Podstawy Programowania w Języku C

LABORATORIUM 1

11/10/2019

Tworzenie programu w C można podzielić na 3 etapy:

- **Napisanie kodu źródłowego w dowolnym edytorze tekstu.**
- **Kompilacja kodu źródłowego.**
- **Uruchomienie programu.**

Pierwszy program

```
#include <stdio.h>                    /* włączenie informacji o bibliotece standardowej */

main ()                               /* definicja funkcji o nazwie main */

{                                     /* nawiasy klamrowe otaczają instrukcje funkcji */

    printf("Hello World!\n");          /* funkcja main do wypisania ciągu znaków wywoluje
                                       biblioteczną funkcję printf  */

}
```

Komentarze

Komentarze dzieli się na dwa rodzaje:

- **Komentarz liniowy**
  Komentarz liniowy zaczyna się od dwóch znaków ukośnika **//** i kończy się wraz ze znakiem nowej linii.

- **Komentarz blokowy**
  Komentarz blokowy zaczyna się od znaków **/***, a kończy się na znakach ***/**. Komentarze te mogą obejmować większą ilość wierszy niż jeden.

```
#include <stdio.h> //  – Komentarz liniowy
/*  Komentarz
      wieloliniowy –
      blokowy
*/
main ()
{
      printf("Hello World!\n");
}
```

Typy zmiennych

Zmienne i stałe to obiekty, które zajmują pewien obszar w pamięci komputera, do którego możemy się odwołać podając ich nazwę lub adres (wskaźnik). Każda zmienna lub stała ma swój typ, co oznacza tyle, że może przyjmować wartości z zakresu danego typu.

Typ zmiennej		Rozmiar (bajty)	Zakres	
			Od	Do
int	Liczby całkowite	4	-2 147 483 648	2 147 483 647
float	Liczby rzeczywiste pojedynczej precyzji	4	$1.5 * 10^{-45}$	$3.4 * 10^{38}$
double	Liczby rzeczywiste podwójnej precyzji	8	$5.0 * 10^{-324}$	$3.4 * 10^{308}$
char	Pojedyncza litera (pojedynczy bajt)	1	-128	127
short int	Krótsze liczby całkowite, niż int	2	-32 768	32 767
long long int	Bardzo duże liczby całkowite	8	-9 223 372 036 854 775 808	9 223 372 036 854 775 807
long double	Dłuższe liczby rzeczywiste, niż double	12	1.9 * 10-4951	1.1 * 10 4932

Nazwy zmiennych i deklaracja zmiennych

Deklaracja zmiennych w języku C jest bardzo prosta:

1. Podajemy typ zmiennej

2. Podajemy nazwę zmiennej

3. Na końcu definicji stawiamy średnik

* Jeśli tworzymi kilka zmiennych danego typu, to możemy wypisywać ich nazwy po przecinku

Zastanawiacie się pewnie, jakie nazwy możemy nadawać zmiennym. Otóż istnieją pewne reguły:

1. **Nazwy mogą zawierać cyfry, litery oraz znak podkreślenia**

2. **Słowa kluczowe języka C nie mogą być używane jako nazwy zmiennych.**

3. **Pierwszym znakiem nazwy musi być litera.**

Specyfikatory konwersji (te dziwne znaczki %u, %lu, itp..)

Są to tzw. specyfikatory konwersji nakazują one funkcji printf ( ) wyświetlać na ekranie wartości zmiennych określonego typu.

char	%c
int, short	%d
long	%ld
float, double	%f
unsigned int, unsigned short	%u
unsigned long	%lu

Ćwiczenie 1

Zadeklaruj 5 zmiennych i dobierz dla nich odpowiednie typy. Powinny one zawierać np..

a) procent miesięcznego zarobku, który wydajemy na papierosy:
b) wiek twojego sąsiada,
c) odległość od twojego domu do najbliższego sklepu 24h w centymetrach (powinieneś znać),
d) temperaturę powietrza w pokoju (wartość stałoprzecinkowa).

Napisz programik, który deklaruje zmienne z Ćwiczenia 1 oraz przypisuje im odpowiednie wartości. A żeby było śmieszniej spraw, aby zostały wyświetlone na ekranie.

*__deklaracje funkcji printf wygląda następująco:__

Funkcja (“Treść”); printf(“Dzisiejsza data”);

lub Funkcja (“Treść”, Zmienna); printf(“Dzisiejsza data”, Data);

Ćwiczenie 2

Zadeklaruj dwie zmienne typu int oraz float, przypisując im jakieś wartości. Następnie spraw, aby zostały one wyświetlone na ekranie.

Stałe

Można powiedzieć, że stałe to zmienne tylko do odczytu. Raz przypisana wartość do stałej podczas pisania kodu nie może zostać zmieniona przez użytkownika podczas używania programu. Stałą definiuje się poprzez użycie słowa kluczowego const przed typem i nazwą zmiennej. A więc deklaracje stałych typu float oraz int wyglądają następująco:

```
const float nazwa StalejFloat = yyy;

const int nazwa StalejInt = xxx;

#define NAZWA wartość
```

Przykładowe definicje stałych:

```
const int dwaMiliony = 2E6;              // 2000000

const int liczbaHex = 0x3E8;                 // 3E8 to dziesiętnie 1000

const double malaLiczba = 23E-10;        // 0.0000000023

#define PI 3.14
```

Przykładowe definicje stałych:

W języku C zdefiniowano jeszcze jeden rodzaj stałej - s*tałą wyliczenia* (ang. e*numeration constant*). Wyliczenie jest listą wartości stałych całkowitych, np.

enum boolean { NO, YES }

Pierwsza nazwa na liście wyliczenia (między nawiasami klamrowymi) ma wartość 0, następna wartość 1 i tak dalej, chyba że wystąpi jawnie podana wartość.

Przykładowe definicje stałych:

Jeśli nie podano jawnie wszystkich wartości dla nazw, to kolejne nie sprecyzowane wartości stanowią postęp arytmetyczny, który rozpoczyna się od ostatnio określonej wartości.

```
enum escapes { BELL = ‘\a’, BACKSPASE = ‘\b’, TAB = ‘\t’,
               NEWLINE = ‘\n’, VTAB = ‘\v’, RETURN = ‘\r’};

enum months { JAN =1, FEB, MAR ,APR, MAY, JUN, JUL,
              AUG, SEP, OCT, NOV, DEC };
              /* miesiące: luty jest drugi, marzec - trzeci itd. */
```

Nazwy występujące w różnych wyliczeniach muszą być różne. W tym samym wyliczeniu wartości mogą się powtarzać.

Operatory arytmetyczne

+	Dodawanie
-	Odejmowanie
*	Mnożenie
/	Dzielenie
%	Dzielenie modulo (reszta z dzielenia)

```
main ()
{
	int a, b, wynik;
	a = 10;
	b = 7;
	wynik = a + b;		// wynik = a - b; lub wynik = a * b;
	printf(”%d\n”, wynik);
}
```

A co z dzieleniem?

Z dzieleniem jest w zasadzie tak samo, natomiast trzeba wspomnieć o bardzo ważnej rzeczy. Dzielenie liczb całkowitych (typ int) w wyniku da liczbę całkowitą, czyli cyfry po przecinku zostaną obcięte. Aby tego uniknąć, tzn aby w wyniku dostać liczbę rzeczywistą przynajmniej jeden z argumentów musi być liczbą rzeczywistą (float, double) oraz zmienna przetrzymująca wynik też musi być typu rzeczywistego.

Aby dokonać to można postąpić na kilka sposobów.

Pierwszy z nich, chyba najprostszy – zadeklarować argument jako zmienną typu rzeczywistego.

```
float a, wynik;
int b;
a = 10.0;
b = 7;
wynik = a / b;                  // wynik = 1.428571
printf(”%f\n”, wynik);
```

Drugim sposobem jest pomnożenie jednego z argumentów przez liczbę rzeczywistą, czyli żeby nie zmienić tej liczby, a zmienić tylko jej typ, możemy pomnożyć ją przez 1.0.

```
int a, b;
float wynik;
a = 10;
b = 7;
wynik = a*1.0 / b;                    // wynik = 1.428571
printf(”%f\n”, wynik);
```

Trzeci sposób korzysta z operatora rzutowania. Operator rzutowania ma postać:

(typ_rzutowania) wyrazenie;

```
int a, b;
float wynik;
a = 10;
b = 7;
wynik = (float)a / b;
printf(”%f\n”, wynik);
```

Program, który oblicza pole kuli.

```
P4
/* Oblicza pole kuli */
#include <stdio.h>
#define PI 3 .14
float PoleKuli;
const int R = 5;
main ( )
{
      PoleKu1i = 4*PI*R*R;
      printf ("Pole Kuli wynosi %f\n", PoleKuli) ;
      return 0;
}
```