INFORMATYKA – MÓJ SPOSÓB NA POZNANIE I OPISANIE ŚWIATA

PROGRAM NAUCZANIA INFORMATYKI Z ELEMENTAMI PRZEDMIOTÓW MATEMATYCZNO-PRZYRODNICZYCH

Informatyka – poziom rozszerzony

Szyfrowanie i inne algorytmy tekstowe

Iwona i Ireneusz Bujnowscy

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: ***Szyfrowanie i inne algorytmy tekstowe***

Autor: ***Iwona i Ireneusz Bujnowscy***

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

SZYFROWANIE I INNE ALGORYTMY TEKSTOWE

INFORMATYKA – POZIOM ROZSZERZONY

OPRACOWANY W RAMACH PROJEKTU:

INFORMATYKA – MÓJ SPOSÓB NA POZNANIE I OPISANIE ŚWIATA.

PROGRAM NAUCZANIA INFORMATYKI

Z ELEMENTAMI PRZEDMIOTÓW MATEMATYCZNO-PRZYRODNICZYCH

Streszczenie

W niniejszym scenariuszu tematycznym znajdą się opisy i informacje o sposobie realizacji tematów dotyczących wybranych algorytmów szyfrowania.

Czas realizacji

3 x 45 minut

Tematy lekcji

1. Sposoby reprezentacji tekstów w komputerze i ich przetwarzanie przy pomocy języka programowania C++.
2. Szyfrowanie tekstów. Szyfry podstawieniowe – Cezara, atBash, ROT13.
3. Szyfry przestawieniowe – szyfr płotkowy, szyfr kwadrat, własne szyfry przestawieniowe.

Podstawa programowa

Etap edukacyjny IV, przedmiot: informatyka (poziom rozszerzony)

Cele kształcenia wymagania ogólne

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

Treści nauczania – wymagania szczegółowe

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

   11.d) opisuje podstawowe algorytmy i stosuje:
   - sprawdzanie, czy dany ciąg znaków tworzy palindrom, anagram,
   - porządkowanie alfabetyczne,
   - wyszukiwanie wzorca w tekście.

   11.e) opisuje podstawowe algorytmy i stosuje algorytmy kompresji i szyfrowania, np.:
   - kody znaków o zmiennej długości, np. alfabet Morse'a, kod Huffmana,
   - szyfr Cezara,
   - szyfr przestawieniowy,
   - szyfr z kluczem jawnym (RSA),
   - wykorzystanie algorytmów szyfrowania, np. w podpisie elektronicznym.

Cel

Zapoznanie uczniów z rozwojem myśli algorytmicznej w kontekście kryptografii i kryptoanalizy.

Słowa kluczowe

kryptografia, kryptoanaliza, szyfr, szyfrogram, szyfr Cezara, szyfr ROT13, atBash, szyfr Viegenere'a, szyfr płotkowy, szyfr kwadrat, szyfr Playfeira

LEKCJA NR 1

TEMAT: Sposoby reprezentacji tekstów w komputerze i ich przetwarzanie przy pomocy języka programowania C++

Co przygotować

- Prezentacja 1 „Teksty w komputerze”

MATERIAŁ TEORETYCZNY

Rozważania zacznijmy od zastanowienia się, jak można reprezentować znaki i teksty w komputerze tak, by można je było później przetwarzać w językach programowania, np. w C++.

Dokonując pewnego uproszczenia można powiedzieć, że teksty, jak każde inne dane, powinny mieć swoją reprezentację w pamięci komputera, tak by można je było potem w programach przetwarzać. Wiemy, że komputery w zasadzie potrafią przetwarzać jedynie liczby i to zapisane w systemie binarnym. W związku z tym każdy znak, każdy wyraz i każde zdanie muszą zostać zamienione na liczby. W celu ułatwienia operacji na tekstach przyjęto, że każdemu znakowi będzie odpowiadała zawsze taka sama liczba. Aby nie było wątpliwości wprowadzono tabelę, w której zawarto wszystkie potrzebne znaki i odpowiadające im liczby – kody.

Tablica kodów ASCII

Jak wiemy, w komputerze każda litera, znak interpunkcyjny czy też niewidoczny znak sterujący ma swój numer. Jest to liczba, która wskazuje pozycję w tzw. tablicy kodów ASCII. W wyniku takiego podejścia np. w języku C++ można odwoływać się do poszczególnych znaków tekstu tak, jakby były znakami i tak, jakby były liczbami. Dzięki takiej dwoistości możliwy jest matematyczny opis np. algorytmów szyfrowania.

Poniżej pokazane są kody z tablicy kodów ASCII liter współczesnego alfabetu łacińskiego (26 znakowego).

97	98	99	100	101	102	103	104	105	106	107	108	109
a	b	c	d	e	f	g	h	i	j	k	l	m

110	111	112	113	114	115	116	117	118	119	120	121	122
n	o	p	q	r	s	t	u	v	w	x	y	z

Przetwarzanie tekstów w języku C++

Większość algorytmów, programów, procedur i funkcji działa według określonego schematu: musi otrzymać (np. wczytać z klawiatury) jakieś dane, przetworzyć je, a następnie zwrócić (np. wypisać na ekranie) wynik. Taki schemat wymusza znajomość implementacji wymienionych jego etapów. Podobnie jest z tekstami, dla których poniżej opiszemy sposoby realizacji wymienionych etapów realizacji algorytmów.

Deklarowanie i nadawanie wartości zmiennym tekstowym

Chcąc posługiwać się napisami w programie warto już podczas pisania programu to przewidzieć i zadeklarować korzystanie z biblioteki <string>. Zamieszczony poniżej krótki program ilustruje sposób deklaracji zmiennej s, wczytywanie napisu poleceniem cin oraz wypisywanie na ekranie wczytanego napisu litera po literze z odstępami w postaci znaków podkreślenia po każdej literze.

Przyjrzyjmy się przykładowemu programowi napisanemu w języku C++:

```
1. #include <iostream>
2. #include <string>
3. using namespace std;
4.
5. string s;
6. int main()
7.  {
8.      cin<<s;
9.      for (int i=0;i<s.size();i++)
10.       cout << s[i] <<" _ ";
11.      cin.ignore(2);
12. }
```

Powyższy program wczytuje, zgodnie z działaniem instrukcji cin, napis jedynie do pierwszego „białego znaku", np. pierwszej spacji lub tabulatora. W przypadku, gdy tekst zawiera kilka słów należy go wczytać stosując polecenie getline. Wiersz 8 powinien zostać zastąpiony następującą instrukcją:

```
8.      getline(cin, s);
```

W programie widać, że string może być traktowany jak tablica znaków. W wierszach 9-10, w pętli for, kolejno przetwarzamy i-ty znak napisu, odwołując się do kolejnych znaków poprzez indeksy – s[i]. Należy pamiętać, że indeksy w zmiennych typu string zaczynają się od wartości 0.

Instrukcja w wierszu 11 pozwala zatrzymać ekran wynikowy, tak by nie wyłączył się zaraz po skończonym działania programu. W środowisku Code:: Blocks wiersz ten nie jest potrzebny, bowiem po zakończeniu działanie programu ekran wynikowy pozostaje widoczny.

Wykonywanie operacji łączenia tekstów, usuwania i zamiany znaków

Poza możliwością odwoływania się do kolejnych znaków, język C++ umożliwia wiele operacji na zmiennych tekstowych. Wśród tych, które znajdują największe zastosowanie można wymienić operacje łączenia tekstów, usuwania pojedynczych znaków lub fragmentów tekstu oraz zmiany znaków.

Polecenia te zilustrujemy krótkim programem w C++:

```
1. #include <iostream>
2. #include <string>
3. using namespace std;
4.
5. string s1="Algorytmy", s2="struktury danych";
6. string w;
7. int main()
8.  {
9.      w = s1 + " i " + s2;
10.     cout << w <<endl;
11.      w.erase(10,2);
```

```
12.      cout << w << endl;
13.
14.      cin.ignore(2);
15. }
```

W wierszu 5 programu znajduje się deklaracja zmiennych tekstowych wraz z nadaniem im wartości początkowych, w wierszu 9 – najprostszy sposób łączenia tekstów, natomiast w wierszu 11 zawarta jest instrukcja usunięcia z tekstu dwóch kolejnych znaków począwszy od znaku o indeksie 10.

Aby zamienić znaki, możemy stosować instrukcje zamiany elementów tablicy.

Instrukcje przedstawione poniżej zamienią litery małe na wielkie i odwrotnie:

```
1.      for (int i=0;i<s.size();i++)
2.       if (islower(s[i])) s[i]=toupper(s[i]);
3.           else s[i]=tolower(s[i]);
```

Powyższe wiersze kodu ilustrują sposób użycia funkcji testujących znaki np. islower(), isupper(), czy isalpha() oraz sposób zamiany pojedynczych znaków w tekstach.

Przebieg zajęć

Czynności organizacyjne (5 minut)

Sprawdzenie obecności, podanie tematu, określenie celów lekcji.

Wprowadzenie (10 minut)

Omówienie materiału teoretycznego przygotowanego do lekcji – prezentacja 1:

- sposoby reprezentowania znaków i tekstów w komputerze,
- sposoby wczytywania i wypisywania zmiennych tekstowych na ekranie,
- przetwarzanie tekstów znak po znaku,
- wykonywanie operacji łączenia, usuwania i zmiany znaków.

Praca zespołowa (20 minut)

Uczniowie pracując w grupach rozwiązują proste zadania (programy) ilustrujące przetwarzanie tekstów.

➠ Ćwiczenie 1.1

Wczytaj tekst i wypisz liczbę małych i wielkich liter.

➠ Ćwiczenie 1.2

Wczytaj ciąg znaków oraz liczbę *k*. Wypisz na ekranie tekst z pominięciem *k* znaków na początku i na końcu tekstu.

➠ Ćwiczenie 1.3

Wczytaj tekst i zastąp w nim znakiem '–' wszystkie samogłoski.

Dyskusja podsumowująca (5 minut)

Pisząc programy w C++ możemy wczytywać i wypisywać teksty wykorzystując strumienie: polecenia cin i cout, jak również korzystając w poleceń znanych wcześniej w języku C: scanf i printf. Sposób wczytywania nie ma wpływu na dalsze przetwarzanie tekstów, jednak w niektórych środowiskach, operacje scanf/printf są szybsze niż operacje cin/cout.

Bardzo ważna jest umiejętność przetwarzania tekstów znak po znaku oraz możliwość odwoływania się do poszczególnych znaków tak, jakby cały tekst był tablicą, a poszczególne znaki jej elementami. Dzięki temu można w łatwy sposób zaimplementować nawet bardzo złożone algorytmy tekstowe.

Omówienie z młodzieżą występujących trudności i sposobów unikania w przyszłości popełnianych błędów.

Praca domowa (5 minut)

- Ćwiczenie 1.4

Wczytaj tekst i zamień w nim wszystkie litery małe na wielkie, a wielkie litery na małe.

- Ćwiczenie 1.5

Wczytaj tekst zawierający kilka wyrazów i wypisz pozycje, pod którymi znajdują się spacje.

- Ćwiczenie 1.6

Wczytaj tekst zawierający kilka wyrazów i wypisz najdłuższy wyraz w tekście.

- Ćwiczenie 1.7

Wczytaj tekst zawierający kilka wyrazów i wypisz pierwsze i ostatnie litery każdego z nich, np. dla tekstu **Ala ma kota, a Ola ma psa**, program powinien wypisać A-a ma k-a a O-a ma p-a.

LEKCJA NR 2

TEMAT: Szyfrowanie tekstów. Szyfry podstawieniowe – Cezara, atBash, ROT13

Co przygotować

- Prezentacja 2 „Szyfrowanie ”

MATERIAŁ TEORETYCZNY

Kryptologia to nauka o szyfrowaniu informacji. Zgodnie z definicją zawartą w Wikipedii, kryptologia to dziedzina wiedzy o przekazywaniu informacji w sposób zabezpieczony przed niepowołanym dostępem. Od wielu wieków starano się przesyłać teksty w sposób uniemożliwiający ich odczyt przez osoby nieupoważnione. Warto poznać te metody, ponieważ można się przy okazji zapoznać z rozwojem ludzkiej wiedzy, zmianami sposobów myślenia i różnymi metodami rozwiązywania problemów.

Szyfrowanie, bo temu tematowi poświęcimy najwięcej czasu, polega na tym, że tekst, który chcemy ukryć, przetwarzamy algorytmem szyfrującym i zamiast pierwotnego tekstu otrzymujemy ciąg liter lub symboli nazywany szyfrogramem. Proces odwrotny do szyfrowania polega na odczycie z szyfrogramu zaszyfrowanej treści.

Najbardziej znanymi metodami są szyfry starożytne. W większości polegały one na zamianie liter w tekście na inne zgodnie z jakąś przyjętą zasadą. Mówiąc prościej: w całym szyfrowanym tekście zamieniano np. wszystkie litery a na jakąś, jedną literę; wszystkie występujące litery b na literę inną niż b itd.

Przyjmijmy dla uproszczenia, że alfabet, którym posługujemy się, składa się z 26 liter. W takim przypadku liczba wszystkich różnych przyporządkowań liter literom w alfabecie wynosi aż 26! (26 silnia). Spośród wszystkich tych możliwości przypisania, co najmniej trzy były wykorzystywane i dorobiły się nazw własnych – zostaną opisane poniżej.

Dość ciekawym sposobem szyfrowania jest zmiana kolejności liter w szyfrogramie. W szyfrogramach występują te same litery, które znajdują się w tekście szyfrowanym, jednak ich kolejność jest zmieniana zgodnie z jakąś przyjętą zasadą. Jest to grupa tzw. szyfrów przestawieniowych (permutacyjnych). Przedstawicielami tej grupy są szyfr płotkowy i szyfr macierzowy.

Do szyfrów klasycznych zalicza się również szyfr Playfair. To bardzo ważny algorytm dający całkowicie różniące się szyfrogramy nawet przy stosunkowo niewielkiej różnicy w użytym kluczu.

Kolejnym wartym poznania jest wieloalfabetowy szyfr podstawieniowy Viegenere'a. Wszystkie te szyfry zostaną omówione na kolejnych lekcjach poświęconych szyfrowaniu.

Szyfr Cezara

Omawianie szyfrów rozpoczniemy od jednego z najstarszych i, jak się wydaje, najprostszego sposobu szyfrowania. Przy pomocy tego szyfru podstawieniowego porozumiewali się Juliusz Cezar i Cyceron. To szyfr, który polegał na zastąpieniu każdej litery inną, leżącą w alfabecie o 3 pozycje w prawo.

Przyjrzyjmy się tabeli ilustrującej podstawienia:

litery tekstu	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
litery szyfrogramu	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c

W tabeli widać, że np. litera a zastępowana jest przez literę d, litera b – przez literę e itd. Litery z końca alfabetu, takie jak x, y, z zastępowane są odpowiednio przez litery a, b oraz c.

Przykład szyfrowania tekstu:

tekst	ala ma kota
szyfrogram	dod pd qrwd

Taki sposób szyfrowania jest bardzo łatwy do zaimplementowania na komputerze.

Poniższy program prawidłowo szyfruje tekst złożony z małych liter alfabetu łacińskiego.

Choć przedstawiony program szyfruje tekst składający się wyłącznie z małych liter należy wiedzieć, że szyfr może uwzględniać wielkości liter. W takim przypadku w zaprezentowanym algorytmie należy zmodyfikować wiersze 10 i 11.

```
1. #include <iostream>
2. using namespace std;
3.
4. string s;
5. int main()
6.  {
7.      cin<<s;
8.      for (int i=0;i<s.size();i++)
9.       {
10.          s[i] = s[i]+3;
11.          if (s[i] > 122) s[i] = s[i]-26;
12.          cout << s[i];
13.       }
14.      cin.ignore(2);
15. }
```

W linii 10 zmienia się i-ty znak na leżący 3 pozycje w prawo. W linii 11 rozpatrywany jest przypadek, gdy kod nowego znaku leży na prawo od litery z. W takim przypadku i-tym znakiem będzie znak będący w tabeli ASCII o 26 pozycji w lewo.

To rozwiązanie, choć skuteczne nie jest jedynym. Dokładnie taki sam efekt można uzyskać stosując (wymyślając) jakiś wzór, który zrealizuje odpowiednią zamianę.

Poniżej przedstawiono przykładowy wzór:

s[i]= ((s[i]-97+3) mod 26) + 97

Program z wykorzystanym wzorem ma postać:

```
1. #include <iostream>
2.
3. using namespace std;
4.
5. string s;
```

```
6. int main()
7.  {
8.      cin<<s;
9.      for (int i=0;i<s.size();i++)
10.       {
11.          s[i] = ((s[i]-97 + 3) % 26) + 97;
12.          cout << s[i];
13.       }
14.      cin.ignore(2);
15. }
```

Spróbuj rozwikłać ten wzór i zweryfikuj poprawność wzoru i przedstawionego programu. A może wymyślisz swój własny?

Sposób deszyfrowania jest operacją przeciwną do algorytmu szyfrującego. Podczas tej czynności po prostu za odpowiednie litery podstawić należy te, które leżą w alfabecie o 3 pozycje w lewo. Sposób zapisu algorytmu deszyfrującego jest tworzony podobnie do algorytmu szyfrującego, choć oczywiście jest nieco inny. Spróbuj napisać program, który szyfrogram otrzymany w wyniku zastosowania szyfru Cezara odszyfruje i wyświetli na ekranie komputera.

Szyfr ROT13

Szyfr ROT13 jest kolejnym z szyfrów podstawieniowych. Od szyfru Cezara różni się tylko przesunięciem. W przypadku ROT13 litera z szyfrogramu leży o 13 pozycji w prawo od litery szyfrowanej.

Przyjrzyjmy się tabeli ilustrującej podstawienia w ROT13:

litery tekstu	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
litery szyfrogramu	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m

Szyfr ROT13 ma bardzo ważną cechę, która sprawia, że szyfr ten stał się popularny. Otóż przesunięcie o 13, przy 26 znakach alfabetu sprawia, że do szyfrowania i deszyfrowania można użyć takiego samego algorytmu.

tekst-szyfr	a	b	c	d	e	f	g	h	i	j	k	l	m
szyfr tekst	n	o	p	q	r	s	t	u	v	w	x	y	z

Jest to szyfr, w którym szyfrowana litera jest jednocześnie literą otrzymaną po powtórnym zaszyfrowaniu szyfrogramu.

Szyfr ROT13 jest zmodyfikowanym szyfrem Cezara. Łatwo sobie można wyobrazić szyfry wykorzystujące inne przesunięcia niż stosowane w klasycznym szyfrze Cezara – 3, czy w ROT13 przesunięcie o 13 znaków.

Szyfr Atbash

Szyfr Atbash to ostatni z prostych szyfrów podstawieniowych, które chcemy tu przedstawić. Algorytm szyfrujący polega na wyborze liter leżących symetrycznie po przeciwnej stronie alfabetu. Tak więc litera a zostanie zastąpiona przez literę z, a litera z przez literę a. Z pozostałymi literami postępujemy podobnie.

litery tekstu	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
litery szyfrogramu	z	y	x	w	v	u	t	s	r	q	p	o	n	m	l	k	j	i	h	g	f	e	d	c	b	a

Sposób „łamania" szyfrów podstawieniowych

Opisane szyfry podstawieniowe nie zapewniają bezpieczeństwa danych. Skuteczną metodą jest sprawdzenie częstotliwości liter w szyfrogramach. Okazuje się, że w każdym kraju można wskazać, z jaką częstotliwością występują poszczególne litery. Metoda ta działa w przypadku zgromadzenia odpowiedniej liczby szyfrogramów, ale jej skuteczność jest niepodważalna nawet przy ich małej liczbie.

Aby wyszukać częstotliwość występowania liter alfabetu w języku polskim i angielskim, można skorzystać z wyszukiwarki google. W języku polskim najczęściej powtarza się litera a, natomiast w języku angielskim litera e. Na tej podstawie z łatwością można już podać algorytm łamania szyfrogramów otrzymanych metodą podstawieniową.

Przebieg zajęć

Czynności organizacyjne (5 minut)

Sprawdzenie obecności, podanie tematu, określenie celów lekcji.

Wprowadzenie (10 minut)

Omówienie materiału teoretycznego przygotowanego do lekcji – prezentacja 2

- ogólna klasyfikacja szyfrów,
- szyfry podstawieniowe i ich implementacja,
- kryptoanaliza – sposoby łamania szyfrów podstawieniowych.

Praca zespołowa (20 minut)

Uczniowie pracując w grupach rozwiązują proste zadania (programy) ilustrujące szyfrowanie tekstów.

➠ Ćwiczenie 2.1

Zaproponuj wzór do szyfru Cezara, gdy dana jest litera oraz tabela kodów ASCII.

Chodzi o to, by po podaniu litery, znając jej pozycję w tabeli kodów ASCII, obliczyć pozycję litery, która powinna znaleźć się na jej miejscu w szyfrogramie.

➠ Ćwiczenie 2.2

Wczytaj tekst i zaszyfruj go metodą ROT13.

➠ Ćwiczenie 2.3

Zaproponuj wzór do szyfru atBash, gdy dana jest litera oraz tabela kodów ASCII.

Chodzi o to, podobnie jak w ćwiczeniu 2.1, by po podaniu litery, znając jej pozycję w tabeli kodów ASCII, obliczyć pozycję litery (a więc i literę), która powinna znaleźć się w szyfrogramie.

Dyskusja podsumowująca (5 minut)

Szyfry podstawieniowe w obecnych czasach nie zapewniają bezpieczeństwa przesyłanych danych. Analiza częstościowa umożliwia w łatwy sposób przypisanie literom z szyfrogramu właściwe, pierwotne litery. Pomysłem, choć też jak dziś wiemy, mało skutecznym, jest kodowanie każdej litery innym kluczem. Szyfrem podstawieniowym, gdzie każdą literę szyfruje się z różnym przesunięciem jest szyfr Vigenere'a.

Omówienie z uczniami występujących trudności i sposobów unikania w przyszłości popełnianych błędów. W celu sprawdzenia poprawności rozwiązań należy testować je dla różnych danych. W przypadku szyfrowania warto ocenić zachowanie programu dla wartości brzegowych, zarówno przy szyfrowaniu, jak i deszyfrowaniu. Najlepszym sposobem pozwalającym na pełną ocenę poprawności jest zaszyfrowanie i zdeszyfrowanie pełnego alfabetu.

Praca domowa (5 minut)

Ćwiczenie 2.4

Zaszyfruj jakiś tekst metodą atBash.

Ćwiczenie 2.5

Zaszyfruj tekst podstawiając zamiast parzystych liter w tekście, litery leżące o 3 pozycje dalej w tabeli kodów ASCII, a zamiast liter z pozycji nieparzystych litery leżące o 3 pozycje bliżej. W algorytmie uwzględnij zawijanie się alfabetu tak, jak w zwykłym szyfrowaniu Cezara.

Ćwiczenie 2.6

Zaproponuj swój własny algorytm szyfrowania metodą podstawieniową. Opracuj algorytm, ewentualny wzór i napisz program, który go realizuje.

Ćwiczenie 2.7

Znajdź w Internecie informacje dotyczące szyfru Viegenere'a i zaszyfruj nim tekst INFORMATYKA mając za klucz tekst LITWOOJCZYZNOTYMOJA i traktując kody liter klucza jako wartość przesunięcia. Zwróć uwagę na to, że każda litera w tekście szyfrowana jest podobnie jak w szyfrze Cezara, z tym, że litera w szyfrogramie jest uzyskiwana poprzez przesunięcie litery z testu o liczbę znaków wynikającą z wartości odpowiedniej litery klucza.

LEKCJA NR 3

TEMAT: Szyfry przestawieniowe – szyfr płotkowy, szyfr kwadrat, własne szyfry przestawieniowe

Co przygotować

- Prezentacja 3 „Szyfrowanie ”

MATERIAŁ TEORETYCZNY

Szyfry przestawieniowe, w odróżnieniu do szyfrów podstawieniowych, tworzą szyfrogram z liter, które występują w pierwotnym tekście, zmienia się jedynie ich kolejność. Do najbardziej znanych szyfrów przestawieniowych można zaliczyć szyfr kwadrat (macierz) oraz szyfr płotkowy.

Szyfr kwadrat

Algorytm szyfrowania składa się z dwóch etapów. Po wyznaczeniu boku minimalnego kwadratu, w który można wpisać szyfrowany tekst należy wpisać go wierszami do kwadratu, a niewypełnione pola uzupełnić np. spacjami. Szyfrogram otrzymuje się wczytując litery w kolejnych kolumnach. Na przykład tekst WIEMZENICNIEWIEM byłby zaszyfrowany przy pomocy następującej tabeli:

W	I	E	M
Z	E	N	I
C	N	I	E
W	I	E	M

Uzyskany szyfrogram to: WZCWIENIENIEMIEM.

```
1. #include <iostream>
2.
3. using namespace std;
4. int n,w;
5. string s;
6. char t[10][10];
7. int main()
8.  {
9.    cin<<s;
10.   n=s.size();
11.   int m=0;
12.   while (m*m<n) m++; //bok kwadratu
13.
```

```
14.      for (w=0;w<m;w++)
15.           for (int k=0;k<m;k++)
16.              t[k][w]=s[w*m+k];
17.
18.  for (int k=0;k<m;k++)
19.     for (w=0;w<m;w++)
20.              cout<<t[k][w];
21.
22.      cin.ignore(2);
23. }
```

w						w	
	a				a		a
		r		z			
			s				

Powyższy program realizuje szyfrowanie tekstu szyfrem KWADRAT (plik kwadrat - materiały pomocnicze 1).

W linii 12 obliczany jest bok najmniejszego kwadratu, do którego w całości można wpisać szyfrowany tekst. W wierszach 14-16 kwadrat wypełniany jest kolejnymi literami tekstu. Wiersze 18-20 to wypisanie zawartości kolejnych kolumn kwadratu.

Szyfr płotkowy

Szyfr płotkowy jest szyfrem, który wymaga podania klucza. W zależności od klucza zmienia się wysokość płotu. Najlepiej widać to na przykładach:

warszawa -> wwaaarzs (wysokość płotu 4)

w				z			
	a		s		a		a
		r				w	

warszawa -> wzasaarw (wysokość płotu 3)

w		r		z		w	
	a		s		a		a

warszawa -> wrzwasaa (wysokość płotu 2)

Poniżej prezentujemy przykładowy kod obliczający szyfrogram szyfru płotkowego po podaniu tekstu oraz wysokości płotka (plik plot - materiały pomocnicze 2):

```
1. #include <iostream>
2. #include <string>
3. using namespace std;
4.
5. string s;
6. string wynik;
7. int wys,poz;
8. int a,b;
9. int main()
10. {
11.   cin>> s;
12.   cin >> wys;
13.
14.   for (int w=0; w<wys; w++)
15.    {
16.      cout <<s[w];
17.      poz=w;
18.      a=2*(wys-1)-2*w;
19.      b=2*w;
20.      while ((poz+a)<s.size() || (poz+b)<s.size())
21.        {
22.         if (a>0 && (poz+a)<s.size()) cout << s[poz+a];
23.         if (a>0) poz+=a;
24.         if (b>0 && (poz+b)<s.size()) cout << s[poz+b];
25.         if (b>0) poz+=b;
26.        }
27.    }
28. }
```

Algorytm płotkowy, jak widać, nie jest skomplikowany w zapisie, chociaż wymaga chwili refleksji. Otóż z przykładów wynika, że do szyfrogramu pobierane są litery, których odległość od siebie jest zmienna. Autor programu wykorzystał fakt istnienia w każdym wierszu dwóch odległości – oznaczonych zmiennymi *a* i *b*, z których odległość *a* maleje, a *b* rośnie.

Szyfr Playfaira

Ciekawym szyfrem przestawieniowym, który warto pokazać uczniom jest szyfr Playfaira.

Do szyfrowania i deszyfrowania wykorzystuje się tablicę 5 x 5 wypełnioną literami alfabetu łacińskiego (26 znakowego). Aby zmieścić 26 liter w tablicy 25-elementowej przyjęto, że dwie litery i oraz j umieszcza się w jednej komórce.

Poniżej przedstawiono przykładową tabelę utworzoną dla klucza MUZYK. Tabela została utworzona w ten sposób, że najpierw umieszczono w niej litery klucza, a następnie pozostałe litery alfabetu, czyli te, które nie występowały w kluczu. W przypadku, gdyby w kluczu którekolwiek litery się powtarzały,

w tabeli umieszcza się tylko ich pierwsze wystąpienia. Na przykład gdyby kluczem było słowo ABRAKADABRA, to w tabeli umieszczone zostałoby słowo ABRKD.

M	*U*	*Z*	*Y*	*K*
A	*B*	*C*	*D*	*E*
F	*G*	*H*	*I/J*	*L*
N	*O*	*P*	*Q*	*R*
S	*T*	*V*	*W*	*X*

Zastosowanie szyfru Playfeira wymaga podzielenia słowa na pary liter. Słowo INFORMATYK byłoby więc szyfrowane z wykorzystaniem par IN-FO-RM-AT-YK.

Gdyby w szyfrowanym tekście liczba liter byłaby nieparzysta, to można np. uzupełnić tekst jakąś literą rzadko występującą w alfabecie np. literą X.

Szyfrujemy parami:

1.jeśli para jest po przekątnej, to bierzemy parę z drugiej przekątnej

2.jeśli para jest w kolumnie, to bierzemy parę pod nią, z zawinięciem kolumny

3.jeśli para jest w wierszu, to bierzemy parę po prawej, z zawinięciem wiersza

Stosując podaną metodę, szyfrogramem słowa INFORMATYK jest słowo FQGNKNBSKM.

Przebieg zajęć

Czynności organizacyjne (5 minut)

Sprawdzenie obecności, podanie tematu, określenie celów lekcji.

Wprowadzenie (10 minut)

Omówienie materiału teoretycznego przygotowanego do lekcji – prezentacja 2

- szyfry przestawieniowe,
- czym są, jaka jest istota ich tworzenia,
- szyfr kwadrat, zasada szyfrowania tekstu,
- szyfr płotkowy, klucz szyfru płotkowego, przykłady szyfrowania tego samego tekstu z różnym kluczem ,
- przykład szyfru „wspak".

Praca zespołowa (20 minut)

Uczniowie pracując w grupach rozwiązują proste zadania (programy) ilustrujące szyfrowanie tekstów.

➠ Ćwiczenie 3.1

Zaszyfruj tekst INFORMATYKA szyfrem płotkowym o kluczu 2 i 3.

Porównaj uzyskane szyfrogramy.

➠ Ćwiczenie 3.2

Napisz program, który wczytany tekst szyfruje zgodnie z algorytmem szyfru kwadrat.

Dyskusja podsumowująca (5 minut)

Podczas pisania programu zawierającego rozwiązanie dowolnego problemu należy bardzo dokładnie go przetestować. Testowanie programu powinno polegać na sprawdzeniu, czy dla różnych danych program zwraca zawsze poprawne wyniki, czy nie ma sytuacji, że program zawiesi się, nie zakończy się prawidłowo. W przypadkach bardziej złożonych problemów bierze się pod uwagę również szybkość działania podanych programów, szacując ich złożoność.

Szyfry przedstawione na tej lekcji nie zapewniają bezpieczeństwa danych. Specjaliści od kryptoanalizy dość łatwo radzą sobie z odczytaniem wiadomości zaszyfrowanych szyframi przestawieniowymi. Dzisiejsze szyfry są bardzo złożone, opierają się na teorii liczb, często wykorzystują fakt, że niektóre problemy matematyczne nie dają się szybko rozwiązać nawet z wykorzystaniem super szybkich komputerów.

Praca domowa (5 minut)

➠ Ćwiczenie 3.4

Zaszyfruj tekst wypisując w szyfrogramie kolejne wyrazy wspak. Na przykład zamiast tekstu Litwo Ojczyzno ty moja powinien zostać wypisany tekst owtiL onzyzcjO yt ajom

➠ Ćwiczenie 3.5

Zaproponuj swój własny algorytm szyfrowania metodą przestawieniową. Opracuj algorytm, ewentualny wzór i napisz program, który go realizuje.

Sprawdzanie wiedzy

Sprawdzenie wiedzy na podstawie testu.

Ocenianie

Ocena pracy na podstawie rozwiązanych ćwiczeń.

Dostępne pliki

1. Prezentacje
2. Zadania
3. Materiały pomocnicze
4. Test
5. Film

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