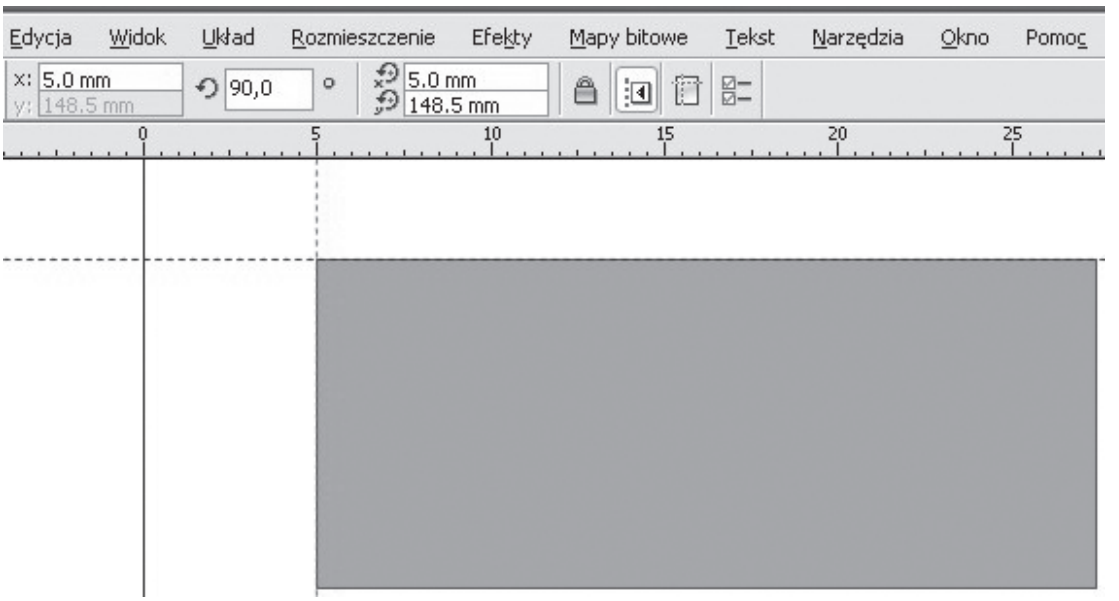


Pomocne przy ustawianiu obiektów będą również strzałki na klawiaturze. Możemy zaznaczyć obiekt i przesuwać go skokowo za pomocą strzałek. Możemy również stosować kombinację strzałka + Shift. Jeżeli chcemy mieć pełną kontrolę nad skokami przesunięć, możemy to ustawić w menu Narzędzia > Opcje > Dokument > Linijki >Podsuwanie i tu możemy ustawić wielkość skoku. Jest to rozbudowane, ale potrzebne okno dialogowe.



Warto zauważyć, że skoki przesunięć za pomocą kombinacji klawiszy strzałka + Shift, strzałka + Ctrl stosuje się jako standard w wielu programach graficznych.
Wspomnieliśmy powyżej o tzw. „prowadnicy". Kiedy wyciągamy z linijki prowadnicę możemy dokładnie ustawić jej położenie. Do tego służą pola w górnej części interfejsu.

Wejdźmy jednak najpierw do menu Widok i zaznaczmy „przyciągaj do prowadnic ...". To oznacza, że włączy nam się opcja „magnesu", który będzie przyciągał obiekty do prowadnicy. Następnie wyprowadźmy 2 prowadnice, jedną z góry, drugą z lewej strony i wpiszmy w polach, jak na poniższym obrazku wartości w minimetrach. Następnie stwórzmy prostokąt i zaobserwujemy przy zbliżaniu go do prowadnic efekt „przyciągania". Całość powinna wyglądać tak:



Takie układy prowadnic stosuje się często przy tworzeniu arkusza z wizytówkami. Teraz kiedy już wiemy jak stosować prowadnice, nie powinno nam sprawdzić problemu zastosowanie ich do bardziej złożonych czynności.

Automatyzacja powtarzalnych czynności w MS Word - makra

1. Cele lekcji

a) Wiadomości

Uczeń:

- dowiaduje się czym jest „makro”,
- rozumie potrzebę pracy zautomatyzowanej,
- wie, jak tworzyć skróty klawiaturowe do makr,
- dowiaduje się o panelu Visual Basic w MS Word.

b) Umiejętności

Uczeń potrafi:

- utworzyć makro,
- zarządzać makrami.

2. Metoda i forma pracy

Wykład, zajęcia praktyczne, praca grupowa.

3. Środki dydaktyczne

Pracownia komputerowa wraz z oprogramowaniem MS Office 2007.

4. Czas trwania lekcji

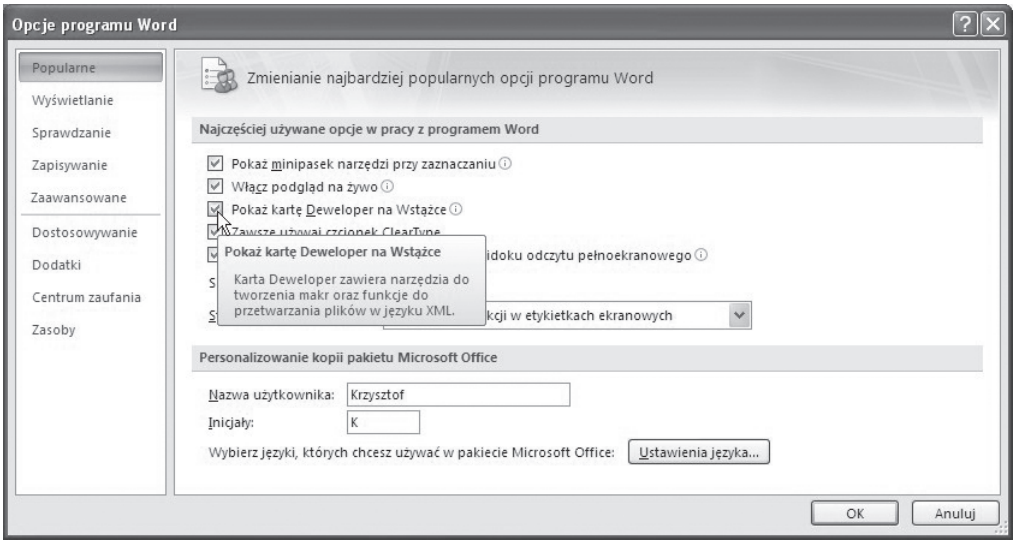
45 minut

5. Przebieg lekcji

Wyjaśnienie pojęcia „makra”. Pod nazwą „makro” rozumiemy funkcję polegającą na stworzeniu zapisu pewnych czynności w celu późniejszego ich odtworzenia. Zdarza się wielokrotnie, że pracując np. w programie biurowym MS Word, będziemy zmuszeni do wykonywania raz zarazem, tej samej lub bardzo podobnej czynności. W celu ułatwienia i przyspieszenia pracy możemy się posłużyć funkcją makr.

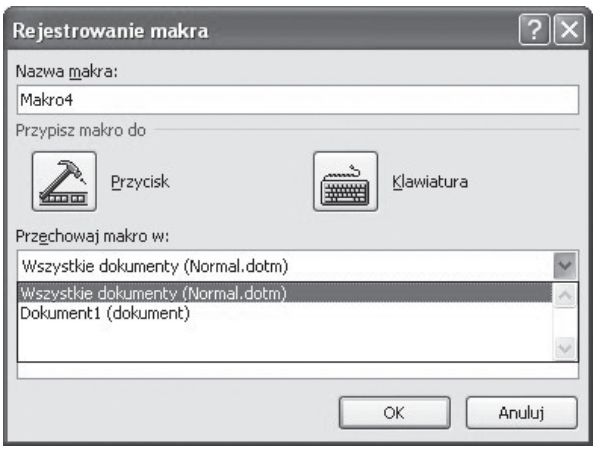
Makra zatem to rodzaj programowania, ale nie wymagają wiedzy programistycznej, ponieważ możemy je tworzyć automatycznie. Większość makr w programach MS Office jest napisanych w języku Microsoft Visual Basic for Applications, czyli VBA.

Praca na makrach będzie możliwa, kiedy na wstążce będziemy mieć zakładkę „Deweloper”. Możemy wejść również w Widok > makra, ale warto jest poznać miejsce, w którym dodajemy tą nową zakładkę. Domyślnie ta zakładka jest wyłączona. W celu jej dodania wchodzimy do „Opcji programu Word” i zaznaczamy „Pokaż kartę Deweloper na wstążce”. Po kliknięciu na OK karta doda się do wstążki.



Udało nam się wstawić zakładkę Deweloper. Znajdziemy w niej sporo zaawansowanych funkcji, co w pewnym sensie tłumaczy dlaczego nie jest ona widoczna w domyślnych ustawieniach systemu.

Działanie makr najlepiej jest prześledzić na przykładzie. Wchodzimy do zakładki i klikamy na guzik Zarejestruj makro. Powinno się pokazać takie okno:



To okno dialogowe ma kilka ciekawych elementów:

a) nazwa makra – będzie to po prostu nazwa pod jaką się pokaże na liście makr,

b) przypisz makro do Przycisku lub Klawiatury – mamy tu możliwość dodania przycisku do

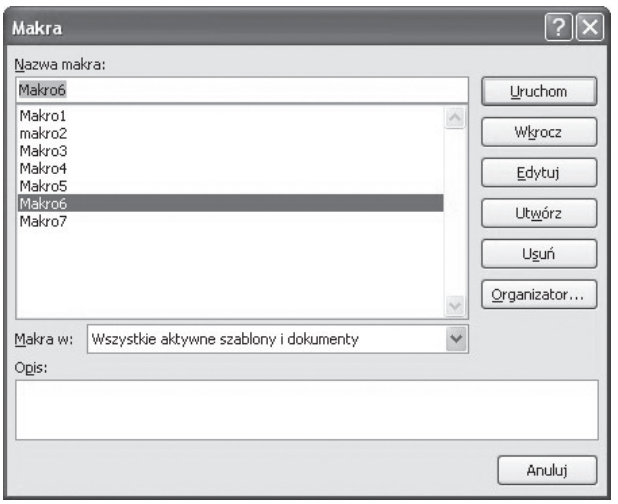
paska szybkiego dostępu lub utworzenie skrótu klawiaturowego,
c) przechowaj makro we wszystkich dokumentach lub bieżącym dokumencie. Jest to przydatna opcja, ponieważ raz stworzony zapis makra możemy wykorzystać przy innej okazji.

Możemy przystąpić do stworzenia prostego makra. Kliknijmy na „Zarejestruj makro" i dajmy OK.
Z pozoru nic się nie zmienia, ale przy kursorze pojawia się ikonka „taśmy magnetofonowej", która jest symbolem procesu nagrywania. Możemy teraz zrobić sobie, np. tabelkę, sformatować ją.

kolumna 1	kolumna2
	2
	2

Podczas pracy mamy 2 możliwości – wstrzymać i zatrzymać rejestrowanie. Wstrzymanie „pauzuje" nam pracę i możemy do niej wrócić, zatrzymanie oznacza zakończenie etapu rejestracji makra.

Kiedy zatrzymamy proces rejestrowania makra, możemy kliknąć poniżej, a następnie na ikonkę Makra. Pojawi się okno dialogowe z wyborem makr – będzie tam również nasze.
Wybieramy je i wciskamy guzik „Uruchom". To spowoduje „odtworzenie" procesu „nagrywania", a w konsekwencji zobaczymy powielony wynik naszych prac. Praca z makrami może być przydatna i z pewnością warto się zapoznać z tym narzędziem.



Po stworzeniu makra możemy zobaczyć jego kod źródłowy w VBA. W tym celu wchodzimy w Deweloper > Visual Basic > Tools > Macros... , wybieramy swoje makro i klikamy na Edit. Przykładowe makro, które pozwoliło utworzyć tą tabelkę powyżej, wygląda tak:

```
Sub Makro6()
'
', Makro6 Makro
'
'
    ActiveDocument.Tables.Add Range:=Selection.Range, NumRows:=3, NumColumns:= _
        6, DefaultTableBehavior:=wdWord9TableBehavior, AutoFitBehavior:= _
        wdAutoFitFixed
    With Selection.Tables(1)
        If .Style <> „Tabela - Siatka” Then
            .Style = „Tabela - Siatka”
        End If
        .ApplyStyleHeadingRows = True
        .ApplyStyleLastRow = False
        .ApplyStyleFirstColumn = True
        .ApplyStyleLastColumn = False
        .ApplyStyleRowBands = True
        .ApplyStyleColumnBands = False
    End With
    Selection.Tables(1).Style = „Średnie cieniowanie 2 — akcent 3”
    Selection.Tables(1).Style = „Średnie cieniowanie 2 — akcent 4”
    Selection.TypeText Text:=”kolumna 1”
    Selection.MoveRight Unit:=wdCell
    Selection.TypeText Text:=”kolumna2”
    Selection.MoveDown Unit:=wdLine, Count:=1
    Selection.TypeText Text:=”2”
    Selection.MoveDown Unit:=wdLine, Count:=1
    Selection.TypeText Text:=”2”
    Selection.MoveDown Unit:=wdLine, Count:=1
End Sub
```

Kod nie jest trudny i może warto go wykorzystać do ręcznej modyfikacji makra, będzie to na pewno cenna umiejętność.

Korespondencja seryjna krok po kroku (MS Word 2007)

1. Cele lekcji

a) Wiadomości

Uczeń:

- dowiaduje się czym jest „korespondencja seryjna”,
- rozumie korespondencję seryjną w szerszym kontekście – automatyzacji pracy,
- zna pojęcie „pola bazy danych”,
- wie, do czego służą pola w Wordzie,
- wie, że w zakresie korespondencji seryjnej Word może współpracować z programem Excel.

b) Umiejętności

Uczeń potrafi:

- korzystać z kreatora korespondencji seryjnej,
- przygotować w MS Word prostą bazę adresową,
- wykorzystać właściwości pól w MS Word,
- powiązać pliki Excela z procesem korespondencji seryjnej.

2. Metoda i forma pracy

Wykład, zajęcia praktyczne, praca grupowa.

3. Środki dydaktyczne

Pracownia komputerowa wraz z oprogramowaniem MS Office 2007.

4. Czas trwania lekcji

45 minut

5. Przebieg lekcji

Korespondencja seryjna jest to jedna z trudniejszych czynności do wykonania w programie Word, dlatego proszę się nie przejmować, jeśli coś nie wyjdzie za pierwszym razem.

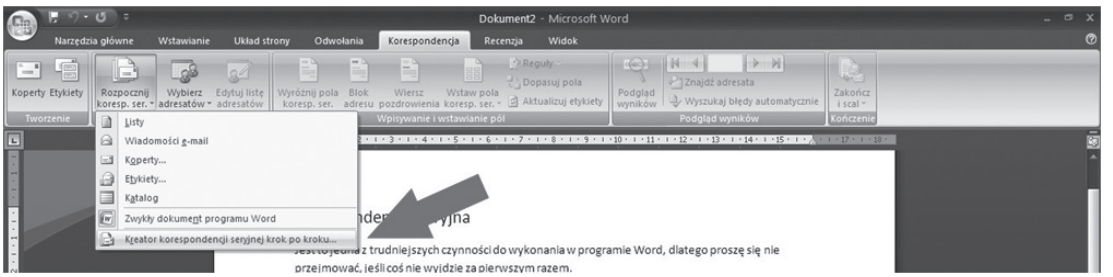
Co to jest korespondencja seryjna?

Najlepiej będzie wytłumaczyć to na przykładzie: mamy np. do dyspozycji przygotowaną listę nazwisk, adresów i musimy wysłać 500 listów. Byłoby stratą czasu adresowanie takiej ilości listów i kopert, prawdopodobnie w wielu sytuacjach byłoby to nawet nierealne z braku czasu, dlatego stosuje się różnego rodzaju procedury automatyzujące taką pracę.

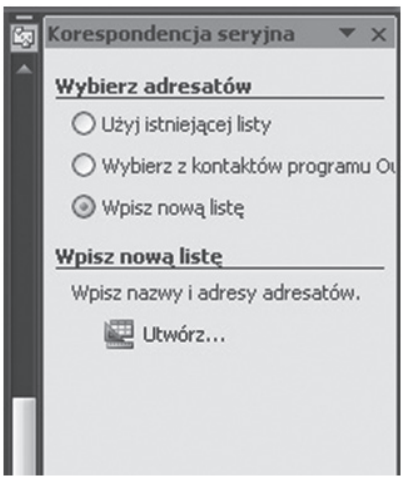
Pamiętajmy, że do zrobienia korespondencji potrzebujemy listy w formie, która jest obsługiwana przez procedury korespondencji w Wordzie 2007.

Teraz zajmiemy się wyjaśnieniem krok po kroku o co w tym wszystkim chodzi.
Przygotujemy korespondencję opierając się na bazie danych, którą sobie sami przygotujemy.

1. Otwieramy pusty dokument Worda.
2. Zapisujemy go np. jako „korespondencja listowa”.
3. Wchodzimy w menu „Korespondencja”.



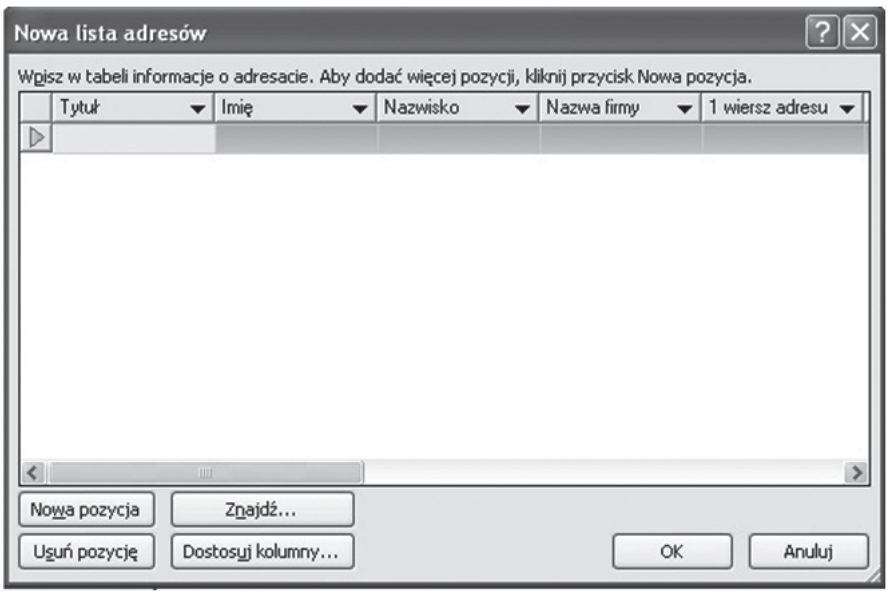
4. Klikamy na „Rozpocznij korespondencę seryjną”.
5. Klikamy na „Kreator korespondencji seryjnej krok po kroku”.
6. Po prawej stronie otworzy się dodatkowe okno „Korespondencja seryjna”.
7. Zaznaczamy „Listy”, ale prawdopodobnie będzie to już zaznaczone domyślnie.
8. Na dole tego okienka po prawej znajduje się strzałka i napis „Następny: Dokument początkowy” – klikamy na to.
9. Wybierz dokument początkowy, zaznaczamy „Użyj bieżącego dokumenty”, chodzi o to, że naszym dokumentem podstawowym będzie właśnie ta pusta kartka, którą mamy otworzoną.
10. Na dole okienka po prawej jest strzałka i napis „Następny: Wybierz adresatów” – klikamy na to.
11. Wybierz adresatów, zaznaczamy „Wpisz nową listę”.



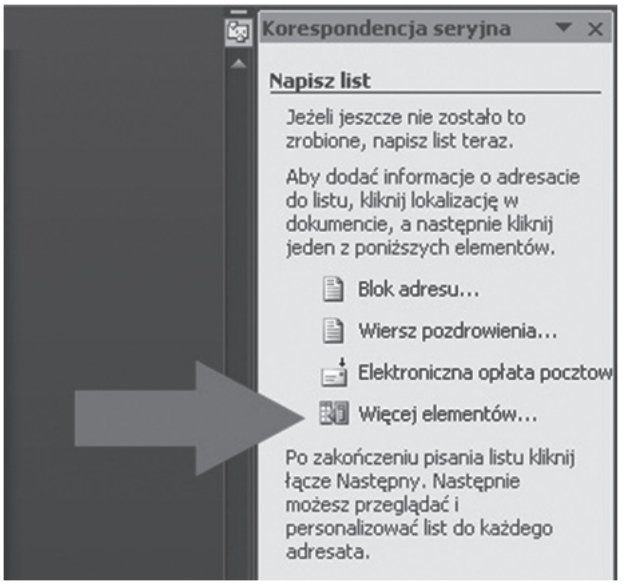
Wybieramy tą opcję, ponieważ zakładamy, że na razie nie mamy jeszcze listy adresatów; gdybyśmy taką posiadali zaznaczylibyśmy opcję „Użyj istniejącej listy".
Jest to możliwe w sytuacji, gdy mamy dokument przygotowany, np. w programie MS Excel. Robimy w nim zwykłą tabelkę, pamiętając, że to co będzie na samej górze, będzie jednocześnie tytułem pola tekstowego, więc ważne jest to, aby pierwsze komórki od góry zawierały słowa, które będą zrozumiałe i oczywiste dla osób pracujących nad korespondencją.

	A	B	C	D	E
1	Imię	nazwiski	ulica	kod	miasto
2	przykład1	przykład1	przykład1	przykład1	przykład1
3	przykład2	przykład2	przykład2	przykład2	przykład2

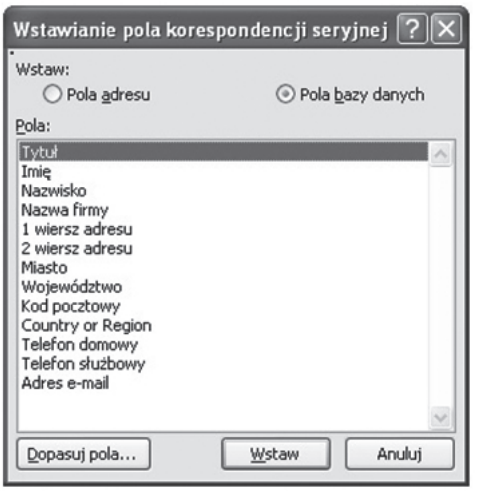
12. Klikamy na „Utwórz" i powinno się pojawić takie okienko:



13. To jest okienko, w którym możemy wpisywać dane osób: imię, nazwisko, adres itd. Gdy zapełnimy interesujące nas pozycje, dajemy na „Nowa pozycja" i możemy wpisywać dane drugiej osoby itd.
14. Kiedy kończymy tworzenie swojej listy klikamy „OK."
15. Pojawia się okno dialogowe do zapisania dokumentu. Traktujemy to tak, jakbyśmy zapisywali dokument Worda. Wybieramy miejsce, w którym chcemy plik zapisać, nadajemy nazwę np. „baza danych", dajemy Zapisz, potem OK.
16. Na dole okna po prawej powinno pisać „Krok 3 z 6" „Następny: Napisz list", klikamy na niego.
17. Klikamy na opcję „Więcej elementów..."



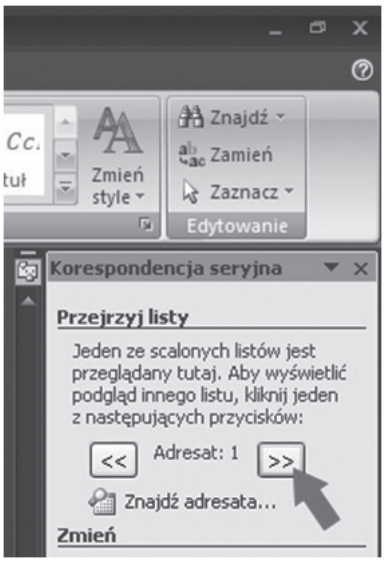
18. Pokaże się takie okno:



19. Wybieramy te elementy, które mają być w nagłówkach listów czyli, np. Tytuł, Imię, Nazwisko, Miasto, za każdym razem klikamy na „Wstaw”.
20. Powinny nam się pojawić następujące pola:

«Tytuł»«Imię»«Nazwisko»«Nazwa_firmy»«M_1_wiersz_adresu»
Możemy je dowolnie formatować, np.
«Tytuł»
«Imię»«Nazwisko»
«Nazwa_firmy»
«M_1_wiersz_adresu»

21. Na dole okna po prawej powinniśmy znaleźć „Krok 4 z 6”, „Następny: Przejrzyj listy”. W miejscach <<Tytuł>> czy <<Imię>> pojawią się te dane, które wpisaliśmy w swojej bazie danych.
22. Jeżeli chcemy przeglądać wszystkich po kolei, musimy jeszcze kliknąć na strzałki w oknie „Korespondencja seryjna”.



Jeżeli jesteśmy pewni, że wszystko jest w porządku, klikamy na „Ukończ scalanie”.
Tak przygotowany dokument gwarantuje, że na każdej stronie wydrukuje się inny adres korespondencyjny.

Możemy przygotować również krótką bazę adresów w Excelu i podpiąć ją do korespondencji seryjnej.
Cały proces tworzenia korespondencji będzie przebiegał podobnie, w punkcie 11 musimy tylko wybrać opcję „Użyj istniejącej listy” i wybrać plik Excela (nie bazy danych tylko Excela). Pozostałe kroki będzie można już przeprowadzać automatycznie.

Layout jednokolumnowy oparty na div (HTML/CSS)

1. Cele lekcji

a) Wiadomości

Uczeń:

- wie, co oznacza pojęcie „layoutu”,
- wie, na czym polega layout oparty o div,
- zna pojęcia „container”, „header”, „footer”,
- wie, jak wygląda prawidłowa składnia CSS,
- wie, jak wygląda dokument bez styli i ten sam z podpiętym arkuszem,
- zna pojęcia „float” i „clear”.

b) Umiejętności

Uczeń potrafi:

- połączyć html z css,
- stworzyć poprawnie szablon html i arkusz css,
- stworzyć menu w oparciu o atrybut ul.

2. Metoda i forma pracy

Wykład, zajęcia praktyczne, praca grupowa.

3. Środki dydaktyczne

Pracownia komputerowa.

4. Czas trwania lekcji

45 minut

5. Przebieg lekcji

Pojęcie „layoutu”. Layout to geometryczny układ treści, ma on szczególne znaczenie w wypadku projektowania stron, ponieważ od niego zależy architektura serwisu. Niezwykle ważne jest zrozumienie na jakich zasadach funkcjonuje układ oparty o bloki div.

Tworzymy prosty szkielet html

```
<!DOCTYPE html PUBLIC „-//W3C//DTD XHTML 1.0 Transitional//EN”
   „http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd”>
<html xmlns=”http://www.w3.org/1999/xhtml” lang=”pl”>
<head>
<meta http-equiv=”Content-Type” content=”text/html; charset=utf-8”>
<link rel=”Stylesheet” type=”text/css” href=”style.css” />
```

```
<title>Layout 1 kolumna</title>
</head>
<body>

<div id=”container”>
        <div id=”header”>
                <h1>
                        Nazwa firmy
                </h1>
        </div>
        <div id=”navigation”>
                <ul>
                        <li><a href=”#”>Strona główna</a></li>
                        <li><a href=”#”>O nas</a></li>
                        <li><a href=”#”>Oferta</a></li>
                        <li><a href=”#”>kontakt</a></li>
                </ul>
        </div>
        <div id=”content”>
                <h2>
                        Tytuł artykułu
                </h2>
                <p>
                        Lorem ipsum dolor sit amet consect etuer adipi scing elit sed diam nonummy nibh euismod tinunt ut laoreet dolore magna aliquam erat volut. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi.
                </p>
                <p>
                        Lorem ipsum dolor sit amet consect etuer adipi scing elit sed diam nonummy nibh euismod tinunt ut laoreet dolore magna aliquam erat volut. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla
```

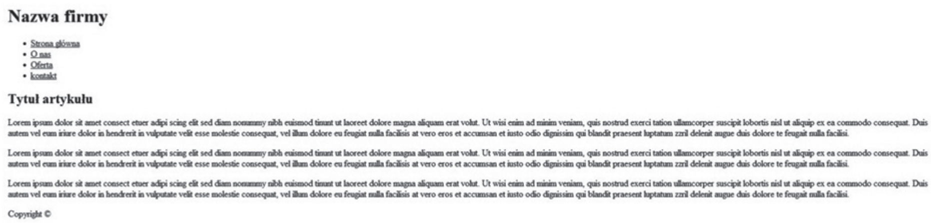
```
                    </p>
                    <p>
                        Lorem ipsum dolor sit amet consect etuer adipi scing elit sed
diam nonummy nibh euismod tinunt ut laoreet dolore magna aliquam erat volut. Ut wisi enim
ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex
ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse
molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto
odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla
facilisi.
                    </p>

            </div>
            <div id="footer">
                Copyright ©
            </div>
    </div>
    </body>
            </html>
```

Warto zwrócić uwagę na 2 elementy:

1. Będziemy musieli utworzyć arkusz CSS, który nazywa się slyle.css i jest umiejscowiony w katalogu głównym. Oczywiście nic nie stoi na przeszkodzie, żeby nadać mu inną nazwę lub lokalizację.
2. Dobrą praktyką jest wprowadzania angielskich nazw pod elementy layoutu. Wynika to z tego, że systemy CMS są tworzone z myślą o międzynarodowym użytkowniku i posługując się Wordpressem czy Joomlą właśnie na takie terminy trafimy. Warto więc przyzwyczajać się do tego, że:

„container" – to kontener, czyli zewnętrzny div, w który wtłacza się inne divy jak do pudełka,
„header"– nagłówek, czasami jednak to coś ważniejszego niż element layoutu, w systemach CMS header może zawierać elementy stałe dla wszystkich stron,
„navigation" – równie dobrze można napisać „menu",
„content" – zawartość, treść, choć słowo „kontent" przyjęło się już w języku polskim,
„footer" – stopka.

Mamy już stworzony szkielet – jak będzie wyglądał bez CSS? Chyba mało atrakcyjnie...
Pierwszy rzut oka wystarczy, aby zobaczyć, że owszem mamy jakieś elementy, ale one układają się domyślnie jeden pod drugim, a graficznie wyglądają mało interesująco.



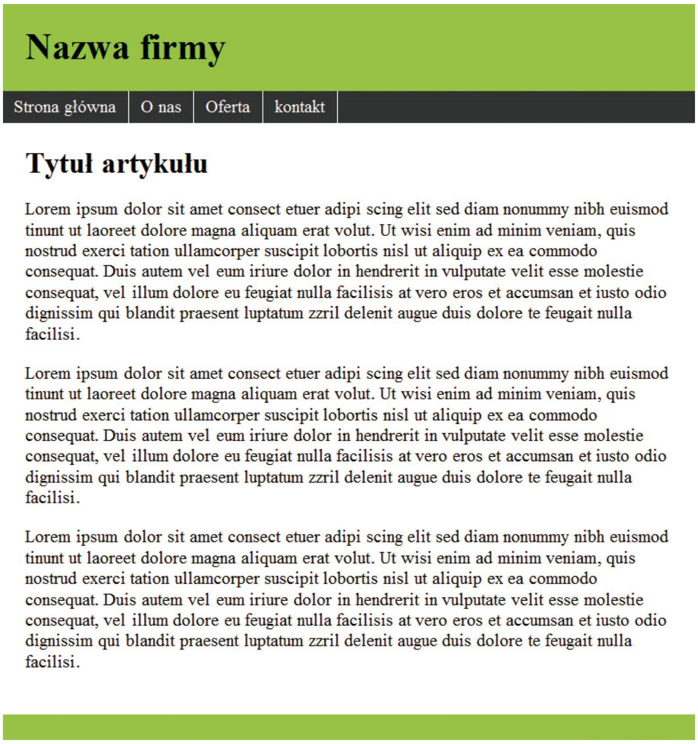
Dopiszmy więc arkusz stylów, nazwijmy go arkusz.css i zapiszmy w tym samym katalogu, co plik html.

Przykładowy arkusz stylów:

```
#container
{
        margin: 0 auto;
        width: 600px;
        background:#fff;
}

#header
{
        background:#92e94b;
        padding: 20px;
}

#header h1 { margin: 0; }

#navigation
{
        float: left;
        width: 600px;
        background:#333;
}

#navigation ul
{
        margin: 0;
        padding: 0;
}
```

```
#navigation ul li
{
        list-style-type: none;
        display: inline;
}

#navigation li a
{
        display: block;
        float: left;
        padding: 5px 10px;
        color:#fff;
        text-decoration: none;
        border-right: 1px solid#fff;
}

#navigation li a:hover { background:#383; }

#content
{
        clear: left;
        padding: 20px;
}

#content h2
{
        color:#000;
        font-size: 160%;
        margin: 0 0 .5em;
}

#footer
{
        background:#92e94b;
        text-align: right;
        padding: 20px;
        height: 1%;
}
```

Efekt powinien być taki:



Co warto skomentować?

- #container {width: 600px;} – czyli taka jest szerokość kontenera, a więc to, co w środku będzie się w tej szerokości zawierać, o ile nie opiszemy inaczej przy konkretnym elemencie.
- #navigation ul li { list-style-type: none; display: inline;} – ten fragment oznacza, że menu to po prostu lista wypunktowania, ale bez punktorów, a w zamian ustawiona nie z góry na dół, tylko w linii.

Layout div nie może obyć się bez elementu float i clear.Float służy do tworzenia obiektów „pływających" od lewej lub prawej krawędzi. Dzięki temu, że sąsiednie elementy, mające nadany float divy, będą „pływać" (układać się) obok siebie.

W celu zachowania porządku stosuje się jeszcze clear, po to, aby zakończyć w danym miejscy proces „opływania". Bardzo często stopka zawiera atrybut clear: both, który oznacza, że nie będą na nią nachodziły żadne opływające się elementy. Clear: left będzie natomiast

oznaczał, że dany element będzie zsunięty poniżej elementów z atrybutem float:left, ale nie przeszkodzi to elementom z float:right opływać go po prawej. Te zjawiska będą z pewnością lepiej widoczne przy bardziej złożonych layoutach, ten powyżej jest na tyle prosty, że efekty te nie są zbyt wyraźne, ale zachęcamy do eksperymentowania.

Layout płynny oparty na div (HTML/CSS)

1. Cele lekcji

a) Wiadomości

Uczeń:

- wie, co oznacza pojęcie „layoutu”,
- wie, na czym polega layout oparty o div,
- zna pojęcia „container”, „header”, „footer”,
- wie, jak wygląda prawidłowa składnia CSS,
- wie, jak wygląda dokument bez styli i ten sam z podpiętym arkuszem,
- zna pojęcia „float” i „clear”.

b) Umiejętności

Uczeń potrafi:

- połączyć html z css,
- stworzyć poprawnie szablon html i arkusz css.

2. Metoda i forma pracy

Wykład, zajęcia praktyczne, praca grupowa.

3. Środki dydaktyczne

Pracownia komputerowa.

4. Czas trwania lekcji

45 minut

5. Przebieg lekcji

Pojęcie „layoutu”. Layout to geometryczny układ treści, ma on szczególne znaczenie w wypadku projektowania stron, ponieważ od niego zależy architektura serwisu. Niezwykle ważne jest zrozumienie na jakich zasadach funkcjonuje układ oparty o bloki div. „Płynny” layout to taki, którego szerokość dostosowuje się do szerokości okna przeglądarki.

Tworzymy szkielet html

```
<!DOCTYPE html PUBLIC „-//W3C//DTD XHTML 1.0 Transitional//EN”
   „http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd”>
<html xmlns=”http://www.w3.org/1999/xhtml” lang=”pl”>
<head>
<meta http-equiv=”Content-Type” content=”text/html; charset=utf-8”>
<link rel=”Stylesheet” type=”text/css” href=”style1.css” />
<title>Layout plynny</title>
```

```
</head>
<body>

<div id=”container”>
        <div id=”header”>
                <h1>
                        Nazwa firmy
                </h1>
        </div>
        <div id=”navigation”>
                <ul>
                        <li><a href=”#”>Strona główna</a></li>
                        <li><a href=”#”>O nas</a></li>
                        <li><a href=”#”>Oferta</a></li>
                        <li><a href=”#”>Kontakt</a></li>
                </ul>
        </div>
        <div id=”content-container1”>
                <div id=”content-container2”>
                        <div id=”menuboczne”>
                                <ul>
                                        <li><a href=”#”>Pozycja 1</a></li>
                                        <li><a href=”#”>Pozycja 2</a></li>
                                        <li><a href=”#”>Pozycja 3</a></li>
                                        <li><a href=”#”>Pozycja 4</a></li>
                                </ul>
                        </div>
                        <div id=”content”>
                                <h2>
                                        Page heading
                                </h2>
                                <p>
                                        Lorem ipsum dolor sit amet consect etuer adipi scing
```

elit sed diam nonummy nibh euismod tinunt ut laoreet dolore magna aliquam erat volut. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi.

```
                    </p>
                    <p>
                        Lorem ipsum dolor sit amet consect etuer adipi scing
elit sed diam nonummy nibh euismod tinunt ut laoreet dolore magna aliquam erat volut. Ut
wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut
aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate
velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan
et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait
nulla facilisi.
                    </p>
                    <p>
                        Lorem ipsum dolor sit amet consect etuer adipi scing
elit sed diam nonummy nibh euismod tinunt ut laoreet dolore magna aliquam erat volut. Ut
wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut
aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate
velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan
et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait
nulla facilisi.
                    </p>
                </div>
                <div id="prawy">
                    <h3>
                        Jakiś tytuł
                    </h3>
                    <p>
                        Duis autem vel eum iriure dolor in hendrerit in
vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros
et accumsan.
                    </p>
                </div>
                <div id="footer">
                    Copyright ©
                </div>
            </div>
        </div>
</div>
</body>
        </html>
```

Warto zwrócić uwagę na 2 elementy:

1. Będziemy musieli utworzyć arkusz CSS, który nazywa się style.css i jest umiejscowiony w katalogu głównym. Oczywiście nic nie stoi na przeszkodzie, żeby nadać mu inną nazwę lub lokalizację.

2. Dobrą praktyką jest wprowadzania angielskich nazw pod elementy layoutu. Wynika to z tego, że systemy CMS są tworzone z myślą o międzynarodowym użytkowniku i posługując się Wordpressem czy Joomlą właśnie na takie terminy trafimy. Warto więc przyzwyczajać się do tego, że:

„container" – to kontener, czyli zewnętrzny div, w który wtłacza się inne divy jak do pudełka

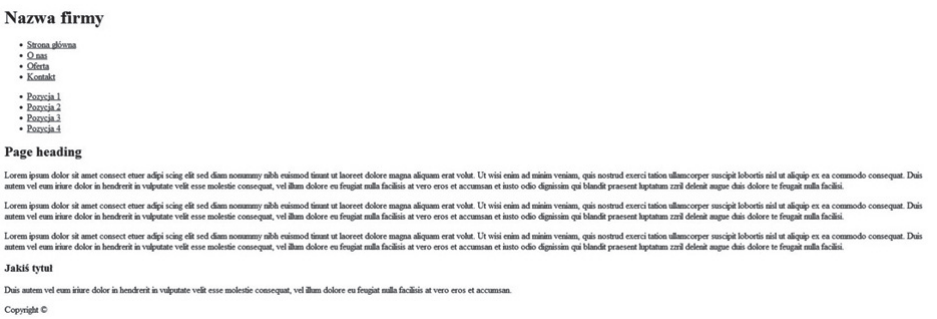
„header"– nagłówek, czasami jednak to coś ważniejszego niż element layoutu, w systemach CMS header może zawierać elementy stałe dla wszystkich stron,

„navigation" – równie dobrze można napisać „menu",

„content" – zawartość, treść, choć słowo „kontent" przyjęło się już w języku polskim,

„footer" – stopka.

Mamy już stworzony szkielet – jak będzie wyglądał bez CSS?



Dodajmy więc jakiś arkusz CSS i nazwijmy go style1.css

Przykładowy arkusz

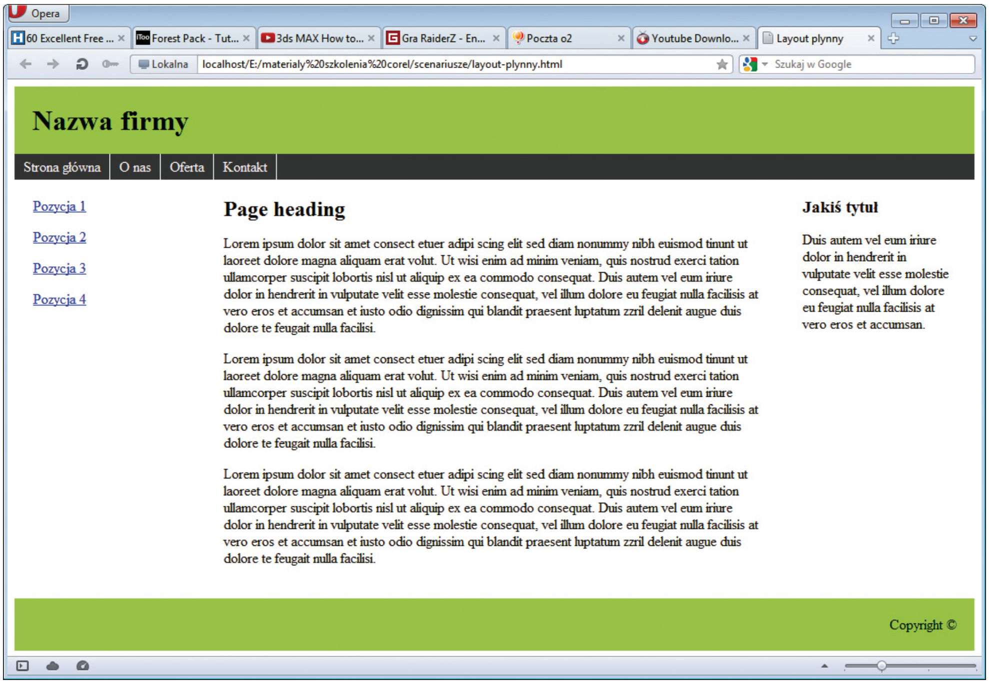
```
#container
{
        margin: 0 auto;
        width: 100%;
        background: #fff;
}

#header
{
```

```
        background: #92e94b;
        padding: 20px;
}

#header h1 { margin: 0; }

#navigation
{
        float: left;
        width: 100%;
        background: #333;
}

#navigation ul
{
        margin: 0;
        padding: 0;
}

#navigation ul li
{
        list-style-type: none;
        display: inline;
}

#navigation li a
{
        display: block;
        float: left;
        padding: 5px 10px;
        color: #fff;
        text-decoration: none;
        border-right: 1px solid #fff;
}

#navigation li a:hover { background: #383; }

#content-container1
```

```
{
        float: left;
        width: 100%;
        background: #fff ;repeat-y 20% 0;
}

#content-container2
{
        float: left;
        width: 100%;
        background: url; repeat-y 80% 0;
}

#menuboczne
{
        float: left;
        width: 16%;
        padding: 20px 0;
        margin: 0 2%;
        display: inline;
}

#menuboczne ul
{
        margin: 0;
        padding: 0;
}

#menuboczne ul li
{
        margin: 0 0 1em;
        padding: 0;
        list-style-type: none;
}

#content
{
        float: left;
```

```
        width: 56%;
        padding: 20px 0;
        margin: 0 0 0 2%;
}

#content h2 { margin: 0; }

#prawy
{
        float: right;
        width: 16%;
        padding: 20px 0;
        margin: 0 2% 0 0;
        display: inline;
}

#prawy h3 { margin: 0; }

#footer
{
        clear: both;
        background: #92e94b;
        text-align: right;
        padding: 20px;
        height: 1%;
}
```

Teraz efekt powinien wygladać tak:



Layout przylega do krawędzi przegladarki. Warto przy tej okazji zwrócić uwagę na parametr: #container{margin: 0 auto;width: 100%;}. Width: 100% pojawia się jeszcze w kilku innych miejscach i oznacza, że zawartość wypełnia okno przeglądarki od lewej do prawej.

Layout div nie może obyć się bez elementu float i clear.
Float służy do tworzenia obiektów „pływających" od lewej lub prawej krawędzi, dzięki temu, że sąsiednie elementy, mające nadany float divy, będą „pływać" (układać się) obok siebie.

W celu zachowania porządku stosuje się jeszcze clear, po to, aby zakończyć w danym miejscy proces „opływania". Mardzo często stopka zawiera atrybut clear: both, który oznacza, że nie będą na nią nachodziły żadne „opływające się" elementy. Clear: left będzie natomiast oznaczał, że dany element będzie zsunięty poniżej elementów z atrybutem float:left, ale nie przeszkodzi to elementom z float:right opływać go po prawej.

Pliki robots.txt i sitemap.xml

1. Cele lekcji

a) Wiadomości

Uczeń:

- wie, do czego służą pliki „robots.txt” oraz „sitemap.xml”,
- wie, jak wygląda składnia pliku „robots”,
- wie, jakich wpisów używać w pliku „sitemap”,
- zna przykłady takich plików.

b) Umiejętności

Uczeń potrafi:

- stworzyć plik dla robotów internetowych,
- stworzyć mapę strony,
- poszukać na witrynach internetowych plików „robots.txt” oraz „sitemap.xml”.

2. Metoda i forma pracy

Wykład, zajęcia praktyczne, praca grupowa.

3. Środki dydaktyczne

Pracownia komputerowa.

4. Czas trwania lekcji

45 minut

5. Przebieg lekcji

Wyjaśnienie celu istnienia tych plików na serwerze. Plik „robots.txt” służy do kontrolowania zachowania robotów internetowych na stronie www (ograniczenie dostępu do zasobów). Mapa strony (sitemap.xml) to plik zawierający spis wszyskich stron witryny wraz z dodatkowymi informacjami na ich temat (o czym niżej), ułatwiający robotom indeksowanie stron.

Do stworzenia pliku robots.txt wystarczy notatnik, ważne jest zrozumienie składni i funkcji poszczególnych wpisów.

User-agent: *
User-agent: Googlebot

User-agent – to nazwa robota, w wypadku „User-agent: *” mamy do czynienia ze wszystkimi robotami.

Disallow: /moje_pliki/

zabrania robotom odwiedzania tego folderu

Przykłady plików robots.txt

User-agent: *
Disallow:

oznacza, że każdy robot może odwiedzić każdy plik

User-agent: *
Disallow: /

zabrania wszystkim robotom odwiedzania wszystkich katalogów

User-agent: Googlebot
Disallow: /moje_pliki/

Googlebot nie ma dostępu do katalogu „moje_pliki”

User-agent: *
Disallow: /moje_pliki/profil.html

Żaden robot nie ma dostępu do tej konkretnej strony html

User-agent: ConveraCrawler
Disallow: /

Tylko ConveraCrawler nie może niczego pobrać

User-agent: googlebot-image
Disallow: /*.png$

Googlebot-image nie może pobierać plików png

Sitemap: http://www.mojawitryna.pl/sitemap.xml

Mapa strony jest dostępna właśnie pod tym adresem.

User-agent: *
Request-rate: 1/10
Visit-time: 0700-0900

Wszystkie roboty mogą pobierać tylko jedną stronę co 10 sekund, do tego w godzinach 7.00-9.00

Te informacje powinny wystarczyć, żeby stworzyć poprawny plik, np. dla strony szkoły.
Bardziej rozbudowany może być plik sitemap.xml

Należy pamietać o:
- sitemap.xml musi mieć kodowanie znaków UTF-8,
- musi zawierać pełne adresy URL,
- data i czas w formacie W3C Datetime.,- „priority" informuje o ważności strony w obrębie witryny, nie ma znaczenia dla indeksowania witryny,
- maksymalny rozmiar tego pliku to 10 MB (10 485 760 bajtów),
- maksymalna ilość adresów url to 50 000 URL.

Przykładowy plik ze strony sitemaps.org wygląda tak:

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
  <url>
    <loc>http://www.example.com/</loc>
    <lastmod>2005-01-01</lastmod>
    <changefreq>monthly</changefreq>
    <priority>0.8</priority>
  </url>
  <url>
    <loc>http://www.example.com/catalog?item=12&amp;desc=vacation_hawaii</loc>
    <changefreq>weekly</changefreq>
  </url>
  <url>
      <loc>http://www.example.com/catalog?item=73&amp;desc=vacation_new_zealand</
loc>
    <lastmod>2004-12-23</lastmod>
```

```
    <changefreq>weekly</changefreq>
  </url>
  <url>
      <loc>http://www.example.com/catalog?item=74&amp;desc=vacation_newfoundland</
loc>
    <lastmod>2004-12-23T18:00:15+00:00</lastmod>
    <priority>0.3</priority>
  </url>
  <url>
    <loc>http://www.example.com/catalog?item=83&amp;desc=vacation_usa</loc>
    <lastmod>2004-11-23</lastmod>
  </url>
</urlset>
```

Element „urlset" jest wymagany. Zawiera on wewnątrz wszystkie adresy URL, jakie są zawarte w mapie witryny.
Element „url" jest wymagany. Występuje wewnątrz „urlset". Opisuje on pojedynczy adres URL.
Element „loc" jest wymagany. Występuje wewnątrz „url". Ustala adres URL pojedynczego wpisu mapy.
Element „lastmod" jest opcjonalny. Występuje wewnątrz url. Ustala datę ostatniej modyfikacji dokumentu, którego adres URL jest podany w elemencie url.
Element „changefreq" jest opcjonalny. Występuje wyłącznie wewnątrz url. Ustala częstotliwość zmian dokumentu. Może przybierać wartości
always
hourly
daily
weekly
monthly
yearly
never
Element „priority" jest opcjonalny. Występuje wewnątrz url. Ustala ważność strony względem innych podstron serwisu. Wartość powinna być z zakresu od 0 do 1. Wartością domyślną jest 0.5.

Warto na koniec prześledzić różne witryny wpisując: nazwawitryny/robots.txt, nazwawitryny/sitemap.xml

Charakterystyka firm, w których zorganizowano staże w ramach projektu „Mistrz Kształcenia Zawodowego”

Poniżej przedstawiono charakterystykę firm, w których zorganizowano staże w ramach projektu „Mistrz Kształcenia Zawodowego” oraz opinie uczestników/uczestniczek i opiekunów staży.

1 SOTRONIC SP. Z O.O.

Firma SOTRONIC Sp. z o.o., działająca od 1994 roku w Krakowie, jest producentem i dystrybutorem: kabli krosowych, światłowodowych i miedzianych, pasywnych elementów do sieci teleinformatycznych (szaf, paneli, przełącznic i itp.), sprzętu telekomunikacyjnego i teleinformatycznego. SOTRONIC specjalizuje się w wykonawstwie usług światłowodowych, takich jak budowa linii i rurociągów światłowodowych, montaż złącz, spawy termiczne, pomiary reflektometryczne i interferometryczne, montaż przełącznic i inne.
Dodatkowo SOTRONIC Sp. z o.o. świadczy kompleksowe usługi projektowe i wykonawcze w zakresie instalacji światłowodowych, okablowania strukturalnego, systemów automatyki budynkowej i instalacji niskoprądowych. Firma dostarcza także urządzenia sieciowe do sieci LAN i WAN (switche, konwertery światłowodowe, transcievery, karty sieciowe, routery).
SOTRONIC zwraca bardzo dużą uwagę na wysoką jakość swoich produktów, którą osiąga dzięki stosowaniu najnowszych technologii produkcji i kontroli, a także wdrożonym nowoczesnym systemom zarządzania. Od czerwca 2004 posiada on certyfikat Systemu Zarządzania Jakością ISO 9001:2008.

Opinia stażysty:

W trakcie trwania stażu uczestnicy zapoznali się z technologią łącz światłowodowych i wykonywania instalacji sieci światłowodowej. Staż umożliwił zapoznanie się ze strukturą i działaniem firmy Sotronic i jej poszczególnych działów (zakresy współdziałania, obszary działań, współpraca). Dzięki niemu nauczyciele mieli dostęp do najnowszych technologii stosowanymi w IT; pozwolił również na poszerzenie wiedzy na temat najnowszych technologii

sieciowych. Uczestnicy podkreślali możliwość wykorzystania zdobytej wiedzy i umiejętności w trakcie lekcji z uczniami.

Opinia opiekuna stażu:
Opiekunowie oceniali pracę stażystów bardzo dobrze, na co wpływało duże zaangażowanie, sumienność i staranność wykonywanej pracy, punktualność oraz duża merytoryczna wiedza wyjściowa (teoretyczna). Staż pozwolił stażystom na zapoznanie się z realiami pracy informatyków w nowoczesnych firmach, stosujących współczesne technologie.

2. Agencja Reklamowa Giewont

Agencja Reklamowa Giewont działa od 1997 roku. Zakres działania firmy obejmuje: druk wielkoformatowy, offsetowy, cyfrowy; projekty graficzne: naklejki, etykietki, ulotki, plakaty, foldery, szyldy, reklamy świetlne, bannery; gadżety reklamowe. Firma wykonuje także usługi tworzenia profesjonalnej grafiki na strony internetowe oraz cięcie laserem.
Inwestycje w nowoczesne i profesjonalne urządzenia, współpraca z wyspecjalizowanymi podwykonawcami, korzystanie z wysokiej jakości materiałów i najnowszych technologii, pozwalają Agencji Reklamowej Giewont oferować klientom krótkie terminy realizacji oraz produkty najwyższej jakości.

Opinia stażystów:
Podczas stażu w firmie Giewont w Zakopanem podkreślali oni możliwość zapoznania się z sposobem działania i specyfikacją, zarządzaniem w firmie informatycznej. Firma ta oferuje szereg nowatorskich technik informatycznych w zakresie grafiki komputerowej oraz tworzenia stron WWW. Uczestnicy mieli także możliwość poznać nowoczesne techniki oraz technologie stosowane przy produkcji materiałów reklamowych. Odbyty staż zwiększył wiedzę stażystów z zakresu grafiki komputerowej, zasad produkcji materiałów reklamowych.

Opinia opiekuna stażu:
Podczas stażu w firmie Giewont stażyści zapoznali się ze specyfiką oraz zasadami funkcjonowania firmy informatycznej. Zdobyli oni nowe umiejętności w zakresie grafiki komputerowej, tworzenia stron WWW oraz w zakresie przygotowania, obróbki oraz druku materiałów graficznych (reklamowych). Stażyści zostali zapoznani z kadrami, biurem, działem sprzedaży, handlem, produkcją. Mieli możliwość konfrontacji wiedzy teoretycznej z praktyką. W ramach zajęć zrealizowali wszystkie działania przewidziane w programie stażu.

3. NET-O-LOGY

Firma NET-O-LOGY specjalizuje się w trzech zakresach w obszarze bezpieczeństwa i ciągłości działania w IT:

• doradztwo realizowane poprzez dzielenie się doświadczeniami z klientami, doradztwo w doborze rozwiązania, sposobie realizacji projektu czy badania efektywności projektu (TCO/ROI),

• wdrożenia, kompetencje w obszarze doradztwa wynikające ze świadczonych usług realizacji wdrożeń,

• usługi outsourcingu na potrzeby zarządzania IT i usług w chmurze.

NET-O-LOGY obsługuje ponad 500 stałych klientów. Bogate doświadczenie wdrożeniowe i biznesowe w zakresie obowiązujących standardów, otoczenia biznesowego oraz specyficznych wymagań dotyczących sposobu wykorzystania systemów informatycznych w różnych branżach, NET-O-LOGY zbudowała dzięki realizacji ponad 300 zaawansowanych wdrożeń systemów informatycznych. Klienci firmy w większości należą do sektora przemysłu i usług, ale NET-O-LOGY współpracuje też z sektorem bankowym i telekomunikacyjnym. Swoje usługi firma świadczy zarówno na rynku krajowym, jak i Unii Europejskiej oraz poza jej granicami.

Opinia stażystów:

Staż w firmie Net-o-logy w Katowicach przebiegł w miłej atmosferze. Personel firmy zawsze służył pomocą, a praktyka przebiegła zgodnie z wytycznymi projektu i pozwoliła zgłębić wiadomości z dziedziny sieci i systemów komputerowych. Główny nacisk firma kładła na zdobywanie praktycznych umiejętności związanych z budową sieci i systemów komputerowych oraz ich integracją.

Opinia Opiekuna stażu:

Stażyści zdobywali umiejętności praktyczne związane z systemami komputerowymi i budową sieci komputerowych.

4. F.H.U. Privus

PRIVUS świadczy usługi w zakresie systemów zabezpieczeń mienia SSWiN oraz monitoringu kamer CCTV i CCTV-IP. Firma oferuje wsparcie techniczne IT dla klientów indywidualnych oraz małych i średnich firm (serwis komputerów i laptopów, naprawa oprogramowania, instalacja i konfiguracja systemów operacyjnych, tworzenie i konfiguracja sieci komputerowych, sprzedaż sprzętu komputerowego i akcesoriów).

System monitoringu kamer CCTV/CCTV-IP proponowany przez firmę umożliwia podgląd i zarządzanie za pomocą TV, komputera lub telefonu komórkowego z każdego miejsca na świecie z dostępem do Internetu. W ofercie firmy istnieje sprzęt i oprogramowanie do kontroli dostępu, rejestracji czasu pracy oraz oprogramowanie PRZEDSZKOLE do ewidencji pobytu dziecka w placówce przedszkolnej.

Opinia Opiekuna stażu:

Stażyści to grupa odpowiedzialna, ambitna, zdyscyplinowana, wykonująca powierzone zadania sumiennie. Grupa potrafiła w bardzo dobry sposób połączyć wiedzę merytoryczną z praktyką. Podczas stażu w firmie „Privus" stażyści zapoznali się ze specyfiką oraz zasadami funkcjonowania firmy informatycznej. Zdobyli oni umiejętności w zakresie: budowy sieci komputerowych, bezpieczeństwa systemów informatycznych, administracji serwerami sieciowymi, technologii sieci rozległych, systemów zabezpieczeń mienia. Stażyści zostali zapoznani ze wszystkimi działami, m.in.: kadry, biuro, dział sprzedaży, dział handlowy oraz dział produkcyjny. Mieli możliwość konfrontacji wiedzy merytorycznej z praktyką. W ramach zajęć zrealizowali wszystkie działania przewidziane w programie stażu.

Opinia stażystów:

Stażyści mieli okazję zapoznania się z nowoczesnymi technologiami sieci komputerowych, systemów serwerowych, monitoringu zabezpieczenia mienia. Zdobyte umiejętności praktyczne zostaną wykorzystane na zajęciach laboratoryjnych z młodzieżą. Stażyści podkreślali, że połączenie wiedzy teoretycznej z praktyczną, przyniosło wymierne korzyści w postaci doświadczenia praktycznego, co spowoduje podniesienie jakości pracy szkoły.

Stażyści wyrażali swoje zadowolenie z oferty firmy Privus - dostęp do nowoczesnych technologii teleinformatycznych, kompetentny zespół, dobre merytoryczne przygotowanie opiekuna stażu. W dużej mierze kocentrowali się oni na zadaniach w obszarach: sieci komputerowych, routerów serwerowych, zabezpieczeń.

5. MATINTERNET S.C.

Firma MATINTERNET S.C istnieje od 1999 roku. Głównym profilem działalności jest projektowanie i tworzenie serwisów WWW. Firma świadczy także usługi hostingowe. MATINTERNET opracowała system CMS oraz e-sklep. W jej ofercie znajduje się również duża gama usług multimedialnych: tworzenie filmów w jakości HD, wykonywanie panoram sferycznych, wycieczki wirtualnych, animacje Flash i inne.

Oferta MATINTERNET opiera się o autorskie rozwiązania, dające firmie niezależność oraz brak ograniczeń, a dzięki temu również swobodę w budowaniu oferty i elastyczność w działaniu. Do głównych atutów firmy należy autonomia technologiczna, innowacyjność oraz młoda wykwalifikowana kadra.

Opinia stażysty:

W firmie Matinternet stażyści zapoznali się ze sposobem działania i funkcjonowania firmy informatycznej, oferującej szereg usług w zakresie tworzenie grafiki komputerowej, stron internetowych, wirtualnych wycieczek oraz składania i dorabiania filmów. Mieli okazje poznać nowe rozwiązania oraz technologie stosowane przez firmę, poszerzali również swoją wiedzę w zakresie budowy stron WWW w technologiach PHP, DHTML, XML, JAVA SCRIPT i FLASH. Odbyty staż poszerzył poziom umiejętności w zakresie tworzenia stron za pomocą systemu CMS. Uczestnicy zwracali także uwagę na dobrą atmosferę w firmie.

Opinia opiekuna stażu:

Podczas praktyk stażyści poznali funkcjonowanie i specyfikację działania firmy informatycznej. Zdobyli doświadczenie w posługiwaniu się autorskim systemem zarządzania treścią, jakim jest ArtCMSZ. Zapoznali się również z obróbką grafiki na potrzeby stron WWW oraz ich efektywną optymalizacją. Stażyści poznali technologię Fly-Web-Tour, czyli połączenie klasycznej strony WWW z wirtualną wycieczką, brali udział w tworzeniu prezentacji w formie filmów, panoram sferycznych i animacji. Byli oni otwarci na nowe doświadczenia i chętnie angażowali się w powierzone projekty.

6. TVN S.A. – oddział w Krakowie

Grupa TVN to polska grupa medialno-rozrywkowa należąca do Grupy ITI. W skład grupy ITI wchodzą m.in. stacje telewizyjne (np. TVN, TVN24, TVN Style), portal internetowy (Onet.pl) oraz platforma cyfrowa („n"). Siedzibę główną TVN stanowi kompleks budynków Grupy ITI w Warszawie. Poza Warszawą najważniejszym ośrodkiem TVN jest Kraków.

W Krakowie, liczący około 300 osób zespół, przygotowuje program TVN, TVN Siedem oraz TVN24. Tutaj powstaje ponad połowa własnych produkcji stacji. W oddziale znajduje się m.in. redakcja „Rozmów w Toku", „Uwaga", „Sperwizjer", „Studio Grafiki", „Call TV", czyli teleturnieje interaktywne, „Zmagania Miasta", „Siłacze", „Granice", „Nie do wiary", dział informatyczny oraz księgowość. Programy nagrywane są w pięciu pomieszczeniach studyjnych.

TVN angażuje się także w różnego rodzaju projekty i działania społeczne; uczestniczył na przykład w realizacji projektu mającego na celu stworzenie pilotażowego programu multimedialnych lekcji - „EDU KINO". Do przygotowania programu multimedialnych lekcji zostały wykorzystane możliwości technologii, w które zostały wyposażone małopolskie kina. Projekcje filmów dokumentalnych w połączeniu z materiałami dydaktycznymi (scenariusze lekcji) to doskonały sposób na przeprowadzenie interesującej lekcji, zwłaszcza w połączeniu z wykładami naukowców, twórców i literatów, nadawanymi na żywo do kin ze studia TVN.

TVN wyznacza nowe trendy i dba o zachowanie najwyższych standardów. Troska o dobrą organizację i zarządzanie oraz duży nacisk na komunikację strategii i celów na każdym poziomie organizacji, przekłada się na pozycję TVN na rynku oraz na przyjazne warunki pracy.

Opinia Opiekuna stażu:

Stażyści realizowali swoje zadania terminowo, bezbłędnie na poziomie merytorycznym i technicznym, z zaangażowaniem i pasją. Jednocześnie dzielili się swoimi spostrzeżeniami w grupie stażowej, co pobudzało do ciekawych dyskusji. Grupy stażowe wykazywały duże zainteresowanie stażem z analitycznym podejściem do zadań, aktywnie modyfikowali plany pracy do możliwości i założeń programowych. Duże zainteresowanie budziła zwłaszcza praca działu programistów. Należy podkreślić i bardzo dobrze ocenić pracę zespołową wewnątrz grup stażowych oraz pracę grup stażowych z działami TVN - nawiązanie relacji z pracownikami TVN (współdziałanie przy realizacji stażu). Uczestnicy/uczestniczki wykonywali różnego rodzaju zadania, casy – zgodnie z założeniami. Spora grupa była zainteresowana tematami pracy on-line w firmie.

Opinia stażystów:

Poziom merytoryczny staży został oceniony bardzo wysoko przez wszystkich. Uczestnicy/uczestniczki bardzo zadowoleni ze staży w TVN. Podkreślali możliwość zobaczenia działania nowoczesnego, dużego przedsiębiorstwa. Zajęcia obejmowały m.in.:

- spotkania organizacyjne, zwiedzanie hal produkcyjnych („Julia", „Master Chef"),

- wprowadzenie do zagadnień telewizyjnych, ochrony praw autorskich,

- przedstawienie metod badania oglądalności z wykorzystaniem wskaźników statystycznych,

- zapoznanie się z pracą zespołu programistów online oraz jej efektami oraz działu IT,

- zapoznanie się z podstawami wprowadzającymi do zasad samodzielnego wyszukiwania materiałów w bazach MEDIA AECHIVE i innych,

- zasady opisywania materiałów, współpraca z grafikami, programistami i studiem FX itp.

Staż dawał szansę na poznanie nowoczesnych technologii informatycznych, wykorzystywanych w programach telewizyjnych.

Umiejętności ważne z punktu widzenia uczestników/uczestniczek:

- *praktyczne zastosowanie wiedzy informatycznej w korporacji,*
- *umiejętności pracy zespołowej w dużej firmie i między różnymi działami przedsiębiorstwa,*
- *samodzielne wyszukiwania i wykorzystywanie baz danych,*
- *sposoby opisywania dokumentacji on-line/oraz papierowej,*
- *wskazanie obszarów w firmie, które w specyficznych sposób angażują specjalizacje z branży IT,*
- *umiejętność pracy na styku działów, w tym programistów.*

7. i3D Sp. z o.o. – Wizualizacje interaktywne

i3D sp. z o.o. to grupa spółek wykorzystujących i zajmujących się rozwojem technologii wirtualnej rzeczywistości. To lider w tej branży w Polsce i tej części Europy. Świadczy usługi w zakresie grafiki komputerowej, tworzy interaktywne i trójwymiarowe wizualizacje dowolnych obiektów. Animacje, trójwymiarowe prezentacje, konfiguratory, symulatory, interaktywne strony WWW wykorzystywane są m.in. do prezentacji, szkoleń, w sprzedaży i marketingu czy wzornictwie. Zespół tworzą najlepsi graficy komputerowi i programiści. i3D Inc. – to oddział firmy i3D zarejestrowany w Houston w stanie Texas. To jedyna firma w Polsce oferującą w pełni interaktywne, trójwymiarowe wizualizacje i prezentacje dowolnych modeli i obiektów, takich jak budynki, budowle, urządzenia, linie produkcyjne, środki transportu itp. Wizualizacje wykorzystywane mogą być: w promocji produktów, przy inwentaryzacji obiektów, jako instrukcje obsługowe czy serwisowe, symulatory, konfiguratory itp. Oferta firmy skierowana jest zarówno do biznesu, jak i samorządów terytorialnych, agencji rządowych oraz ośrodków naukowych.

Przy realizacji projektów i3D wykorzystuje najnowocześniejsze na skalę światową oprogramowanie oraz narzędzia wizualizacyjne. Wspólnie z EON Reality, Inc. (dostawcą oprogramowania) oraz konsorcjum firm Microsoft, Christie Digital, NVidia, Philips i Hewlett-Packard (dostawcami sprzętu), powołują pierwsze w Polsce i w tej części Europy Centrum Wizualizacji Interaktywnych, IDC - Interactive Digital Center. W ramach tej inwestycji, wartej ponad 4 mln EURO, na terenie parku technologicznego, zainstalowano kompleksowe rozwiązania sprzętowe do interaktywnych prezentacji 3D, m.in. jedyną w Polsce salę kinową z ekranem cylindrycznym typu ConCave. Ultranowoczesna sala prezentacyjna daje niezwykłe efekty wizualne, zwłaszcza przy projekcjach trójwymiarowych.

Opinie uczestniczek o stażu:

Staż w Technoparku w Gliwicach pozwolił na zapoznanie się z realiami pracy informatyków w nowoczesnych firmach, stosujących współczesne i innowacyjne rozwiązania technologiczne. Dał on możliwość poszerzenia wiedzy z zakresu funkcjonowania firmy, marketingu, księgowości, działu projektów unijnych, a także możliwość obserwacji pracy nowoczesnych maszyn. Kontakt ze specjalistami branży był cennym uzupełnieniem wiedzy, a otrzymane wskazówki niezwykle wartościowe do wprowadzenia do programu nauczania w zawodzie technik informatyk. Staż to dobra konfrontacja treści kształcenia z realiami rynkowymi i wymaganiami pracodawców. Przyjazna atmosfera pracy, zespołowość, pomoc opiekuna stażu były wartością dodaną stażu.

Uczestniczki oceniały staż w ankiecie ewaluacyjnej dobrze i bardzo dobrze. W zakresie przydatności wiedzy i umiejętności pozyskanej na stażu bardzo dobrze, zaznaczając, że

staż odbywał się w firmie, w której ich uczniowie poszukują pracy, co pozwoli im - jako nauczycielom - przygotować ich lepiej pod potrzeby firmy. Podkreślana przez uczestniczki była także możliwość zobaczenia od środka procesów w różnych działach: marketing, księgowość, dział projektowy – tj. funkcjonowanie firmy IT w praktyce, w realizacji działań dnia codziennego. Informacje pozyskane na stażu będą stanowiły materiał do wykorzystania w procesie nauczania w zawodzie technik informatyk.

Opinia Opiekuna stażu:

Opiekun staży w i3D w Gliwicach podkreślał zaangażowanie uczestniczek, otwartość na wiedzę, sumienność i staranność w realizowanych działaniach. Jednocześnie zaznaczył dużą wiedzę wejścia uczestniczek (wiedza teoretyczna – szeroka z branży informatycznej). Z punktu widzenia Opiekuna stażu program był dobrze dopasowany do potrzeb uczestniczek oraz możliwości/zasobów firmy. Nie zgłoszono żadnych rekomendacji do zmian w przebiegu stażu.