

VII. Ciągi znaków – łańcuchy

7.1. Wczytywanie tekstu

Do tej pory poznaliśmy metodę wczytywania i wyświetlania liczb. Tak samo jak liczby możemy wczytać jeden znak, jednak co zrobić jeśli chcielibyśmy wczytać cały wyraz? C++ umożliwia zastosowanie dwóch wariantów:

- pierwszy wywodzi się z języka C
- drugi to użycie klasy bibliotecznej **string**

My przyjrzymy się na razie bliżej pierwszemu sposobowi, drugi sposób będzie omawiany w oddzielnym kursie.

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

```
//Ciąg znaków a tablica -----  
#include <iostream>  
#include <conio.h>  
int main()  
{  
    using namespace std;  
    //deklaracja ciągów  
    char ciag_znak[10] = {'D', 'D', 'T', ' ', '-', ' ',  
        'C', '+', '+', '\0'};  
    char zly_ciag[5] = {'B', 'E', ' ', 'B', 'E'};  
    char wyraz[50];  
  
    cout << "Podaj tekst: ";  
    cin >> wyraz;  
    cout << "Wprowadziles: \"" << wyraz  
    << "\"\" << endl;  
  
    cout << "Poprawny ciag znakow: \""  
    << ciag_znak  
    << "\"\" << endl;  
  
    cout << "Niepoprawny ciag znakow: \""  
    << zly_ciag  
    << "\"\" << endl;  
  
    getch();  
    return(0);  
}  
//-----
```

7.2. Co należy wiedzieć o wczytywaniu tekstu – analiza przykładu

Jeśli wprowadzisz tekst o długości **n** znaków to znak **n+1** będzie zawsze równy **0**. Dla większości funkcji, które operują na **łańcuchach** znaków jest to informacja, aby zakończyć wyświetlanie kolejnych znaków z tablicy. Inaczej mówiąc: tak się oznacza koniec tekstu. Pamiętaj, że numerowanie indeksów tablicy zaczyna się od zera! Ponieważ tablica

```
char zly_ciag[5] = {'B', 'E', ' ', 'B', 'E'};
```

nie posiada znaku **zero** dlatego nie przechowuje łańcucha znaków. Dlaczego jednak jest ona wyświetlana, ponieważ w pamięci jest wiele komórek, które przechowują znak **zera**.

Koniec łańcucha znaków możemy oznaczać jako '0' lub pisząc słowo **NULL**(z ang. zero).

//pierwszy sposób

```
char ciag_znak[10] = {'D', 'D', 'T', ' ', '-', ' ',  
 'C', '+', '+', '\0'};
```

//drugi sposób

```
char ciag_znak[10] = {'D', 'D', 'T', ' ', '-', ' ',  
 'C', '+', '+', NULL};
```

Oczywiście przedstawiony przykład z klamrami ({ }) jest dość czasochłonny, dlatego wprowadzono prostszy i wydajniejszy zapis.

//Ciąg znaków a tablica -----

```
#include <iostream>
```

```
#include <conio.h>
```

```
int main()
```

```
{
```

```
    using namespace std;
```

```
    //deklaracja tabeli łańcuchem znaków
```

```
    char ciag_znak[10] = "DDT - C++";
```

```
    /*kompilator sam liczy, jak dużej będzie potrzebować tablicy by pomieścić cały łańcuch*/
```

```
    char wyraz[] = "To ciekawe forum i nie tylko...";
```

```
    cout << ciag_znak << endl;
```

```
    cout << wyraz << endl;
```

```
    getch();
```

```
    return(0);
```

```
}
```

```
//-----
```

7.3. Kłopoty przy wprowadzaniu tekstu

//Problemy przy wczytywaniu łańcuchów-----

```
#include <iostream>
```

```
#include <conio.h>
```

```
int main()
```

```
{
```

```
    using namespace std;
```

```
    char ciag_znak[40];
```

```
    char tekst[20] = "C++ to jest to!!!";
```

```
    cout << "Podaj swoje imie i nazwisko programisto "
```

```
    << endl;
```

```
    cin >> ciag_znak;
```

```
    cout << "Nazywasz sie: " << ciag_znak
```

```
    << endl
```

```
    << "Ups a co sie stalo z twoim nazwiskiem!!! \n"
```

```
    << endl;
```

```

cout << "Oto tekst: " << tekst << endl;
//co się stanie jak zmienimy 4 znak w tablicy
tekst[3] = '\0';
cout << "Oto tekst: " << tekst << endl;

getch();
return(0);
}
//-----

```

Kolejną sprawą, jaka jest ważna podkreślenia to wczytywanie danych za pomocą strumienia **cin**. Strumień ma to do siebie, że wczytuje dane do zmiennej, aż do napotkania **białego znaku**. Białymi znakami nazywamy wszystkie niewidoczne znaki takie jak **spacja, tabulator, enter** no i jeszcze parę innych mniej znanych. Jeśli napiszemy dwa wyrazy oddzielone spacją, to ten drugi nie zostanie wczytany. C++ oferuje kilka rozwiązań by uporać się z tą niedogodnością. Należy również pamiętać, iż zmiana jakiegokolwiek znaku w łańcuchu na znak **zerowy**('\0'), będzie oznaczało jego skrócenie.

7.4. Metody get(), getline()

```

//getline i get w użyciu-----
#include <iostream>
#include <conio.h>
int main()
{
    using namespace std;
    const short rozmiar = 50;
    char ciag_znak[rozmiar];
    char lancuch[rozmiar];

    cout << "Podaj swoje imie i nazwisko programisto "
    << endl;
    /*składnia getline gdzie kopiować, oraz ile znaków skopiować*/
    cin.getline(ciag_znak, rozmiar);
    cout << "Nazywasz sie: " << ciag_znak
    << endl
    << "Teraz wszystko jest OK !\n";

    cout << "\nPodaj swoje 2 imiona ";
    //składnia get jest identyczna jak getline
    cin.get(ciag_znak, rozmiar);
    cout << "Podaj swoje nazwisko ";
    cin.get(lancuch, rozmiar);
    cout << endl << endl;

    cout << "Nazywasz sie : " << ciag_znak
    << " " << lancuch << endl;
    cout << "Co jest grane ? Nie masz nazwiska!?!?\n";

    getch();
    return(0);
}
//-----

```

Obie metody(klasy **iostream**) mają za zadanie wczytać cały wiersz danych(**razem z białymi znakami**) bez znaku nowego wiersza. Składnia ich obu jest identyczna

składnia metod

```
/*
```

```
cin.getline(nazwa-tabeli, liczba-wczytanych-znaków);
```

```
cin.get(nazwa-tabeli, liczba-wczytanych-znaków);
```

```
*/
```

zapis `cin.(getline, get)` mówi nam o tym, iż `cin` jest obiektem a (`getline` i `get`) są metodami tego obiektu

Metody posiadają jednak jedną różnicę, mianowicie `getline()` pobiera cały wiersz danych ze strumienia do napotkania **znaku nowej linii i usuwa ten znak ze strumienia**, a `get()` **zostawia ten znak w strumieniu**. Jeśli skompilowałeś program to zapewne zauważyłeś, iż drugie podanie nazwiska jest nie możliwe!

```
cout << "\nPodaj swoje 2 imiona ";  
//składnia get jest indentyczna jak getline  
cin.get(ciang_znak, rozmiar);  
cout << "Podaj swoje nazwisko ";  
cin.get(lancuch, rozmiar);
```

Zaraz po podaniu imion program kończy działanie. Dzieje się tak, ponieważ w strumieniu jest przechowywany **znak nowej linii**, po tym jak podałeś swoje dwa imiona i nacisnąłeś ENTER to wtedy właśnie wprowadziłeś znak nowej linii(koniec wiersza).

Pierwszy `get()` pobrał dwa imiona i zostawia znak nowej linii w strumieniu, kolejne wywołanie `get()`

```
cin.get(lancuch, rozmiar)
```

spowodowało, iż nic nie zostało wczytane gdyż w strumieniu jest tylko znak nowej linii, który jak już wiesz nie jest wczytywany. Dlaczego jednak program nie zatrzymał się byś mógł wprowadzić nazwisko? Ponieważ dla kompilatora(metody `cin`) nowa linia(koniec wiersza) to jest ENTER, który został automatycznie przekazany do `cin`.

By zaradzić tej sytuacji wystarczy wprowadzić zapis

```
cin.get(ciang_znak, rozmiar).get();  
//lub można zastosować zapis  
cin.get(ciang_znak, rozmiar);  
cin.get();  
// zapisy te są równoważne
```

W ten sposób po wczytaniu danych, następne wywołanie `get()` powoduje ominięcie znaku nowej linii(a dokładnie `cin` wczyta znak ENTER) co umożliwi wprowadzenie nazwiska. Jest jeszcze jeden sposób na to by ???

7.5. Liczby i ciągi – znów problem z nową linią?!?

```
//miks liczby i ciagi-----  
#include <iostream>  
#include <conio.h>  
int main()  
{  
    using namespace std;  
    const short rozmiar = 50;  
    char lancuch[rozmiar];  
    int rok;
```

```

cout << "W którym roku dostałeś komputer?"
<< endl;
cin >> rok;
cout << "Jak się nazywała ulubiona gra?\n";
cin.getline(lancuch, rozmiar);

cout << "Komputer dostałeś w " << rok
<< " roku." << endl;
cout << "Ulubiona gra to " << lancuch
<< "." << endl;

getch();
return(0);
}
//-----

```

Znowu pojawił się problem przy wprowadzaniu danych. Występuje tutaj podobny problem jak w przykładzie omawianym wcześniej, gdy wprowadzamy rok

```
cin >> rok;
```

i naciskamy ENTER znowu wprowadzamy do strumienia znak nowej linii. Gdy chcemy ponownie wprowadzić dane

```
cin.getline(lancuch, rozmiar);
```

natrafiamy na znak nowej linii i zostaje wprowadzony pusty wiersz. Rozwiązanie problemu jest podobne i eliminuje znak nowej linii ze strumienia.

```

cin >> rok;
cin.get();
//lub
(cin >> rok).get();

```

7.6. Ćwiczenia

1. Napisz program, który prosi o podanie, a potem wyświetla następującą informację:

```

Podaj nazwę ulicy na której mieszkasz ? Stefana Batorego
Podaj nr domu? 43/15
Na ile oceniasz swoje umiejętności programowania w skali [2 – 6]? 3
Twój staż programistyczny ? 5
Adres: Stefana Batorego 43/15
Ocena: 2
Staż: 5 dni

```

Program powinien przyjmować adres składający się więcej niż z jednego wyrazu. Dodatkowo ocena podana przez użytkownika powinna być przy wyświetleniu mniejsza o 1. Pogrubione informacje to te, które wpisuje użytkownik.

2. Napisz program, który prosi o podanie osobno imienia i nazwiska, a wyświetlający to w taki o to sposób:

```

Podałeś takie o to dane:
Nazwisko, Imię – życzę miłego dnia.

```

3. Przerób zadanie 6.3 **Tablice wielowymiarowe**. Użyj łańcuchów do zapisania nazw drużyn w zadaniu.

4.* Zadanie kontrolne

Napisz program, prostą bazę danych. Użytkownik podaje dane techniczne trzech samochodów, marka, model, pojemność silnika, prędkość maksymalna. Dodatkowo wykonaj dwa działania arytmetyczne dodaj pojemności silników poszczególnych samochodów, oraz oblicz ich średnią prędkości maksymalną. Wszystkie dane mają być wprowadzone, oraz wyświetlone w sposób czytelny dla użytkownika podczas działania programu. Sam zaprojektuj czytelny *interfejs*.

Przykładowe samochody

Fiat; 126p; 0,65; 140,

Audi; S6; 4,2; 250,

Syrena; 105; 0,84, 120.