Laboratorium Informatyki I

Ćwiczenie 2

Wstęp do programowania w C

1. Wprowadzenie

Cykl ćwiczeń laboratoryjnych z informatyki dla studentów WEiI ma na celu zapoznanie z zasadami programowania i podstawami składni języka C. W ćwiczeniu pierwszym należy się zapoznać z ogólną strukturą kodu w C, rodzajami i sposobami deklaracji zmiennych oraz z podstawowymi funkcjami wejścia i wyjścia. Instrukcja zawiera skondensowane informacje jakie z powyższej tematyki były przekazane na wykładzie.

2. Struktura kodu w C

Kod każdej aplikacji pisanej w C jest tworzony zgodnie z szablonem pokazanym w tabeli obok. Kod każdego programu napisanego w C musi zawierać funkcję główną: `int main()`. Nazwa funkcji głównej jest zastrzeżona. Funkcja `main` jest zazwyczaj typu całkowitego, ale nie jest to warunek konieczny. W większości przypadków do pisanego kodu programu dołączone będą biblioteki we/wy i systemowa. Istotna cechą kodu pisanego w języka C jest to, że prawie każda linia kodu musi być zakończona znakiem średnika (`;`). Pomiędzy deklaracją bibliotek a kodem funkcji głównej można zamieszczać dyrektywy preprocesora (np.: deklaracja stałych), deklaracje zmiennych globalnych i prototypy funkcji użytkownika.

Definicje bibliotek
`#include <nazwa.h>`
Stałe i makra
`#define nazwa wartość`
Zmienne globalne
`int x,y,z=12;`
Prototypy funkcji
`int funkcja(int x);`
Funkcja główna
`void main(void)`
Pozostałe funkcje
`int funkcja(int x)`

3. Zmienne i stałe

Zmienne w języku C mogą być globalne jak i lokalne. Zmienne globalne definiuje się poza ciałem funkcji, przed prototypami funkcji. Natomiast zmienne lokalne w ciele funkcji. Różnica sprowadza się do tego, że zmienne lokalne są widoczne tylko wewnątrz funkcji, w której zostały zdefiniowane, zaś zmienne globalne są dostępne w całym programie i wszystkich jego funkcjach. Język C dostarcza trzy podstawowe typy zmiennych: **`char`** – zmienna znakowa, **`int`** – liczba całkowita, **`float`** – liczba rzeczywista.

Zmienne definiuje się podając jej typ a następnie nazwę zmiennej. Przy deklaracji można dodatkowo nadać zmiennej wartość początkową.

```
int x,y,z=12;
float a=12.34,b;
char ax='A',BX;
```

Deklaracja stałych może być realizowana analogicznie (stałe lokalne i globalne), aczkolwiek raczej stałych lokalnych się nie stosuje. Sam proces definiowania może być

zrealizowany na dwa sposoby, poprzez definicję stałej jako zmiennej lub poprzez zdefiniowanie globalnego skrótu.

```
const float PI=3.1415;
#define PI 3.1415
```

Definiowaną zmienną można doprecyzować korzystając ze słów kluczowych modyfikujących jej wartość: **signed/unsigned** – liczby ± lub tylko dodatnie, **short** – zawężenie zakresu, **long** – poszerzenie zakresu

Zakresy dopuszczalnych wartości oraz dodatkowe informacje o poszczególnych typach zmiennych można znaleźć w dokumentacji do języka.

4. Generator liczb losowych

Jak praktycznie każdy współcześnie stosowany język programowania C także dostarcza narzędzi generowania liczb pseudolosowych. Generator umożliwia losowanie liczb całkowitych.

x=rand(); // losowanie liczby w przedziale (0, **RAND_MAX**), wartość ta jest zdefiniowana w bibliotece **stdlib.h** i wynosi 32767.

Możliwe jest ręczne sterowanie przedziałem losowania, które realizuje się zgodnie z zależnością:

```
z=x+rand()%(y-x+1);
```

, gdzie **x** jest dolną granicą losowania, a **y** górą.

Takie użycie generatora powoduje jednak, losowane są zawsze te same wartości. Aby temu zaradzić należy zainicjować generator funkcją **srand(seed)** z odpowiednią wartością **seed** zwaną ziarnem. Wygodnie jest w tym celu użyć funkcji **time(0)** lub **time(NULL)** która zwraca wartość czasu jaki minął od 1 stycznia 1970 roku. Funkcja ta wymaga biblioteki **<time.h>**.

5. Operacje wejścia i wyjścia

Funkcje we/wy umożliwiają komunikacje pisanej aplikacji z użytkownikiem. Podstawową funkcją wyjścia jest **printf()**, wysyłająca sformatowany tekst na ekran monitora. Postać ogólną funkcji pokazano poniżej:

```
printf("Tekst sformatowany",argument_1, argument_2…);
```

Tekst sformatowany składa się z ciągu znaków jakie chcemy wysłać na ekran wraz z kodami formatującymi i znakami specjalnymi. Argumenty są zmiennymi wysyłanymi do kodów formatujących umieszczonych w tekście. Przykładowy kod funkcji pokazano poniżej:

```
printf("Suma liczb %i i %i wynosi %i.\n",x,y,z);
```

W tabeli zestawiono podstawowe kody formatujące i znaki specjalne.

%c	znak z zmiennej (char)
%s	ciąg znakowy (char[])
%i	liczba całkowita (int)
%hi	liczba całkowita (short int)
%hu	liczba całkowita (unsigned short int)
%li	liczba całkowita (long, unsigned long)
%u	dodatnia liczba całkowita (unsigned int)
%f	liczba rzeczywista (float, double)
%Ld	rozszerzona liczba rzeczywista (long double)
%e	liczba zmiennoprzecinkowa x.xxxxxxe±xxx
%g	skrócona forma liczby rzeczywistej
Niektóre znaki nie mogą być wyświetlone w sposób standardowy	
\a	alarm (dźwięk)
\b	cofa kursor o jeden znak
\r	kursor na początek wiersza
\n	przejście do nowego wiersza
\t	tabulacja
Znaki specjalne umożliwiają sterowanie położeniem kursora w konsoli **\", \', \\, \?** – znaki specjalne	

Dostępne są także funkcje **puts()** i **putchar()** wysyłające na ekran ciąg znaków zakończony znakiem końca linii (**\n**) i pojedynczy znak.

Wyświetlanie danych liczbowych (rzeczywistych) za pomocą określonej liczby znaków i zadaną dokładnością jest sterowane za pomocą dwóch parametrów podawanych razem z kodem formatującym. Formatowanie liczby rzeczywiste realizowane jest zgodnie z składnią.

%[flagi][szerokosc][.precyzja][modyfikator]typ

flaga	znaczenie
-	wyrównanie do lewej (do prawej jest domyślne),
+	wymusza znak + przed liczbami dodatnimi,
spacja	wstawia spację przed liczbami dodatnimi,
0	jeżeli liczba jest, krótsza niż podana długość to uzupełnia zerami (domyślnie spacjami)

Szerokość określa minimalną liczbę znaków jaka będzie użyta do wyświetlenia liczby. Jeżeli szerokość liczby (liczba znaków) jest mniejsza od podanej to liczba zostanie uzupełniona wymaganą liczbę pustych znaków.

Precyzja umożliwia określenie liczby znaków przeznaczonych dla części ułamkowej liczby rzeczywistej.

Modyfikator umożliwia zmianę zakresu danego typu: `h – short, l – long`

Jeżeli liczbę określającą szerokość i precyzję zastąpi się '*' to konieczne jest podanie na liście parametrów przed formatowanym argumentem wartości liczbowych (zmiennych) które określą te parametry.

```
int a=5,b=2;
printf(„Iloraz liczb %f / %f = %f\n”,x,y,x/y);
printf(„Iloraz liczb %+.2f / %+.2f = %+.2f\n”,x,y,x/y);
printf(„Iloraz liczb %6.2f / %6.2f = %6.2f\n”,x,y,x/y);
printf(„Iloraz liczb %*.*f / %*.*f = %*.*f\n”,4,2,x,4,2,y,4,2,x/y);
printf(„Iloraz liczb %*.*f / %*.*f = %*.*f\n”,a,b,x,a,b,y,a,b,x/y);
```

Pobieranie danych od użytkownika realizowane jest głównie za pomocą funkcji **`scanf()`**. Uogólnioną składnie funkcji pokazano poniżej.

```
scanf(”znaki formatujące”, &argument_1, &argument_2….);
```

Znaki formatujące informują kompilator jakiego typu dane będą pobierane, argumenty natomiast informują do jakich zmiennych będą przypisane. Każdy argument musi zostać poprzedzony znakiem **`&`** wskaźnika do zmiennej. Przykładowy kod użycia funkcji pokazano poniżej.

```
scanf(”%i %f %c”,&x,&y,&z);
```

Dostępne są także funkcje umożliwiające przypisanie do zmiennej ciągu znaków (tekstu) **`gets()`** i pojedynczego znaku **`getchar()`**. Funkcja **`getchar()`** często jest wykorzystywana jako zatrzymanie działania programu na czas potrzebny użytkownikowi do przeczytania komunikatów i wznawiania działania aplikacji po naciśnięciu dowolnego klawisza.

Polskie znaki w konsoli można uzyskać ustawiając lokalizację polską. Konieczne jest podpięcie biblioteki <locale.h> i ustawienie kodowania poleceniem setlocale:

```
setlocale(LC_CTYPE,"Polish");
```

6. Obliczenia matematyczne

Podstawowe działania matematyczne realizuje się za pomocą znaków **`+`**,**`-`**,**`*`** i **`/`**. Do tej grupy zalicza się także operator dzielenia z resztą **`%`**. Przy budowaniu wzorów obliczeniowych obowiązują powszechnie znane zasady odnoszące się do kolejności działań. Dodając do kodu bibliotekę **`math.h`** uzyskujemy dostęp do szeregu funkcji matematycznych podzielonych na grupy: funkcje potęgowe, wykładnicze, trygonometryczne, … .

Szczegółowe informacje o funkcja z biblioteki **`math.h`** można znaleźć w dokumentacji dostępnej w Internecie.

Działania matematyczne można analogicznie jak wartości przypisywać do zmiennych lub wstawiać całą zależność do funkcji jak w przykładzie poniżej:

```
printf("Suma liczb %i i %i wynosi %i.\n",x,y,x+y);
```

7. Zadania

W ramach ćwiczeń należy wykonać następujące zadania w oparciu od funkcje wejścia wyjścia (printf i scanf):

a) Napisać program przeliczający temperaturę w stopniach Fahrenheita na stopnie Celsjusza. Przeliczenie realizuje się zgodnie z zależnością: $T_F=(T_C*9/5)+32$ lub $T_F\approx1.8*T_C+32$.
$T_C=(T_F-32)*5/9$ lub $T_C\approx0{,}56*(T_F-32)$.
Program operuje na liczbach rzeczywistych, z dokładnością do jednego miejsca po przecinku.

b) Napisać program generujący liczbę losową w zadanym przez użytkownika przedziale.

c) Zmodyfikować program generujący liczby losowe, aby zwracał wartości rzeczywiste z zadana przez użytkownika dokładnością (określona liczbą miejsc po przecinku: 0-4).