# Lekcja 9
# Pliki

*klawiatura-plik – liczby-plik – kolumny – odczyt*

Dane do programów wprowadzaliśmy do tej pory z klawiatury (lub generowaliśmy liczby losowe). Wyniki obliczeń wypisywane były na ekranie komputera. Duże ilości danych trudno jednak wpisywać ręcznie i w większości przypadków pobieramy je z plików zapisanych na dysku. Podobnie jest z wynikami obliczeń – można je zapisać w postaci pliku i przeglądać w dowolnym momencie na ekranie komputera.

Plik na dysku można porównać do taśmy magnetofonowej, na której wykonujemy trzy operacje: zapis taśmy, odczyt taśmy, przesunięcie taśmy o jeden element dalej (albo na początek i na koniec).

Dane do plików zapisujemy sekwencyjnie, to znaczy, że muszą być zapisywane po kolei. Aby zapisać element setny należy najpierw zapisać 99 elementów. Podobnie jest z odczytywaniem. Plik należy otworzyć (do zapisu lub odczytu) OPEN i po zakończeniu działań koniecznie go zamknąć CLOSE. Plik można otworzyć do odczytu ISTREAM, 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.**

**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**, tóry reaguje standardowo na spację i enter. Typu char należy używać raczej do wczytywania pojedynczych znaków.

```
#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;
}
```

```
piszemy do pliku
128
Ala ma kota
cześć
```

**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).

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.

```
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();
```

## Zapis do pliku: liczby i kolumny

Pisanie liczb do pliku wygląda tak samo, jak pisaliśmy je na ekranie w konsoli.

```
6
6
5
5
6
5
1
1
5
3
```

```
#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();
```

```
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();
```

```
   1   2   3   4   5   6   7   8   9  10
   2   4   6   8  10  12  14  16  18  20
   3   6   9  12  15  18  21  24  27  30
   4   8  12  16  20  24  28  32  36  40
   5  10  15  20  25  30  35  40  45  50
   6  12  18  24  30  36  42  48  54  60
   7  14  21  28  35  42  49  56  63  70
   8  16  24  32  40  48  56  64  72  80
   9  18  27  36  45  54  63  72  81  90
  10  20  30  40  50  60  70  80  90 100
```

## 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).

Przedstawiony obok fragment programu czyta pojedyncze znaki z pliku ”dane.txt” i wyświetla na ekranie.

```
piszemydopliku128Alamakotacześćć
```

bez spacji i przejść do nowego wiersza.

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

## Odczyt z pliku: czytanie wierszami

Funkcja **getline** odczytuje całe wiersze, łączenie ze spacjami. Jeśli odczytujemy w ten sposób liczby i teksty, konieczne będzie wyodrębnianie z tekstu fragmentów i konwersja.

```
piszemy do pliku
128
Ala ma kota
cześć
```

```
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.

```
6655651153
```

```
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!

```
66556511530
```

## 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**

*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.*

*01) 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.*
*a) aby plik zawierał jakieś „ciekawa" 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.*

**********

*02) Wygeneruj plik tekstowy **hasla.txt**, zawierający 200 słów, składających się z małych liter alfabetu angielskiego, każde w osobnym wierszy, których długość wynosi od 3 do 10 znaków.*

*2) Plik **hasla.txt** zawiera hasła używane w pewnej firmie (200). Odpowiedz na poniższe pytania: ekran i plik*

*a) ile jest haseł z parzystą - nieparzystą liczbą znaków*
*b) wypisz hasła będące palindromami - czytane wspak dadzą taki sam rezultat (kajak)*
*c) wypisz hasła, w których występuje obok siebie dwa identyczne znaki (kajjak)*

**********

*03) Wygeneruj plik tekstowy **ciagi.txt**, zawierający 1000 słów składających się z trzech liter A, B, C, każde słowo w osobnym wierszu, litery mogą się powtarzać.*

*3) W pliku **ciagi.txt** znajduje się 1000 wierszy, a w każdym wierszu ciąg o długości trzech liter ze zbioru {A, B, C} -litery mogą się powtarzać. Napisz program, który:*

*a) obliczy prawdopodobieństwo, że w wierszu będzie ciąg składającym się z takich samych znaków*
*b) obliczy prawdopodobieństwo, że w wybranym wierszu będzie palindrom*
*Prawdopodobieństwo: $P(A)=|A| / |\Omega|$, gdzie $|A|$ - ilość wierszy spełniających kryteria zadania, $|\Omega|$ - wszystkie wiersze.*

**********

*04) Wygeneruj plik tekstowy **cyfry.txt**, zawierający 1000 liczb naturalnych, mniejszych niż 10000, każda w osobnym wierszu.*

*4) Plik **cyfry.txt** zawiera liczby naturalne (maksymalnie 1000). Odpowiedz na poniższe pytania: ekran i plik*

*a) wylicz ich średnią*
*b) ile jest parzystych i nieparzystych liczb*
*c) podaj liczbę, której suma cyfr jest największa*
*d) która liczba powtarza się najczęściej*

**********

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

*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.*
*e) potraktuj każdy napis, jak liczbę dwójkową, zamień ją na dziesiętny odpowiednik i wykonaj polecenia z zadania 4*