

Wykład: 9

Łańcuchy znaków



Tablica znaków

Łańcuch znaków – w językach C i C++ (oraz pochodnych) łańcuch znaków przechowywany jest jako tablica znaków zakończona znakiem o numerze zero (NULL)

"Ala ma kota"

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|----|----|
| A | l | a | | m | a | | k | o | t | a | /0 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |



Obsługa zmiennych łańcuchowych (łańcuchy znaków)

C++ umożliwia zastosowanie dwóch wariantów:

1. Tablica znaków (wywodzi się z języka C)
2. Użycie klasy bibliotecznej string (częściej spotykane – w języku c++ oraz językach pochodnych)



Tablica znakowa

Tablica znaków

Ciąg znaków (łańcuch) to kolejno zapisane znaki w pamięci. Dlatego można do jego zapisania użyć tablicy przechowującej znaki (char).

```
char napis[20];
```

Maksymalna długość
tablicy

Możliwe jest zainicjalizowanie takiej tablicy łańcuchem znaków

```
char napis[] = {"Ala ma kota"};
```

Tablica znaków

Można też przypisywać znak do każdego pola:

```
char tab[10] = {'a', 'b', 'c', 'd', '\0'};
```

W takim przypadku należy pamiętać o zakończeniu listy znakiem o kodzie zero `'\0'`

```
char tab2[] = {'a', 'b', 'c', 'd', '\0'};
```

Tablica znaków

Tablice znaków są kłopotliwe w obsłudze. Oto jeden z problemów:

| | | | |
|---|--|--|--|
| 5 | | | |
| 6 | | | |

```
char napis[] = {"Ala ma kota"};  
napis = "Psa";
```

Błąd – nie można przypisać
łańcucha innej długości

Rozwiązaniem jest potraktowanie tablicy jako wskaźnika na pierwszy element (char)

| | | | |
|---|--|--|--|
| 5 | | | |
| 6 | | | |

```
char *napis = {"Ala ma kota"};  
napis = "Psa";
```



Wypisywanie tablicy

```
char napis[] = {"Ala ma kota"};
```

```
cout << napis << endl;
```


Tablica znaków

Wczytywanie tablicy

```
char napis[100];  
cin >> napis;  
cout << napis << endl;
```

UWAGA - ta metoda wczyta tekst
tylko do pierwszej spacji
(obiekt cin traktuje spacje jako
separator)

Tablica znaków

Wczytywanie tablicy

Drugim sposobem, jaki można tu zastosować jest wykorzystanie metody `getline()` obiektu `cin`. Funkcja ta jest ukierunkowana na wczytywanie całych wierszy.

5
6
7
8
9

```
char napis[100];  
int bufor = 100;  
  
cin.getline(napis, bufor);  
cout << napis << endl;
```

Bufor to wielkość tej tablicy (tablica może przechować bufor - 1 znaków + znak końca tablicy)

Tablica znaków

Funkcja **get()** - może działać podobnie jak `getline`, z tą różnicą, że znak nowego wiersza nie jest odrzucany tylko pozostaje w kolejce wejściowej.

Oznacza to, że ponowne użycie `get()` nie pobierze ciągu znaków, ponieważ zakłada, że nastąpił już koniec wiersza.

Aby zaradzić temu problemowi można użyć metody `get()` bez argumentów. Pobiera ona następny znak, a więc znak końca linii zostanie usunięty i można ponownie wczytywać dane

```
5 char tablica[100];
6 cout<<"Podaj imie i nazwisko: ";
7 cin.get(tablica,100);
8 cout<<"Twoje dane osobowe: "<<tablica<<endl;
9 cin.get();
10 cout<<"Ponownie podaj imie i nazwisko: ";
11 cin.get(tablica,100);
12 cout<<"Twoje dane osobowe: "<<tablica<<endl;
```



Biblioteka string

String

Znacznie więcej możliwości daje zestaw funkcji zawartych w bibliotece String.

Aby z nich skorzystać należy dołączyć plik nagłówkowy **#include <string>**

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4  int main()
5  {
6      string napis;
7      string napis2 = "Ala ma kota";
8      return 0;
9  }
```

String

Wczytywanie tekstu ze standardowego wejścia

```
string napis;  
cin >> napis;
```

Ten sam problem jaki wystąpił przy wczytywaniu tablicy znaków – wczytany zostanie tylko pierwszy wyraz (do spacji)

Rozwiązanie:

| | |
|---|--|
| 6 | |
| 7 | |
| 8 | |

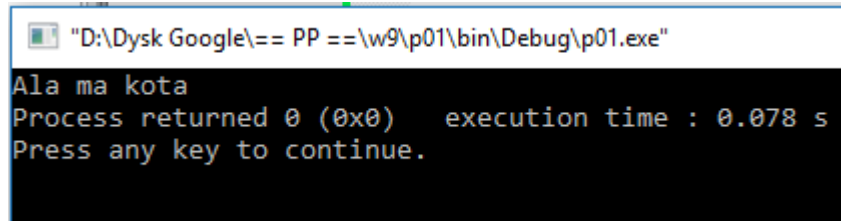
```
string napis;  
getline (cin, napis);  
cout << napis;
```

String

Sklejanie tekstów

```
6  string napis1, napis2, napis3;  
7  napis1="Ala "  
8  napis2 = "ma kota";  
9  napis3 = napis1 + napis2;  
10 cout << napis3;
```

```
6  string napis="Ala";  
7  napis+=" ma";  
8  napis+=" kota";  
9  cout << napis;
```



```
"D:\Dysk Google\== PP ==\w9\p01\bin\Debug\p01.exe"  
Ala ma kota  
Process returned 0 (0x0) execution time : 0.078 s  
Press any key to continue.
```



Sklejanie tekstów

```
6  string napis|;  
7  napis = "pierwszy " + "drugi ";
```

Taka konstrukcja nie jest dozwolona.
Można ją zastąpić następującą:

```
6  string napis;  
7  napis = "pierwszy " ;  
8  napis += "drugi";
```


String

Czyszczenie zmiennej łańcuchowej

```
string napis="Ala ma kota";  
napis = "";  
napis= "I znow zapelniam zmienna";  
napis.clear();
```

String – jako tablica

String może być traktowany jako tablica znaków.

```
string napis = "Ala ma kota";  
napis[0]='0';
```

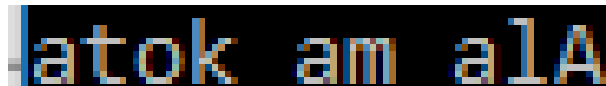
Do określania długości string-a służy metoda **length()** obiektu typu string.

```
string napis = "Ala ma kota";  
cout << napis.length();
```

String – jako tablica

Przykład – wypisywanie łańcucha od tyłu

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4  int main()
5  {
6      string napis = "Ala ma kota";
7      for (int i=napis.length()-1; i>=0; i--)
8          cout << napis[i];
9      return 0;
10 }
```



String

Wyszukiwanie znaku

```
1  #include <string>
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6      string napis = "To jest przykładowy tekst";
7      size_t pozycja = napis.find( 'y' );
8      cout << pozycja;
9      return 0;
10 }
```

String

Wyszukiwanie znaku

```
1  #include <string>
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6      string napis = "To jest przykładowy tekst";
7      size_t pozycja = napis.find( 'y' );
8      cout << pozycja;
9      return 0;
10 }
```



Wyszukiwanie znaku

Funkcja `find()` może przyjmować 2 argumenty.

1. Argument pierwszy to znak poszukiwany, natomiast
2. drugi określa miejsce od którego należy szukać znaku w tekście.

String

Wyszukiwanie wszystkich wystąpień znaku

```
1  #include <string>
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7      string tekst = "Przykładowy tekst do przeszukania";
8      size_t pozycja = tekst.find( 'a' );
9      if( pozycja == string::npos )
10     {
11         std::cout << "Nie znaleziono znaku" << endl;
12         return 0;
13     }
14     do
15     {
16         cout << "Znak na pozycji " << pozycja << endl;
17         pozycja = tekst.find( 'a', pozycja + 1 );
18     } while( pozycja != string::npos );
19     return 0;
20 }
```

String

Porównywanie łańcuchów

```
string tekst1 = "tekst1";  
string tekst2 = "tekst1|";
```

```
if (tekst1==tekst2) cout<< "TAK";  
else cout << "NIE";
```

```
string tekst1 = "napis";  
if (tekst1.compare("napis")==0) cout<< "TAK";  
else cout << "NIE";
```

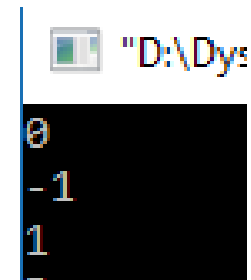



String

Porównywanie łańcuchów

```
string tekst1 = "aaa";  
string tekst2 = "bbb";  
cout<< tekst1.compare("aaa") <<endl;  
cout<< tekst1.compare(tekst2) <<endl;  
cout<< tekst2.compare(tekst1);
```

Łańcuchy można porównywać „alfabetycznie” – czyli ściślej mówiąc, po kodzie ASCII znaków.



String

substr() – kopiowanie fragmentu z łańcucha

```
string tekst = "To jest tekst";  
string fragment;  
  
fragment = tekst.substr(3, 5);  
cout << fragment;
```



"D:\Dysk C
jest

od - do

```
string tekst = "To jest tekst";  
string fragment;  
  
fragment = tekst.substr(8, tekst.length());  
cout << fragment;
```



"D:\Dy
tekst

String

replace() – zastępowanie części łańcucha

```
string tekst = "To jest tekst";  
tekst = tekst.replace(3, 4, "był");  
cout << tekst;
```

Zauważmy – liczba zastępowanych znaków nie musi być taka sama jak długość łańcucha który wklejamy.

od – ile znaków - czym

String

insert () – wstawianie znaków do łańcucha

```
string tekst = "To jest tekst";  
tekst = tekst.insert(8, "UZUPELNIIONY ");  
cout << tekst;
```

 "D:\Dysk Google\== PP ==\w9\p01\k

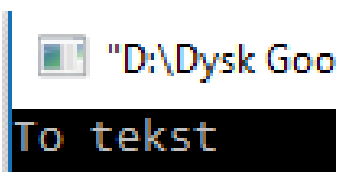
To jest UZUPELNIIONY tekst

od -- co wstawiamy

String

erase () – usuwanie części łańcucha

```
string tekst = "To jest tekst";  
tekst = tekst.erase(3, 5);  
cout << tekst;
```



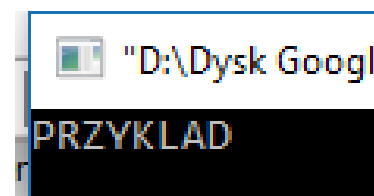
"D:\Dysk Goo
To tekst

od -- do

String

Zamiana znaku – małe litery <—> duże litery

```
string s = "PrzyKLad";  
for(int i = 0; i < s.length(); i++)  
{  
    s[i] = toupper(s[i]);  
}  
cout << s << endl; |
```



I odwrotnie..

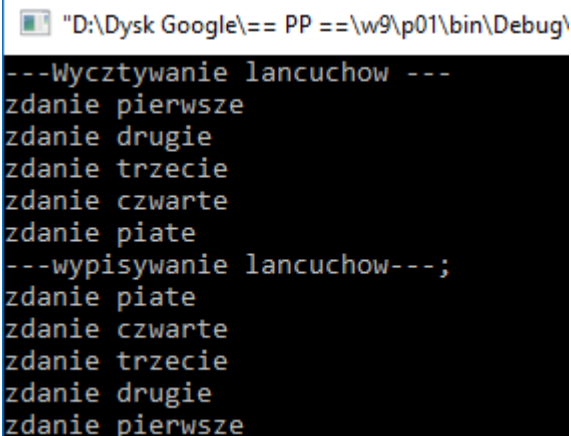
```
s[i] = tolower(s[i]);
```



Tablica string-ów

Tablice string-ów

```
string tablica[5];  
cout<<"---Wyczytywanie lancuchow ---"<<endl;  
for (int i=0; i<5; i++)  
    getline(cin, tablica[i]);  
cout<<"---wypisywanie lancuchow---;"<<endl;  
for (int i=4; i>=0; i--)  
    cout << tablica[i]<<endl;
```



```
"D:\Dysk Google\== PP ==\w9\p01\bin\Debug\  
---Wyczytywanie lancuchow ---  
zdanie pierwsze  
zdanie drugie  
zdanie trzecie  
zdanie czwarte  
zdanie piate  
---wypisywanie lancuchow---;  
zdanie piate  
zdanie czwarte  
zdanie trzecie  
zdanie drugie  
zdanie pierwsze
```


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 zajrzeć 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