



## Wykład: 9

### łańcuchy znaków

## Rodzaje plików

---

Dane przechowywane w pliku mogą mieć reprezentację binarną (taką samą, jak w pamięci komputera) lub tekstową (taką, jaka używana jest do wprowadzania informacji z klawiatury i wyprowadzania jej na ekran monitora lub drukarkę).

Reprezentacjom tym odpowiadają

- ✓ **elementowe** (inaczej nazywane zdefiniowanymi lub binarnymi) – mogą przechowywać dane dowolnego typu. Ich interpretacja zależy od programu, który je odczytuje. Wszystkie dane przechowywane w plikach elementowych muszą być tego samego typu.
- ✓ **tekstowe** – przechowują tekst w zapisany w kodzie ASCII

## Rodzaje plików

---

Zawartość plików elementowych jest na ogół nieczytelna dla użytkownika. Treść pliku tekstowego daje można odczytać w każdym programie obsługującym kod ASCII.

Pliki tekstowe mogą być użyte do przechowywania mieszanych typów danych (np. tekstów i liczb), gdyż wszelka informacja przechowywana jest w nich w postaci kolejnych linii z zawartością.

**Pliki elementowe są plikami o dostępie swobodnym**, to znaczy, że w każdym momencie można odwołać się do dowolnego elementu pliku.

**Pliki tekstowe są plikami o dostępie sekwencyjnym**, co oznacza, że aby dostać się do wybranego elementu pliku, należy przeczytać wszystkie elementy znajdujące się przed nim.



## Pliki tekstowe



Operacja na pliku przebiega w czterech etapach:

1. Przypisanie zmiennej plikowej do pliku na dysku
2. Otwarcie lub utworzenie i otwarcie pliku powiązanego przez zmienną.
3. Operacje na danych (zapis lub odczyt)
4. Zamknięcie pliku



Funkcje niezbędne do obsługi plików znajdują się w bibliotece:

```
#include <fstream>
```

Aby móc używać pliku deklaruje się tzw. zmienną plikową, w której przechowywany będzie uchwyt do pliku.

```
fstream zmienna_plikowa;
```

Klasa **fstream** (umieszczona w przestrzeni nazw `std::`) udostępnia cały interfejs, dzięki któremu można obsłużyć dowolny plik znajdujący się na dysku lub innym nośniku danych.



Tryb	Opis trybu
<b>ios::app</b>	<b>append - dopisywanie danych do pliku</b> - ustawia wewnętrzny wskaźnik zapisu pliku na jego koniec. Plik otwarty w trybie tylko do zapisu. Dane mogą być zapisywane tylko i wyłącznie na końcu pliku.
<b>ios::in</b>	<b>input - wejście/odczyt</b> - zezwolenie na odczytywanie danych z pliku.
<b>ios::out</b>	<b>output - wyjście/zapis</b> - zezwolenie na zapisywanie danych do pliku.
<b>ios::trunc</b>	<b>truncate</b> - zawartość pliku jest tracona, plik jest obcinany do 0 bajtów podczas otwierania.
<b>ios::ate</b>	<b>at end</b> - ustawia wewnętrzny wskaźnik pliku na jego koniec w chwili otwarcia pliku.
<b>ios::binary</b>	Informacja dla kompilatora, aby dane były traktowane jako strumień danych binarnych, a nie jako strumień danych tekstowych.

## Zamknięcie pliku

---

Otwarty plik należy zamknąć – w przeciwnym razie system operacyjny może traktować plik jako używany (nie można skasować ani przenieść), nawet po zamknięciu naszego programu.

```
fstream plik;  
plik.open("test.txt", ios::in);  
    //operacje na pliku  
plik.close();
```



## Obsługa błędów

---

Funkcja sprawdzająca czy plik otwarto prawidłowo:  
`plik.good() ; isopen()`

```
fstream plik;  
plik.open("test.txt", ios::in);  
if (plik.good() == false)  
{  
    cout << "plik nie istnieje";  
    return 0;  
}  
plik.close();
```

Rezultat wykonania operacji odczytujemy funkcją `plik.good()`  
Jeżeli zwróci ona wartość `true` oznacza to, że wykonanie operacji  
we/wy przebiegło pomyślnie.



---

## Przy próbie odczytu nie uzyskamy dostępu do pliku gdy :

1. plik nie istnieje na dysku;
2. nie posiadamy uprawnień odczytu do pliku.

## Przy próbie zapisu nie uzyskamy dostępu do pliku gdy :

1. nie posiadamy uprawnień pozwalających nam modyfikować plik;
2. nie posiadamy uprawnień do katalogu w którym chcemy utworzyć plik;
3. nośnik, na którym chcemy dokonać zapisu jest tylko do odczytu.

## Odczyt z pliku tekstowego

---

Odczytanie pojedynczego wiersza z pliku:

```
getline(zmienn_plikowa, zmienna_typu_string);
```

```
string s;  
fstream plik;  
plik.open("test.txt", ios::in);  
if (plik.good() == false)  
{  
    cout << "Brak dostępu do pliku";  
    return 0;  
}  
getline(plik, s);  
cout << s;  
  
plik.close();
```

## Odczyt z pliku tekstowego

---

Odczytanie wszystkich wierszy z pliku (druga metoda):

```
string s;  
char c;  
fstream plik;  
plik.open("test.txt", ios::in);  
if (plik.good()==false)  
{  
    cout<<"Brak dostępu do pliku";  
    return 0;  
}  
while (!plik.eof())  
{  
    getline(plik, s);  
    cout<<s<<endl;  
}  
plik.close();
```

Funkcja `.eof( )` zwraca wartość `true`, jeżeli wskaźnik plikowy dotarł do końca pliku.

## Zapis do pliku tekstowego

---

Zapis pojedynczego wiersza:

```
fstream plik;  
plik.open("test.txt", ios::out);  
plik<<"tekst zapisany w pliku";  
plik.close();  
return 0;
```

```
fstream plik;  
plik.open("test.txt", ios::out);  
plik<<"tekst zapisany w pliku"<<endl;  
plik.close();  
return 0;
```

**endl** powoduje przejście do nowego wiersza

```
fstream plik;  
plik.open("test.txt", ios::out);  
plik << 10;
```

Można także zapisywać bezpośrednio zmienne typu int

## Zapis do pliku tekstowego

---

Zapis pojedynczego wiersza za pomocą funkcji

```
plik.write(początek_bufora , dlugosc_bufora) ;
```

```
fstream plik;  
plik.open("test.txt", ios::out);  
string dane ="Ala ma kota";  
plik.write(&dane[0], dane.length());  
//zapisujemy jakieś dane zaczynając od indeksu 0  
//i kończąc na długości łańcucha  
plik.close();
```

## Zapis do pliku tekstowego

---

Zapis pojedynczego wiersza za pomocą funkcji

```
plik.write(początek_bufora , dlugosc_bufora) ;
```

```
fstream plik;  
plik.open("test.txt", ios::out);  
string dane ="Ala ma kota";  
plik.write(&dane[0], dane.length());  
//zapisujemy jakieś dane zaczynając od indeksu 0  
//i kończąc na długości łańcucha  
plik.close();
```

## Zapis i odczyt liczb

---

Przy użyciu strumienia ( << ) można do pliku tekstowego zapisywać liczby. Zarówno całkowite, jak i zmiennoprzecinkowe.

Liczby są jednak zapisywane do pliku tekstowego jako **kody ASCII poszczególnych cyfr**.

```
fstream plik;|
plik.open("test.txt", ios::out);
int x=20;
double y=20.5;
plik<<y
plik<<endl;
plik<<y;
plik.close();
```



## Zapis i odczyt liczb

Liczby są zapisywane w pliku testowym jako kody ASCII cyfr można jest przekonwertować na wartości numeryczne – za pomocą funkcji:

- `.atoi()` – liczby całkowite
- `.atof()` – liczby zmiennoprzecinkowe

```
fstream plik;  
plik.open("test.txt", ios::in);  
string s;  
getline(plik, s);  
int numer = atoi(s.c_str());  
cout << numer << endl;
```

```
string s;  
getline(plik, s);  
float numer = atof(s.c_str());  
cout << numer << endl;
```



## Przykłady

## Przykład I

Wypisanie wierszy pliku tekstowego w odwrotnej kolejności

```
1  #include <iostream>
2  #include <fstream>
3  using namespace std;
4  int main()
5  {
6      string tab[10], s;
7      int i=0;
8      fstream plik;
9      plik.open("test.txt", ios::in);
10     while (!plik.eof())
11     {
12         getline(plik, s);
13         tab[i]=s;
14         i++;
15     }
16     for (;i>=0;i--) cout<< tab[i]<<endl;
17     plik.close();
18     return 0;
19 }
```

Cała zawartość pliku kopiowana  
jest do tablicy zmiennych typu  
stringt

## Przykład 2 – logi programu

```
1  #include <iostream>
2  #include <fstream>
3  #include <cstdlib>
4  using namespace std;
5  int main()
6  { //program zlicza, ile razy został uruchomiony
7    fstream plik;
8    string s;
9    int numer;
10   plik.open("log.dat", ios::in);
11   if (!plik.good()) {
12       numer=0;
13   }
14   else{
15       getline(plik, s);
16       numer = atoi(s.c_str());
17   }
18   cout << "to jest " << numer+1 << " uruchomienie programu" << endl;
19   plik.close();
20   plik.open("log.dat", ios::out);
21   plik<<numer+1;
22   plik.close();
23   return 0;
24 }
```

Program odczytuje z pliku log.dat ile razy był już uruchomiony, następnie zamyka plik, otwiera go ponownie, tym razem do zapisu i nadpisuje jego zawartość liczbą uruchomień zwiększoną o 1.

## Przykład 3 – wyodrębnienie wyrazów z pliku

```
1  #include <iostream>
2  #include <fstream>
3  using namespace std;
4  int main()
5  {
6      string wyrazy[300], temp,s;
7      fstream plik;
8      int p, lw=0;
9      plik.open("dane.txt", ios::in);
10     if (!plik.good()){
11         cout <<"brak pliku";
12         return 0;
13     }
14     while (!plik.eof())
15     {
16         getline(plik, temp);
17         cout<< temp<<endl;
18         p=0;
19         while (p<temp.length())
20             { //w pobranym lancuchu przeskakuje znaki
39         }
40         //wypisuje tablice wyrazow
41         cout<<endl<<"-----" <<endl;
42         for (int i =0; i<lw;i++)
43             cout <<wyrazy[i] <<"-";
44         return 0;
45     }
```

- Program pobiera kolejno wszystkie linie z pliku, aż natrafi na jego koniec.
- Następnie pętla while() – (z wierszy 19-39 kodu) wyodrębnia wyrazy (jej rozwinięcie na następnej stronie).

## Przykład 3 – wyodrębnienie wyrazów z pliku

```
19     while (p<temp.length())
20     {
21         //w pobranym lancuchu przeskakuje znaki
22         //spacja { } ( )
23         while(p<temp.length() && (temp[p]==' '
24             || temp[p]=='{' || temp[p]=='{'
25             || temp[p]=='(' || temp[p]==')')) {
26             p++;
27         } //dopuki nie trafie na jeden z pomianych znakow
28         //- pobieram znki i skladam n nich lancuch s
29         s="";
30         while (p<temp.length() && temp[p]!=' '
31             && temp[p]!='{' && temp[p]!='{'
32             && temp[p]!='(' && temp[p]!=')') {
33             s+=temp[p];
34             p++;
35         }
36         //wstawiam uczytany wyraz do tablicy wyrazow
37         wyrazy[lw]=s;
38         lw++;
39     }
```

- Ten fragment programu pracuje na pobranej z pliku linii – zapisanej w zmiennej temp.
- Jego zadaniem jest wyodrębnienie wyrazów i zapisanie ich w kolejnych polach tablicy wyrazy[]
- Wewnątrz pętli while (linia 19) pracują dwie kolejne. Pierwsza (linia 22-26) pomija spacje i nawiasy, druga (linia 29-34) wczytuje kolejne znaki (niebędące spacją ani nawiasem) i dopisuje do łańcucha s.
- Na koniec (linie 36-37) łańcuch s wstawiany jest do tablicy wyrazy[] a jej licznik zwiększany o 1.



## Literatura:

---

W prezentacji wykorzystano przykłady i fragmenty:

- Grębosz J. : ***Symfonia C++***, ***Programowanie w języku C++ orientowane obiektowo***, Wydawnictwo Edition 2000.
- Jakubczyk K.: *Turbo Pascal i Borland C++ Przykłady*, Helion.

Warto zająrzeć także do:

- Sokół R. : ***Microsoft Visual Studio 2012 Programowanie w Ci C++***, Helion.
- Kernighan B.W., Ritchie D. M.: ***język ANSI C***, Wydawnictwo Naukowo Techniczne.

Dla bardziej zaawansowanych:

- Grębosz J. : ***Pasja C++***, Wydawnictwo Edition 2000.
- Meyers S.: ***język C++ bardziej efektywnie***, Wydawnictwo Naukowo Techniczne

