Wprowadzenie do programowania w C: Podstawowe typy zmiennych, Operacje wejścia, wyjścia

Struktura prostego programu

Ogólna struktura prostego programu w C:

```
1 dyrektywy preprocesora
2 main()
3 {
4 deklaracje
5 instrukcje
6 }
```

Preprocesor

Dyrektywy preprocesora to komendy, które nie są zamieniane w działanie programu.

Dyrektywy zaczynają się od znaku hash (#). Oto najczęściej używane dyrektywy:

- #include - nakazuje włączenie zestawu funkcji (biblioteki). Dzięki temu, zamiast wklejać do kodu swojego programu deklaracje kilkunastu, a nawet kilkudziesięciu funkcji, wystarczy wpisać jedną linijkę! np

```
1 #include <stdio.h>
2 /* (stdio.h - STandard Input-Output, zawiera np funkcje
printf(), scanf()) */
3 #include <conio.h>
4 /* (conio.h - Console Input-Output, zawiera np funkcje,
getch(), getche(), clrscr()) */
```

- #define - pozwala definiować stałe (czyli wszystkie wystąpienia takiej stałej w programie zostaną zamienione na podane wartości przed kompilacją) np.

```
1 #define MAX_ROZMIAR 100
2 #define MIN_STOSUNEK 12.3
3 #define STACJA_DOMYSLNA A
```

Zmienne

Zmienne służą do przechowywania wartości podczas obliczeń. Każdej zmiennej przydzielony jest pewien fragment pamięci operacyjnej. O tym, jak duży jest to fragment, decyduje typ zmiennej.

Typy zmiennych (ogólniej: Typy danych)

Typ określa dziedzinę zmiennej, czyli zakres wartości, jakie może zmienna przyjmować.

W języku C liczbom rzeczywistym odpowiadają trzy typy: **float**, **double** i **long double**.

Typ *float* jest podzbiorem zbioru liczb rzeczywistych (z przedziału [-3.4*10^{38},3.4*10^{38}], najmniejsza reprezentowana liczba dodatnia wynosi 1.17*10^{-38}.

Typ *double* jest podzbiorem zbioru liczb rzeczywistych z przedziału [-1.79*10^{308},1.79*10^{308}], najmniejsza reprezentowana liczba dodatnia wynosi 2.22*10^{-308}.

Typ *long double* jest podzbiorem zbioru liczb rzeczywistych z przedziału [-1.1*10^{4932},1.1*10^{4932}], najmniejsza reprezentowana liczba dodatnia wynosi 3.4*10^{-4932}.

W języku C liczbą całkowitym odpowiadają także trzy typy: **short int**, **int** i **long int** różniące się zakresem reprezentowanych liczb. W praktyce używa się typu int lub long int. Zazwyczaj typ int reprezentuje wszystkie liczby całkowite z przedziału [-32768,32767], a typ long int reprezentuje wszystkie liczby całkowite z przedziału [-2147483648,2147483647].

Typ **char** (znakowy) zawiera wszystkie wszystkie znaki dostępne na danej maszynie, a w szczególności duże i małe litery angielskiego alfabetu oraz cyfry. Typy rzeczywiste, całkowite i znakowy nazywamy typami **prostymi**.

Deklaracje zmiennych

W języku C zmienne przed użyciem powinny być zadeklarowane. Deklarując zmienną podajemy jej typ i nazwę. Oto przykłady deklaracji zmienny różnych typów.

```
1 int wys; /* deklaracja zmiennej typu int o nazwie wys */
2 double pole; /* deklaracja zmiennej typu double o nazwie pole */
```

Jeżeli chcemy zadeklarować kilka zmiennych tego samego typu, to podajemy najpierw identyfikator typu a potem listę nazw zmiennych pooddzielanych przecinkami.

```
int wys, szer, dl, obj;
int n, m, r;
char litera1, litera2;
```

Przypisanie

Zmiennym można nadawać (oraz zmieniać) wartości za pomocą instrukcji przypisania. Dla przykładu, w wyniku wykonania instrukcji

```
wys = 10;
szer = 5;
dl = 20;
```

Zmiennym *wys*, *szer* i *dl* zostaną przypisane (nadane) wartości, odpowiednio, 10, 5 i 20. Zmienna, która ma nadaną wartość nazywamy zmienną zainicjalizowaną. Zmienne zainicjalizowane mogą występować w wyrażeniach po prawej stronie instrukcji przypisania i służyć do obliczania nowych wartości innych zmiennych (lub nawet tej samej zmiennej), jak np.

```
obj = wys*szer*dl ;
m = 2*m;
```

Inicjalizacji zmiennych można dokonywać podczas ich deklaracji, jak np.

```
int wys = 10, szer = 5, dl = 20 ;
```

Operacje wejścia/wyjścia w C++

Do wprowadzania danych służy instrukcja cin, do wypisywania cout.

```
#include <iostream.h>
main()
{
int metry;
  cout << "Podaj dlugosc boku: " ;
  cin >> metry ;
}
```

Podstawowe operatory

- operatory arytmetyczne: +, -, *, /, %, gdzie +, -, * to, odpowiednio, dodawanie, odejmowanie i mnożenie zarówno na liczbach całkowitych, jak i rzeczywistych, / oznacza dzielenie dla liczb rzeczywistych i dzielenie całkowite dla liczb całkowitych, natomiast % oznacza resztę z dzielenia dla liczb całkowitych.
- operatory porównywania: <, >, <=, >=, ==, !=.

Instrukcja wyboru (if) - przykłady

1. Zamiana wartości x na |x|.

```
1 if ( x < 0 ) x = -x;
```

2. Zamiana wartości zmiennych całkowitych a, b tak, żeby wartość a była nie większa od wartości b.

```
1 if ( a > b ){
2   int c;
3   c = a; a = b; b = c;
4 }
```

3. Podstawienie pod zmienną max wartości większej z liczb a i b.

```
1 if(a>b){
2   max=a;
3 }
4 else{
5   max=b;
6 }
```

4. Ta sama operacja zapisana w sposób skrócony.

```
1 max=(a>b)?a:b;
```

Ciąg instrukcji ujęty w nawiasy klamrowe {, } nazywamy instrukcją złożoną

INFORMACJE DODATKOWE

Operacje wejścia/wyjścia w "czystym" C

Operacje te mogą być wykonywane również w C++, jeśli dołączy się bibliotekę <stdio.h>.

printf()

Funkcja printf służy do zapisywania danych do standardowego strumienia wyjściowego. Dane te możemy przetworzyć poprzez podanie odpowiednich parametrów do funkcji.

Jako pierwsze analizowane są parametry, według których będą wypisywane dane, a następnie zmienne, w których znajdują się kolejne wartości. Przykładowy zapis funkcji mający na celu wypisanie liczby dziesiętnej wygląda tak:

```
1 printf("%d", liczba);
2 /* gdzie %d jest kodem konwersji, oznaczającym, że wypisywana
ma być zmienna liczba w postaci dziesiętnej */
```

Ogólna postać funkcji wygląda tak:

```
1 printf ("%[-][+][szerokość][.][liczba precyzji] kod_konwersji"
, argument);
```

znak **%** rozpoczyna konwersję

- **[-]** wystąpienie znaku – spowoduje wyrównanie wypisywanej wartości do lewej krawędzi wyznaczonego dla niej pola
- **[+]** wystąpienie znaku + spowoduje wyświetlenie wartości zmiennej ze znakiem
- **[szerokość]** określa minimalną szerokość pola jakie zostaje przeznaczone na wyświetlenie wartości, jeżeli liczba jej znaków jest mniejsza niż szerokość pola pozostałe miejsca zostaną wypełnione spacjami
- **[.]** jest znakiem oddzielającym od siebie liczbę określającą szerokość pola i liczbę precyzji
- **[liczba precyzji]** określa liczbę cyfr po kropce dziesiętnej dla wartości zmiennopozycyjnych, maksymalną liczbę znaków dla tekstu, minimalną liczbę cyfr dla wartości całkowitej

Kod konwersji	**Opis stosowanych kodów konwersji oraz typ argumentu**
d, i	daną wyjściową jest liczba dziesiętna, typ int
o	daną wyjściową jest liczba ósemkowa bez znaku oraz bez wiodącego zera, typ int
x, X	daną wyjściową jest liczba szesnastkowa bez znaku, typ int
u	daną wyjściową jest liczba dziesiętna bez znaku, typ int
c	dana wyjściowa to jeden znak, typ int
s	dana wyjściowa to łańcuch znaków, typ char*
f	dana wyjściowa to liczba zmiennoprzecinkowa, domyślnie 6 miejsc po przecinku, typ double
e, E	dana wyjściowa to liczba zmiennoprzecinkowa w postaci wykładniczej, typ double
g, G	dana wyjściowa taka sama jak w przypadku e, E jeżeli wykładnik jest mniejszy niż -4, w przeciwnym wypadku jest to liczba zmiennoprzecinkowa, kropka dziesiętna występuje tylko jeżeli jest po niej jakaś cyfra, typ double
p	wskaźnik (postać zależna od implementacji), typ void*
%	wypisanie znaku % , nie ma przekształcenia argumentu

Można korzystać ze znaków sterujących:

Znak	**Interpretacja**
\n	przejście na początek nowej linii (new line)

\a	alarm (sygnał dźwiękowy)
\t	przejście do następnej pozycji w tabulacji w linii
\b	cofnięcie o jedno miejsce (backspace)
\f	przejście na początek następnej strony (formfeed)
\r	przejście na początek bieżącej linii (powrót karetki)
\v	przejście do następnej pozycji w tabulacji pionowej

Przykłady:

```
1 int a = 5;
2 double b=1.2
3 printf("To jest liczba dopełniona zerami w nowej lini\n
%.4d",a);
4 printf("To jest liczba ze znakiem dopełniona zerami %-.4d",a);
5 printf("To jest liczba całkowita: %d To jest liczba
zmiennoprzecinkowa: %f",a,b);
6 printf("To jest liczba zmiennoprzecinkowa w postaci
wykładniczej %e",b);
```

scanf()

Funkcja scanf służy do odczytywania danych ze standardowego wejścia. Dane te możemy przetworzyć poprzez podanie odpowiednich parametrów do funkcji, po czym zostaną zapisane w pamięci.

Jako pierwsze analizowane są parametry, według których będą wczytywane dane, a następnie odczytywane są kolejne wartości. Przykładowy zapis funkcji mający na celu wczytanie liczby dziesiętnej wygląda tak:

```
1 scanf("%d", &liczba);
```

gdzie %d jest kodem konwersji, oznaczającym, że wczytywana ma być liczba dziesiętna, do adresu (odwołanie poprzez &), gdzie przechowywana jest zmienna liczba. **Należy pamiętać o umieszczeniu znaku & przed zmienna!!!**. Wczytywanie kończy się wraz z wystąpieniem innego rodzaju danych niż określone miedzy znakami " " (tu na przykład wystąpienie litery lub wciśnięcie entera, ponieważ wczytywane mają być wyłącznie cyfry i ewentualnie znak liczby na początku) bądź też po przekroczeniu podanego rozmiaru.

Możemy również użyć zapisu

```
1 scanf("Podaj liczby: %d %d %d", &liczba1, &liczba2, &liczba3);
```

co spowoduje przypisanie trzech liczb, jedna po drugiej pod kolejne zmienne, zastępuje to zapis:

```c
scanf("Podaj liczbę 1: %d", &liczba1);
scanf("Podaj liczbę 2: %d", &liczba2);
scanf("Podaj liczbę 3: %d", &liczba3);
```

Ogólna postać funkcji wygląda tak:

```c
scanf ("%[*] [szerokość] [rozmiar] kod_konwersji" , &argument);
```

znak **%** rozpoczyna konwersję

[szerokość] opcjonalnie możemy podać numeryczną szerokość pola

```c
scanf(" %3d", &a);
/* Do zmiennej a zostanie przypisana liczba złożona z trzech pierwszych wprowadzonych cyfr np. po podaniu na wejście 123456, a = 12 */
scanf(" %*3d %d", &a);
/* przy podaniu na wejście 123456 będzie a = 456 ponieważ wstrzymujemy przypisanie dla trzech pierwszych znaków */
```

[rozmiar] są to litery określające rozmiar przekazywanej zmiennej:

Znak	Interpretacja
h	mogące wystąpić przed następującymi kodami konwersji d, i, o, u i x gdy typem argumentu jest short
L	mogące wystąpić przed e, f i g gdy typem argumentu jest long double
l	mogące wystąpić przed d, i, o, u i x gdy typem argumentu jest long, jak również przed e, f i g gdy typem argumentu jest double

np.

```c
scanf("Podaj a: %ld", &a);
/* gdzie a jest typu long */
```

Kody konwersji są takie same jak dla funkcji printf().

Funkcja scanf zwraca ilość poprawnie dopasowanych i przypisanych wartości. Wartość zwracana może być równa 0, jeśli żaden ciąg znaków nie został dopasowany. Jeśli strumień wejściowy kończy się przed wykryciem konfliktu lub dokonaniu konwersji, zwracana jest wartość EOF.