INFORMATYKA – MÓJ SPOSÓB NA POZNANIE I OPISANIE ŚWIATA

PROGRAM NAUCZANIA INFORMATYKI Z ELEMENTAMI PRZEDMIOTÓW MATEMATYCZNO-PRZYRODNICZYCH

Informatyka – poziom rozszerzony

Razem można więcej – podstawy pracy grupowej

Andrzej Ptasznik

Człowiek - najlepsza inwestycja

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Warszawska Wyższa Szkoła Informatyki

UNIA EUROPEJSKA
EUROPEJSKI FUNDUSZ SPOŁECZNY

Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

Tytuł: ***Razem można więcej – podstawy pracy grupowej***

Autor: ***Andrzej Ptasznik***

Redaktor merytoryczny: ***prof. dr hab. Maciej M. Sysło***

Materiał dydaktyczny opracowany w ramach projektu edukacyjnego
Informatyka – mój sposób na poznanie i opisanie świata.
Program nauczania informatyki z elementami przedmiotów matematyczno-przyrodniczych

www.info-plus.wwsi.edu.pl

infoplus@wwsi.edu.pl

Wydawca: Warszawska Wyższa Szkoła Informatyki
ul. Lewartowskiego 17, 00-169 Warszawa
www.wwsi.edu.pl
rektorat@wwsi.edu.pl

Projekt graficzny: *Marzena Kamasa*

Warszawa 2013

Człowiek - najlepsza inwestycja

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Warszawska Wyższa Szkoła Informatyki

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

SCENARIUSZ TEMATYCZNY

RAZEM MOŻNA WIĘCEJ – PODSTAWY PRACY GRUPOWEJ

INFORMATYKA – POZIOM ROZSZERZONY

OPRACOWANY W RAMACH PROJEKTU:

INFORMATYKA – MÓJ SPOSÓB NA POZNANIE I OPISANIE ŚWIATA.

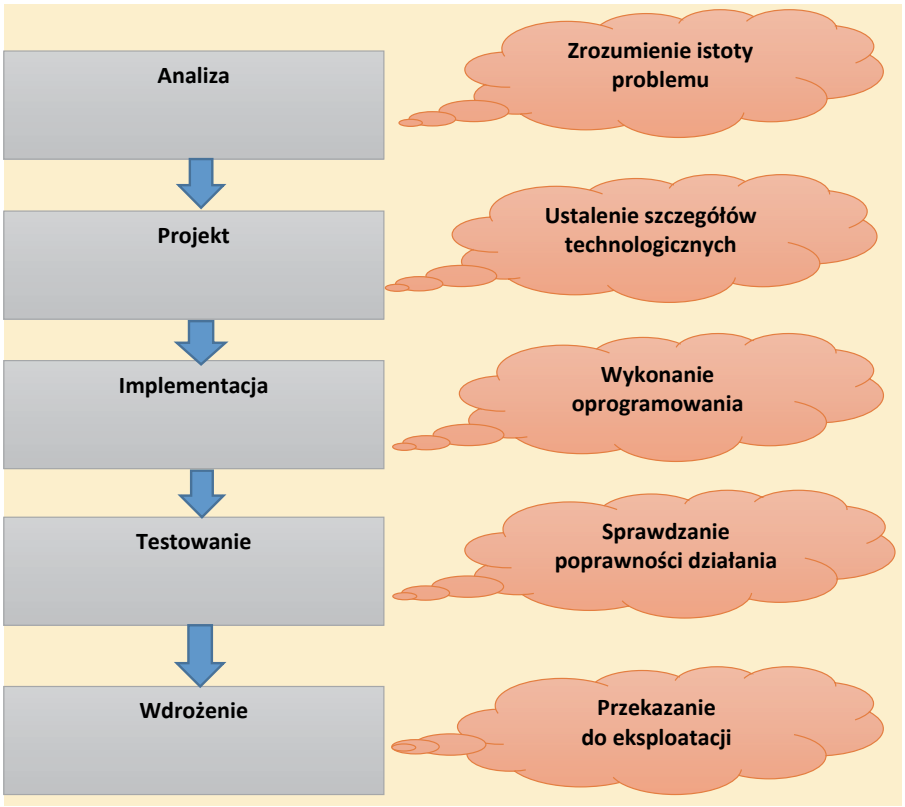
PROGRAM NAUCZANIA INFORMATYKI

Z ELEMENTAMI PRZEDMIOTÓW MATEMATYCZNO-PRZYRODNICZYCH

Streszczenie

Profesjonalne tworzenie oprogramowania jest procesem bardzo złożonym i wymaga stosowania różnych technologii wspierających.

Na rysunku 1 pokazano schemat cyklu tworzenia oprogramowania.



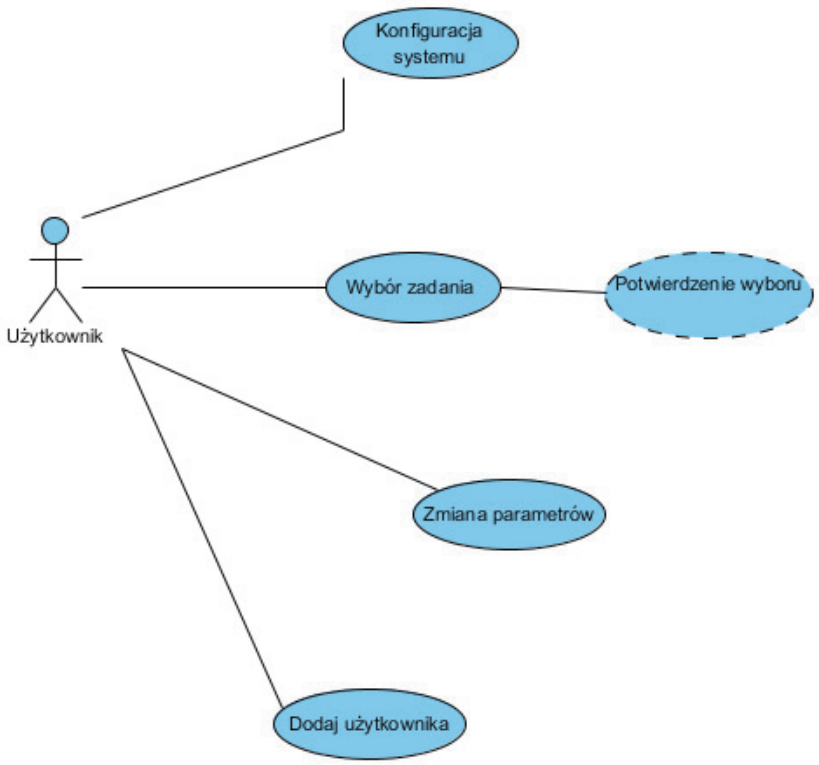
Rysunek 1. Cykl tworzenia oprogramowania.

W kolejnych etapach cyklu pokazanego na rysunku 1 realizowane są różne zadania, których celem jest wytworzenie poprawnie działającego oprogramowania spełniającego wszystkie przyjęte założenia. W procesie wytwarzania oprogramowania uczestniczy wielu specjalistów, zatem bardzo istotne jest zapewnienie właściwej komunikacji w zespole oraz jednolita interpretacja pojęć, które występują w problemie.

Do zarządzania procesem tworzenia oprogramowania stosuje się narzędzia CASE (ang. *Computer Aided Software Engineering*), czyli rozbudowane systemy przeznaczone do wspomagania czynności realizowanych w poszczególnych etapach procesu tworzenia oprogramowania. Dzięki nim projekty tworzy się szybciej i można synchronizować pracę dużej liczby specjalistów wykonujących poszczególne elementy systemu. Do prezentowania wyników analizy stosuje się różne standardy i metodyki. Powszechnie stosowany jest język modelowania UML (ang. *Unified Modeling Language*). Na etapie analizy i projektu systemu tworzy się różne opisy wykorzystując notację języka UML.

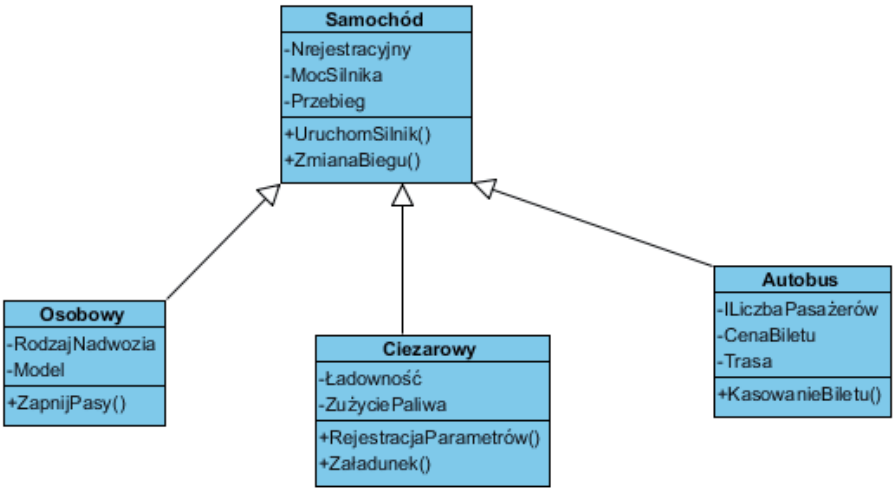
Modele UML tworzone są głównie z dwóch powodów: dla lepszego zrozumienia problemu oraz umożliwienia wymiany informacji pomiędzy analitykami, projektantami i programistami. Celem modelowania jest identyfikacja wszystkich czynników, które mogą wpłynąć na realizację projektu.

Na rysunku 2 pokazano przykładowy diagram przypadków użycia, którego zadaniem jest pokazanie, jakie zadania będą mogli wykonywać różni użytkownicy systemu.



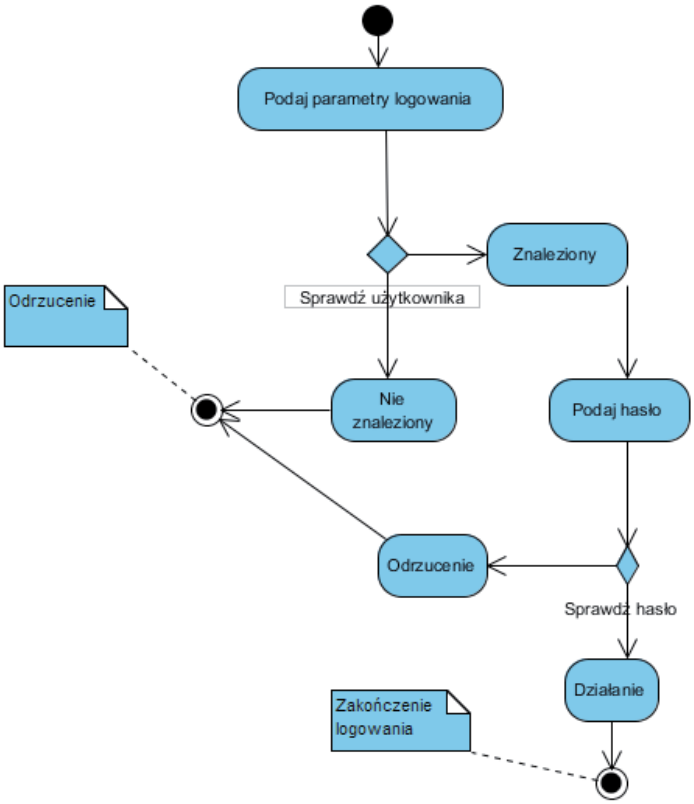
Rysunek 2. Diagram przypadków użycia.

Do zobrazowania struktury systemu i występujących zależności stosuje się diagramy klas. Na rysunku 3 pokazano przykładowy diagram klas wykonany w notacji UML.



Rysunek 3. Diagram klas.

Do wyrażenia sposobu realizacji poszczególnych zadań systemu stosuje się diagramy czynności, zwane także diagramami aktywności. Na rysunku 4 pokazano przykładowy diagram czynności.



Rysunek 4. Diagram czynności.

W języku UML można tworzyć diagramy, które razem mogą stanowić kompletny opis projektowanego systemu.

Projektowanie systemów informatycznych to zadanie bardzo złożone. Duża liczba zależności, zasad i stosowanych algorytmów powoduje, że nie jest możliwe podejście do tego zadania bez niezbędnego przygotowania i bez wykorzystywania różnych technologii wspomagających ten proces.

Czas realizacji

2 x 45 minut

Tematy lekcji

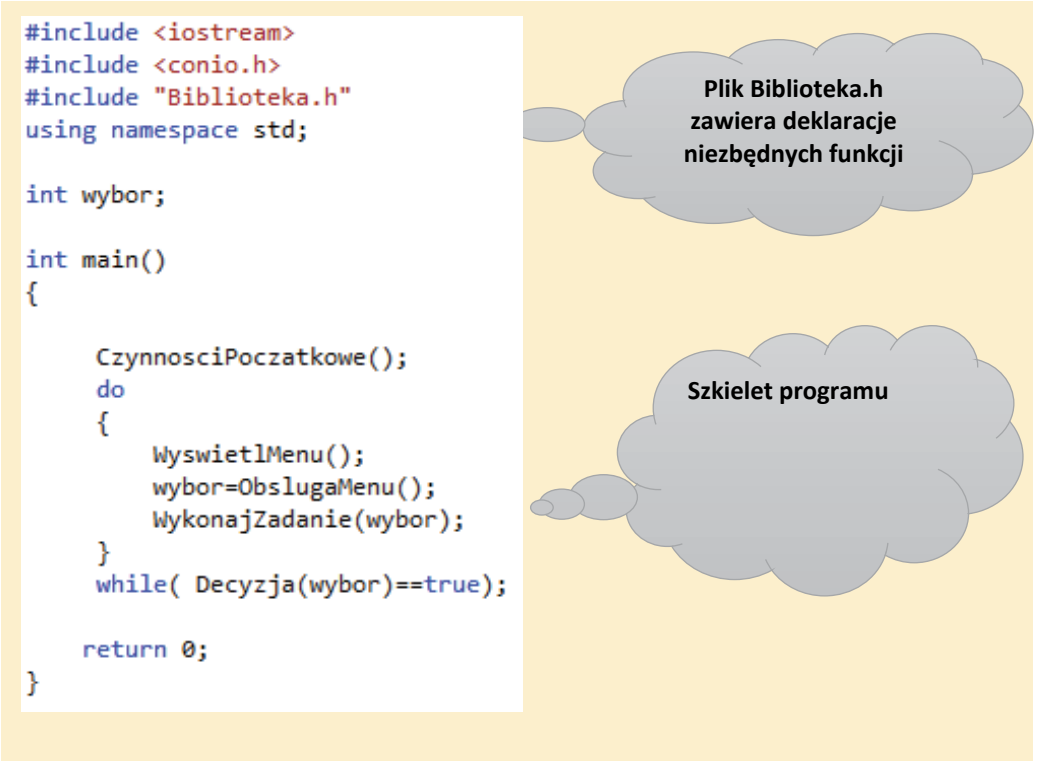
1. Podzielić i przydzielić.
2. Połączyć i sprawdzić.

LEKCJA NR 1

TAMAT: Podzielić i przydzielić

Streszczenie

W trakcie lekcji należy wspólnie z uczniami przygotować szkielet programu, a następnie przydzielić różnym grupom uczniów do wykonania wybrane elementy programu. Na rysunku 5 pokazano przykład programu, który może być wykorzystany do realizacji tego zadania.



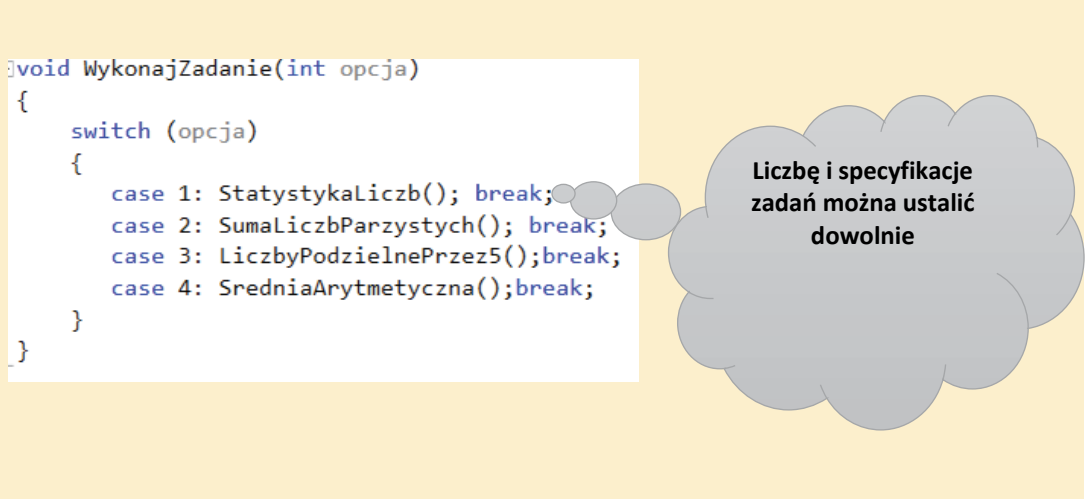
Rysunek 5. Szkielet programu.

Przed przydzieleniem zadania zespołowi uczniów należy opisać działanie poszczególnych funkcji wykorzystanych w programie.

Na szkielecie programu przedstawionym na rysunku 5 zostały wykorzystane funkcje:

- CzynnosciPoczatkowe() – funkcja powinna wykonać wszystkie zadania niezbędne do prawidłowego działania programu,
- WyswietlMenu() – funkcja powinna wyświetlić na ekranie opcje do wyboru, parametrami funkcji są współrzędne potrzebne do wyświetlenia menu,
- ObslugaMenu() – zadaniem funkcji jest określenie, którą opcję wybrał użytkownik,
- WykonajZadanie(opcja) – wykonanie zadanie w zależności od wybranej opcji,
- Decyzja() – zadaniem funkcji jest ustalenie, czy program ma być kontynuowany, czy zakończony.

Podstawowe działanie zaproponowanego programu polega na obsłudze menu i wykonywaniu wybranego zadania. Liczba zadań, które mogą być wykonywane, jest dowolna i mogą to być różnego typu obliczenia na tablicy liczb całkowitych. Na rysunku 6 pokazano funkcję WykonajZadanie(opcja), która w zależności od wartości przekazanego parametru wykonuje wybraną funkcję.



Rysunek 6. Funkcja WykonajZadanie(opcja).

Wszystkie funkcje wykorzystywane w programie należy zapisać w pliku Biblioteka.h. Przydzielając uczniowi do wykonania odpowiednią funkcję, należy określić minimalny zakres jej działania, pozostawiając wykonawcy ustalenie szczegółów. W celu usprawnienia pracy należy jednego z uczniów wybrać jako kierownika projektu i przydzielić mu obowiązki związane z koordynacją poszczególnych zadań.

Podstawa programowa

Etap edukacyjny: IV, przedmiot: informatyka (poziom rozszerzony)

Cele kształcenia – wymagania ogólne

III. Rozwiązywanie problemów i podejmowanie decyzji z wykorzystaniem komputera, stosowanie podejścia algorytmicznego.

Treści nauczania – wymagania szczegółowe

Uczeń:

1) analizuje, modeluje i rozwiązuje sytuacje problemowe z różnych dziedzin;
2) stosuje podejście algorytmiczne do rozwiązywania problemu;
3) formułuje przykłady sytuacji problemowych, których rozwiązanie wymaga podejścia algorytmicznego i użycia komputera;
4) dobiera efektywny algorytm do rozwiązania sytuacji problemowej i zapisuje go w wybranej notacji;
5) posługuje się podstawowymi technikami algorytmicznymi;
6) ocenia własności rozwiązania algorytmicznego (komputerowego), np. zgodność ze specyfikacją, efektywność działania;
7) opracowuje i przeprowadza wszystkie etapy prowadzące do otrzymania poprawnego rozwiązania problemu: od sformułowania specyfikacji problemu po testowa nie rozwiązania;
13) stosuje metodę zstępującą i wstępującą przy rozwiązywaniu problemu;
28) realizuje indywidualnie lub zespołowo projekt programistyczny z wydzieleniem jego modułów, w ramach pracy zespołowej, dokumentuje pracę zespołu.

Cel

Podstawowym celem lekcji jest wyjaśnienie podstaw pracy grupowej przy wytwarzaniu oprogramowania. Realizacja zadania w formie pracy grupowej. Zadanie realizowane będzie w języku C++.

Słowa kluczowe

praca grupowa, narzędzia CASE, Visual Studio, UML

Co przygotować?

- Zainstalować Microsoft Visual Studio Express 2012 for Windows Desktop – oprogramowanie, które jest darmową wersją pakietu Visual Studio 2012, można pobrać pod adresem internetowym: http://www.microsoft.com/en-us/download/details.aspx?id=40787

  Nauczyciel może wykorzystać inne środowisko programistyczne, o ile jest mu dobrze znane.

Przebieg zajęć

Wprowadzenie (15 minut)

1. Na początku zajęć nauczyciel podaje temat lekcji i omawia krótko podstawowe pojęcia związane z pracą grupową nad oprogramowaniem.
2. Po omówieniu podstaw, nauczyciel przystępuje do opisu zadania, które będzie realizowane, zwracając szczególną uwagę na fakt, że do jego realizacji problem zostanie podzielony i fragmenty zadania zostaną przydzielone do wykonania różnym grupom uczniów.
3. Na zakończenie tej części lekcji omawia sposób uruchomienia środowiska Visual Studio 2012 i przystępuje do realizacji zadania. Szczegóły zadania opisane zostały w streszczeniu do lekcji.

Zasadnicza część lekcji (25 minut)

W zasadniczej części lekcji wspólnie z uczniami omawiany jest przewidziany do wykonania problem. Wspólnie z uczniami przygotowany jest szkielet programu, w którym poszczególne elementy zadania zostaną zaprezentowane jako funkcje. Należy ustalić typ wartości zwracany przez funkcje oraz przyjmowane parametry. W kolejnym kroku należy omówić sposób działania poszczególnych funkcji i przydzielić je różnym grupom do realizacji.

Podsumowanie (5 minut)

W ramach dyskusji należy zwrócić szczególną uwagę na fakt, że wykonany wspólnie szkielet programu jest poprawny składniowo, poprawnie się kompiluje i po wykonaniu poszczególnych funkcji powinien realizować założone zadania.

Sprawdzenie wiedzy

Ocenianie

Ocena uczniów na podstawie udziału w realizacji elementów projektu.

Dostępne pliki

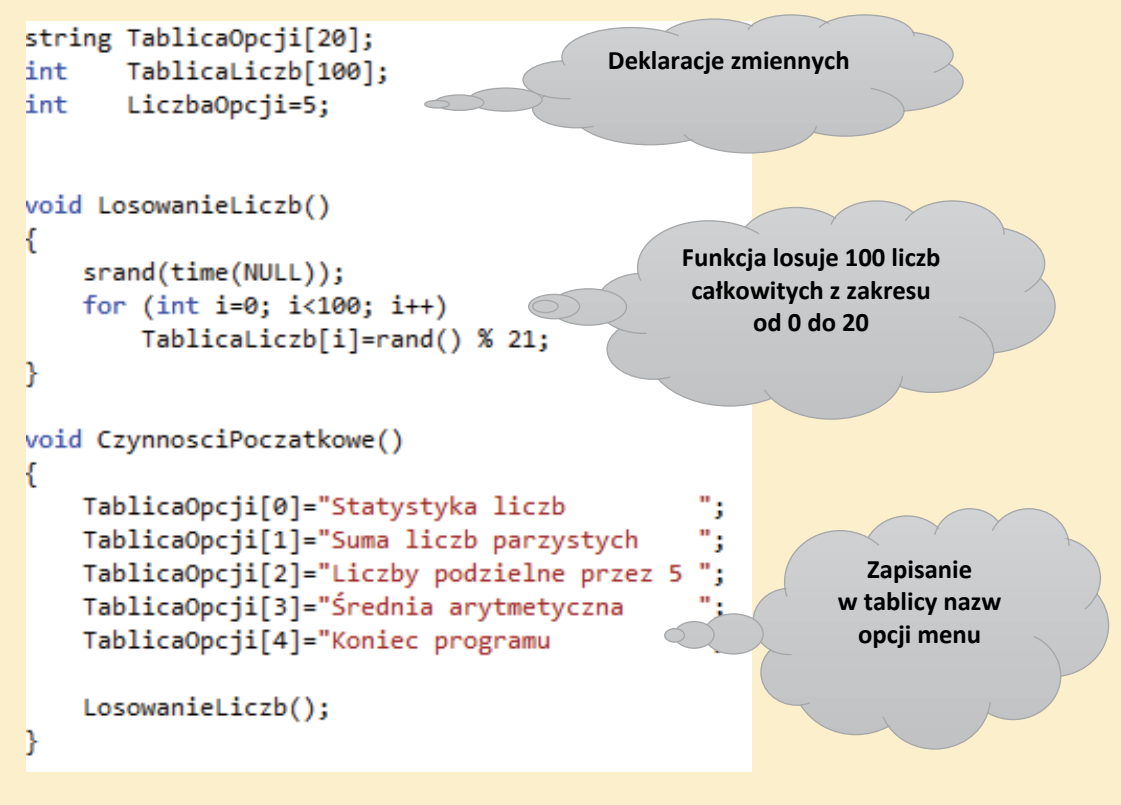
1. Prezentacja – Razem można więcej.pptx
2. Film instruktażowy 1 – Przygotowanie szkieletu programu
3. Plik skompresowany (materiały pomocnicze) zawierający folder omawianego projektu zrealizowanego w środowisku Visual Studio Express 2012 for Windows Desktop

LEKCJA NR 2

TEMAT: Połączyć i sprawdzić

Streszczenie

W ramach pierwszej lekcji scenariusza napisano szkielet programu, a zawarte w nim funkcje zostały przydzielone do wykonania grupom uczniów. W ramach tej lekcji należy przetestować różne kombinacje rozwiązań i ocenić wynik końcowy. Na rysunku 7 pokazano przykładową deklarację funkcji CzynnosciPoczatkowe().




```
string TablicaOpcji[20];
int    TablicaLiczb[100];
int    LiczbaOpcji=5;


void LosowanieLiczb()
{
    srand(time(NULL));
    for (int i=0; i<100; i++)
        TablicaLiczb[i]=rand() % 21;
}

void CzynnosciPoczatkowe()
{
    TablicaOpcji[0]="Statystyka liczb          ";
    TablicaOpcji[1]="Suma liczb parzystych     ";
    TablicaOpcji[2]="Liczby podzielne przez 5 ";
    TablicaOpcji[3]="Średnia arytmetyczna      ";
    TablicaOpcji[4]="Koniec programu

    LosowanieLiczb();
}
```

Rysunek 7. Deklaracje niezbędnych zmiennych i funkcji.

Zadeklarowane zostały dwie tablice, TablicaOpcji, która będzie przechowywała teksty opcji menu, oraz tablica TablicaLiczb, która będzie przechowywała wylosowane liczby całkowite z przedziału od 0 do 20. Pokazana na rysunku 5 funkcja CzynnosciPoczatkowe() jest najprostszym rozwiązaniem problemu, ale zdecydowanie mało elastycznym, ponieważ zmiana tekstów opcji wymaga modyfikacji funkcji. Rysunek 8 przedstawia funkcję CzynnosciPoczatkowe(), która teksty opcji odczytuje z pliku tekstowego.

```
void CzynnosciPoczatkowe()
{

    LiczbaOpcji=0;
    fstream plik;

    plik.open("Opcje.txt", ios::in);
    if(plik.good() == true)
    {
        while(!plik.eof())
        {
            getline(plik, TablicaOpcji[LiczbaOpcji++]);

        }
        plik.close();
    }
    LiczbaOpcji-=1;
 LosowanieLiczb();
}
```

Funkcja odczytuje teksty opcji i ustala wartość zmiennej LiczbaOpcji

Rysunek 8. Funkcja CzynnosciPoczatkowe() odczytująca dane z pliku tekstowego.

Zadania przydzielone uczniom do wykonania mogą zostać zrealizowane w różny sposób, a miarą poprawności jest możliwość wykorzystania przygotowanej funkcji w ramach szkieletu programu, który został przygotowany na pierwszej lekcji. W trakcie realizacji zadań mogą zostać przygotowane funkcje realizujące zadania pomocnicze. Na rysunku 9 pokazano dwie proste funkcje, które mogą pomóc w planowaniu wyświetlanego na ekranie tekstu.

```
#include<iostream>
#include<windows.h>

HANDLE uchwyt=GetStdHandle(STD_OUTPUT_HANDLE);


void PozycjaTekstu(int x, int y)
{
  COORD c;
  c.X = x - 1;
  c.Y = y - 1;
  SetConsoleCursorPosition (uchwyt, c);
}

 void KolorTekstu(WORD kolor)
 {
     SetConsoleTextAttribute(uchwyt,kolor);
 }
```

Funkcja ustalająca pozycję kursora na ekranie konsoli

Funkcja ustalająca kolor i tło dla tekstów

Rysunek 9. Funkcje do ustalania parametrów wyświetlanego tekstu.

Zaprezentowane na rysunku 9 funkcje mogą zostać wykorzystane do wyświetlenia menu na ekranie konsoli. Na rysunku 10 pokazano funkcję WyswietlMenu(), która jest jednym z możliwych sposobów realizacji tego zadania.

```
void WyswietlMenu()
{
    KolorTekstu(0x001E);
    system ("cls");
    for (int i=0;i<LiczbaOpcji;i++)
    {
        if (i==0)
        KolorTekstu(0x00E1);
        else

        KolorTekstu(0x001E);

        PozycjaTekstu(10,10+i);
        cout << TablicaOpcji[i].c_str()<<endl<<endl;
    }

    RysujRamke(5,5,60,20);
}
```

Wyświetlane są teksty zapisane w tablicy TablicaOpcji. Pierwsza opcja została wyróżniona innym kolorem

Rysunek 10. Funkcja WyswietlMenu().

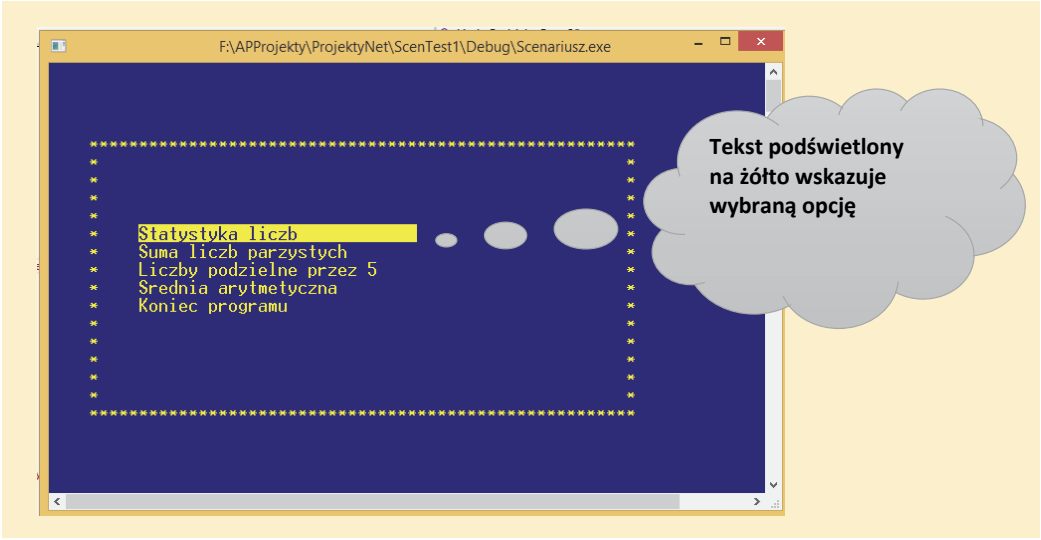
W pokazanym na rysunku 10 kodzie funkcji WyswietlMenu() wykorzystano funkcję RysujRamke(), którą można użyć przy wyświetlaniu wyników na ekranie.

```
void RysujRamke(int x1, int y1, int x2, int y2)
{
    for (int i=0; i<x2-x1;i++)
    {PozycjaTekstu(x1+i,y1);
     cout<<"*";}

    for (int i=0; i<x2-x1;i++)
    {PozycjaTekstu(x1+i,y2);
        cout<<"*";}

    for (int i=0; i<y2-y1;i++)
    {PozycjaTekstu(x1,y1+i);
        cout<<"*";}

    for (int i=0; i<y2-y1;i++)
    {PozycjaTekstu(x2-1,y1+i);
        cout<<"*";  }
}
```

Rysunek 11. Funkcja RysujRamke().

Wykonanie funkcji WyswietlMenu() wyświetla na ekranie dostępne opcje i wyróżnia kolorem wybraną opcję. Zaprezentowany na rysunku 12 wygląd ekranu po wykonaniu funkcji WyswietlMenu() jest jedną

z możliwych propozycji rozwiązania tego problemu. Uczniowie, którzy będą mieli przydzielone do wykonania zadanie napisania tej funkcji mogą zaproponować inny sposób prezentacji dostępnych opcji menu.



Rysunek 12. Wygląd ekranu po uruchomieniu funkcji WyswietlMenu().

Kolejnym elementem do rozwiązania jest obsługa menu. Na rysunku 13 pokazano funkcję ObsługaMenu(), która zmienia wyróżnioną opcję w zależności od wciśniętych klawiszy. Naciśnięcie klawisza Enter kończy działanie funkcji i zwraca liczbę całkowitą reprezentującą wybraną opcję.

```
int ObslugaMenu()
{
    int WybranaOpcja=1, znak;
    znak=getch();
    while  (znak!=13)
    {
           PozycjaTekstu(10,10+ WybranaOpcja -1);
           KolorTekstu(0x001E);
           cout << TablicaOpcji[WybranaOpcja -1].c_str()<<endl<<endl;
            if (znak==224)
            {
                  if( kbhit() )
                 switch( getch() )
                  {
                     case 80 :
                         if (WybranaOpcja==LiczbaOpcji)
                             WybranaOpcja=1;
                         else
                             WybranaOpcja++;
                         break;
                     case 72 :
                         if (WybranaOpcja==1)
                             WybranaOpcja=LiczbaOpcji;
                         else
                             WybranaOpcja--;
                         break;
                  }
                  PozycjaTekstu(10,10+ WybranaOpcja -1);
                  KolorTekstu(0x00E1);
                  cout << TablicaOpcji[WybranaOpcja -1].c_str()<<endl<<endl;
            };
            znak=getch();
    };

    return WybranaOpcja;
}
```

Zmiana wyróżnionej kolorem opcji menu po naciśnięciu klawisza

Rysunek 13. Funkcja ObslugaMenu().

Po wybraniu opcji wywołana zostanie, pokazana na rysunku 5, funkcja WykonajZadanie(). Funkcje wykonujące poszczególne zadania wykonują operacje na tablicy liczb całkowitych (zmienna TablicaLiczb) oraz pokazują na ekranie wynik swoich obliczeń. Na rysunku 14 zaprezentowano przykład funkcji o nazwie LiczbyPodzielnePrzez5().

```
void LiczbyPodzielnePrzez5()
{
    int ileP=0;
    for (int i=0; i<100; i++)
        if (TablicaLiczb[i] % 5 ==0)
            ileP++;
    KolorTekstu(0x001E);
  system ("cls");
    PozycjaTekstu(15,15);

    cout<<"W tablicy jest "<<ileP<<" liczb podzielnych przez 5 "<<endl;
    getch();
}
```

Rysunek 14. Funkcja LiczbyPodzielnePrzez5().

Pełny projekt przykładowego rozwiązania omawianego zadania jest załączony do materiałów scenariusza (materiały pomocnicze).

Podstawa programowa

Etap edukacyjny: IV, przedmiot: informatyka (poziom rozszerzony)

Cele kształcenia – wymagania ogólne

III. Rozwiązywanie problemów i podejmowanie decyzji z wykorzystaniem komputera, stosowanie podejścia algorytmicznego.

Treści nauczania – wymagania szczegółowe

Uczeń:

1) analizuje, modeluje i rozwiązuje sytuacje problemowe z różnych dziedzin;
2) stosuje podejście algorytmiczne do rozwiązywania problemu;
3) formułuje przykłady sytuacji problemowych, których rozwiązanie wymaga podejścia algorytmicznego i użycia komputera;
4) dobiera efektywny algorytm do rozwiązania sytuacji problemowej i zapisuje go w wybranej notacji;
5) posługuje się podstawowymi technikami algorytmicznymi;
6) ocenia własności rozwiązania algorytmicznego (komputerowego), np. zgodność ze specyfikacją, efektywność działania;
7) opracowuje i przeprowadza wszystkie etapy prowadzące do otrzymania poprawnego rozwiązania problemu: od sformułowania specyfikacji problemu po testowanie rozwiązania;
13) stosuje metodę zstępującą i wstępującą przy rozwiązywaniu problemu;
28) realizuje indywidualnie lub zespołowo projekt programistyczny z wydzieleniem jego modułów, w ramach pracy zespołowej, dokumentuje pracę zespołu.

Cel

Podstawowym celem lekcji jest kontynuacja zadania, którego przygotowanie rozpoczęto w ramach poprzedniej lekcji scenariusza. Głównym celem jest sprawdzenie działania programu wykorzystując deklaracje funkcji przygotowane przez różne zespoły uczniów.

Słowa kluczowe

funkcja, parametry funkcji, biblioteka funkcji

Co przygotować

- W ramach lekcji wykorzystywany jest szkielet programu, który został przygotowany w trakcie lekcji 1 oraz deklaracje funkcji przygotowane przez zespoły uczniów.

Przebieg zajęć

Wprowadzenie (10 minut)

W trakcie wprowadzenia należy przypomnieć i pokazać program, który wykonano w ramach lekcji 1. Omówić należy istotę zadania, czyli połączenie elementów wykonanych przez zespoły uczniów oraz testowanie rozwiązań.

Zasadnicza część lekcji (30 minut)

Wspólnie z uczniami omawiamy i wykonujemy testowanie funkcji przygotowanych przez uczniów.

Dyskusja podsumowująca (10 minut)

W ramach dyskusji należy ocenić wykonany program, ocenić poszczególne funkcje przygotowane przez uczniów oraz omówić możliwości dalszych modyfikacji.

Sprawdzenie wiedzy

W celu sprawdzenia wiedzy można wykorzystać, zamieszczony w materiałach scenariusza test 2.

Ocenianie

Ocena uczniów na podstawie aktywności w trakcie lekcji oraz na podstawie wykonania przydzielonych fragmentów zadania.

Dostępne pliki

1. Test
2. Film instruktażowy 2 – Łączymy i testujemy
3. Zadanie

Człowiek - najlepsza inwestycja

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Warszawska Wyższa Szkoła Informatyki

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego