Temat: Pliki i operacje na plikach- ifstream, ofstream, frstream.

Plik można otworzyć do odczytu IFSTREAM, do zapisu OFSTREAM lub do zapisu lub odczytu FSTREAM. Dane do pliku wprowadza się funkcją WRITE i strumieniem <<. Dane odczytuje się z pliku funkcją READ i strumieniem >>. Funkcja logiczna EOF pozwala sprawdzić czy ciągnięto już koniec pliku. Kolejny zapis do tego samego pliku powoduje usunięcie poprzednich danych!

Sprawy z plikami zaczynają komplikować się w zależności od tego, jak chcemy potraktować nasz plik: czy jako zbiór liczb, tekstów, znaków i innych danych. Kolejną komplikacją jest możliwość obsługi plików kilkoma sposobami. Na lekcjach będziemy zajmować się wyłącznie zapisem i odczytem za pomocą strumieni, podobnie zresztą jak robiliśmy to w konsoli.

Zapis do pliku (string): napisy, liczby, konsola
Do obsługi konsoli służyła biblioteka *iostream*, do obsługi strumieni plikowych należy zadeklarować bibliotekę ***fstream***.

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

int main() {
  setlocale(LC_ALL, "");

  ofstream ZAPIS("dane.txt");
  int l=128;
  string t="Ala ma kota";
  ZAPIS << "piszemy do pliku" << endl;
  ZAPIS << l << endl;
  ZAPIS << t << endl;
  string tekst;
  cout << "wpisz tekst:";
  cin >> tekst;
  ZAPIS << tekst<< endl;
  ZAPIS.close();
  return 0;
}
```

ofstream - strumień o nazwie ZAPIS kojarzymy z plikiem ”dane.txt” na dysku w katalogu bieżącym. Możemy zapisywać liczby i teksty, identycznie jak na ekranie konsoli za pomocą polecenia COUT. Bezwzględnie należy pamiętać o zamknięciu strumienia plikowego za pomocą polecenia **close**.

W pliku tekstowym "dane.txt" pojawią się 4 wiersze tekstu. Problemem mogą być polskie znaki diakrytyczne, zapisywane przez Windows w kodzie OEM 852.

Podczas wczytywania tekstów z konsoli posługujemy się typem **string**, który reaguje standardowo na spację i enter. Typu char należy używać raczej do wczytywania pojedynczych znaków.

Zapis do pliku (char): klawiatura-plik tekstowy
Do wczytywania znaków używamy funkcji **getch()** z biblioteki **conio.h**. Program będzie wczytywał znaki z klawiatury, zapisywał je do pliku dyskowego. Pętla zapisu kończy się w momencie naciśnięci klawisza ESC (kod 27).

```
ofstream klaw("klaw.txt");
char c;
do{
  c=getch();
  if (c==13) {
    cout << endl;
    klaw << endl;
  }
  if (c!=27){
    cout << c;
    klaw << c;
  }
} while (c!=27);
klaw.close();
```

Zwróć uwagę, w jaki sposób program obsługuje klawisz ENTER - sami musimy zadbać o przejście do nowego wiersza i klawisz ECK - nie chcemy aby program wstawiał „dziwny” znaczek na końcu.

Zapis do pliku: liczby i kolumny
Pisanie liczb do pliku wygląda tak samo, jak pisaliśmy je na ekranie w konsoli.

```
#include <stdlib.h>
...
ofstream Pliczby("liczby.txt");
int liczba;
for (int i=0;i<10;i++){
  liczba=rand() % 6 + 1;
  Pliczby << liczba << endl;
  cout << liczba << endl;
}
Pliczby.close();
```

Zapis w kolumnach - tabliczka mnożenia

```
ofstream plik("mnoz.txt");
int mnoz;
for (int i=1;i<=10;i++){
  for (int j=1;j<=10;j++){
    mnoz=i*j;
    plik.width(4);
    cout.width(4);
    plik << mnoz;
    cout << mnoz;
  }
  plik << endl;
  cout << endl;
}
plik.close();
```

Odczyt z pliku
Jeżeli znamy dokładnie ilość danych do odczytania, możemy np. korzystać z pętli FOR. W wielu przypadkach jednak nie znamy długości pliku i posługujemy się funkcję **EOF()** i pętlą while, co można przeczytać: „**dopóki nie osiągnęliśmy końca pliku …**”

```
ifstream PLIK("plik.txt");
while (!PLIK.eof()){
        ...
}
PLIK.close();
```

Odczyt z pliku znak po znaku
Podczas czytania danych z plików pojawiają się problemy podobne do tych podczas zapisywania znaków – pomijane są spacja i enter (zarówno dla typu string i char).

```
ifstream ODCZYT("dane.txt");
char z;
while (!ODCZYT.eof()){
  ODCZYT >> z;
  cout << z;
}
ODCZYT.close();
```

Przedstawiony fragment programu czyta pojedyncze znaki z pliku ”dane.txt” i wyświetla na ekranie. bez spacji i przejść do nowego wiersza.

Odczyt z pliku: czytanie wierszami
Instrukcja **getline** odczytuje całe wiersze, łączenie ze spacjami. Jeśli odczytujemy w ten sposób liczby i teksty, konieczne będzie wyodrębnianie z tekstu fragmentów i konwersja.

```
ifstream Wodczyt("dane.txt");
string wiersz;
while (!Wodczyt.eof()){
  getline(Wodczyt,wiersz);
  cout << wiersz << endl;
}
Wodczyt.close();
```

Odczyt z pliku: liczby na ekran i liczby do tablicy
Liczby czytamy dokładnie tak samo jak teksty. W pliku mogą być rozdzielone spacjami lub każda w nowym wierszu - potraktowane zostaną identycznie.

```
ifstream ODCZYT("liczby.txt");
int li;
while (!ODCZYT.eof()){
  ODCZYT >> li;
  cout << li;
}
ODCZYT.close();
```

```
int tabl[20];
int ile=0;
ifstream oplik("liczby.txt");
if(oplik.is_open()){
  while(!oplik.eof()){
    oplik >> tabl[ile];
    ile++;
  }
}
else cout<<"Brak pliku";
oplik.close();

for (int i=0; i < ile; i++)
  cout << tabl[i];
```

Warto zwrócić uwagę na sprawdzenie poprawności otwarcia pliku. Jeśli takiego pliku nie ma - funkcja is_open(), to możemy pominąć wykonywanie programu. Problemy mogą pojawić się, jeśli plik na końcu zawiera pusty wiersz - do tablicy zostanie wczytany jeden więcej element - zero!

Dopisywanie do pliku

Możemy dopisywać na końcu pliku nowe dane

```
fstream dopis;
dopis.open("dane.txt", ios::app);
dopis << "Tekst dopisany na końcu";
dopis.close();
```

Zadania

PALINDROMY

Palindromem nazywamy słowo, które czytane od lewej i od prawej strony jest takie samo. Na przykład palindromami są słowa: JABFDFBAJ, HAJAHAJAH, ABBA, Słowo JANA nie jest palindromem.

Wygeneruj plik tekstowy palindrom.txt, który zawierał będzie 1000 słów o długościach od 2 do 25 znaków, każde w nowym wierszu, składających się z wielkich liter A, B, C, D, E, F, G, H, I, J.

```
HEAJEIICEFFBHBBFCHG
ECDCCBGIFHGBIJCHJ
EDBCDDEBBDIHECHHJDBJI
FACIGA
EIGFAJAAGBDIJDEE
AGGBIEJGDHIICJ
DFJIEAH
DGBF
CAJHDHCGABGFHFEBCAABEG
HBHHHHDDFJJIBICGGADIABCF
JEHIDFBCABGE
```

Aby plik zawierał jakieś „ciekawe” palindromy zmodyfikuj program w następujący sposób:
- co 30 generowany wyraz sprawdź, czy jest krótszy niż 13 znaków

- jeśli tak, to doklej do niego ten sam odwrócony wyraz, np. DCEAB i doklejamy BAECD

1) Plik tekstowy palindrom.txt zawiera słowa wygenerowane przez komputer (maksymalnie 1000). Odczytaj je i sprawdź, które są palindromami, Wynik na ekranie i w pliku tekstowym.

```
//zadanie 1 - PALINDROM
// Tworzenie pliku
cout << "PALINDROMY" << endl;
ofstream PALzapisz("palindrom.txt");
for (int i=1; i <= 1000; i++){
        //ile znaków w wyrazie
        int ile=rand() % 24 + 2;
        string pal="";
        //sklejamy wyrazy z ile znaków
        for (int j=1; j <= ile; j++){
                //losowe znaki o kodach 65-74
                char zna=char(rand() % 10 +65);
                pal=pal+zna;
        }
        //co 30 wyraz i jeśli krótszy niż 1 znaków
        //robimy z wyrazu palindrom
        int dl=pal.length();
        if (i%30 == 0 && dl <= 12){
                for (int k=0; k < dl; k++)
                        pal=pal+pal[dl-k-1];
        }
        //zapisujemy wyrazy w pliku
        PALzapisz << pal << endl;
        cout << pal << endl;
}
PALzapisz.close();
```

DWÓJKOWE

Wygeneruj plik tekstowy napisy.txt, zawierający 1000 napisów od 2 do 16 znaków, każdy w osobnym wierszu, składających się ze znaków ‘0’ i ‘1’.

```
11000000
10101
0101111
11011
01001
011110111010
01000110
010001100
000101
101
11010001100
```

5) W pliku napisy.txt znajduje się 1000 napisów o długościach od 2 do 16 znaków, każdy napis w osobnym wierszu. W każdym napisie mogą wystąpić jedynie dwa znaki: „0” lub „1”. Odpowiedz na poniższe pytania: ekran i plik

a) ile jest napisów o parzystej długości

b) ile jest napisów, które zawierają taką samą liczbę zer i jedynek

c) ile jest napisów składających się z samych zer, z samych jedynek

d) dla każdej liczby k = 2, 3, ...16 podaj liczbę napisów o długości k znajdujących się w pliku, tzn. ile jest napisów 2-znakowych, 3-znakowych itd.

```
//napisy dwójkowe 2..16 znaków 01
ofstream NDzapis("napisy.txt");
int ile;
int znak;
for (int j=1;j<=1000;j++){
        ile=rand() %15 +2; //2-16
        string napis="";
        for (int i=1;i<=ile;i++){
                znak=rand() % 2;//0 lub 1 losowo
                //znak ASCII '0' lub '1'
                napis=napis+char(znak+48);
        }
        NDzapis << napis << endl;
}
NDzapis.close();
```

UWAGA:

Zwróć uwagę na fragmenty kodu programu umożliwiające generowanie losowe liczb całkowitych.

W celu użycia odpowiedniej funkcji rand(), musimy zadeklarować użycie modułu cstdlib

```
#include <cstdlib>
```

Funkcja generuje losowo liczby całkowite, jeżeli chcemy ograniczyć zakres to używamy konstrukcji:

```
rand() % ile_liczb_w_przedziale  + startowa_liczba;
```

Najpierw ustalamy ile liczb mieści się w przedziale z którego chcemy losować, np. będzie to 50 liczb. Następnie ustalamy pierwszą losowaną liczbę - załóżmy 7.

```
rand() % 50 ) + 7;
```

Generuje liczby z przedziału od 7 do 56