Laboratorium Informatyki
Programowanie w języku C
Ćwiczenie 1

1 Wstęp

Ćwiczenie pierwsze obejmuje omówienie struktury kodu pisanego w języku C, podstawowych instrukcji wejścia-wyjścia, deklaracji i inicjacji zmiennych i stałych. W instrukcji omówiono także programowanie funkcyjne oraz podział kodu na pliki nagłówkowe i źródłowe.

2 Struktura kodu w języku C

Jako środowisko programowania dla realizacji zadań laboratoryjnych wybrano Code Blocks wraz z kompilatorem MinGW. Środowisko to jest edytorem tekstowym dedykowanym do pisania kodów w języku C/C++. Autorzy udostępnili kilka wersji aplikacji dla środowiska Windows. Jedna z nich jest zintegrowana z kompilatorem MinGW. Wybranie innej wersji wymaga wcześniejszej instalacji kompilatora i ustawienia ścieżek dostępu w systemie Windows do narzędzi kompilacyjnych. W momencie utworzenia nowego projektu aplikacji konsolowej generowany jest kod startowy aplikacji „Hello Word!". Na jej podstawie można tworzyć własny projekt. Standardowo dołączone są biblioteki we/wy i systemowa.

Kod każdej aplikacji pisanej w C jest tworzony zgodnie z szablonem pokazanym jak na poniższej liście.

- Podpięcie bibliotek (`#include <nazwa.h>`)
- Stałe i makra (`#define nazwa wartość`)
- Zmienne globalne (`typ nazwa;`)
- Prototypy funkcji (`typ nazwa(typ,...);`)
- Funkcja główna (`int main()`)
- Kody prototypowanych funkcji (`typ nazwa(typ,...)`)

Funkcja `main()` jest najważniejszą funkcją programu pisanego w języku C. Od niej rozpoczyna działanie każdy program. Istotna cechą składni kodu pisanego w języka C jest to że prawie każda linia kodu musi być zakończona znakiem średnika (`;`).

3 Zmienne i stałe

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 $\pm$ 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.

W języku C zmienne mogą być zdefiniowane jako lokalne lub globalne. Zmienna globalna jest dostępna we wszystkich funkcjach pisanego kodu. Zmienna lokalna jest przypisana do danego bloku kodu `{}`. Po wykonaniu danego fragmentu kodu zmienna jest kasowana z pamięci. Powoduje to że zmienną lokalną można zdefiniować dla danej funkcji, ale także dla instrukcji wewnątrz funkcji (warunek, pętla, ...). Specyficznym typem zmiennej lokalnej jest zmienna statyczna (`static`) która jest dostępna tylko w ramach danego fragmentu kodu, ale nie jest kasowana po jego zakończeniu. Przy powtórnym wywołaniu danego kodu zmienna jest ponownie dostępna, a jej wartość jest taka jaka była w chwili poprzedniego zakończenia fragmentu kodu.

```
static int x;
static float y=10.23;
```

4 Zmienne zespolone

W języku C po dołączeniu biblioteki `<complex.h>` można zdefiniować zmienną zespoloną. Poza definicją typu bibliotek dodaje także bogatą bibliotekę funkcji matematycznych jakie można wykonywać na zmiennych zespolonych.

```
#include <complex.h>
int complex x,y;
float complex z=10+10*I;
```

Dodatkowe informacje dostępne w dokumentacji biblioteki w Internecie.

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.

Kod formatujący	Opis
`%c`	znak 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 \", \', \\, \?.

Dostępne są także funkcje `puts()` i `puchar()` wysyłające na ekran ciąg znaków zakończony znakiem końca linii (\n) i pojedynczy znak. 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 lub do opróżnienia bufora wejściowego z pojedynczego znaku..

6 Programowanie funkcyjne

Kod programu pisanego w języku C składa się z szeregu powiązanych ze sobą funkcji. Najważniejsza funkcja każdego programu to `main()`. Od niej rozpoczyna swoje działanie każdy program. Pozostałe funkcje można podzielić na dwie grupy, funkcje pochodzące z "bibliotek" podpiętych dyrektywą `#include` i funkcje napisane przez programistę w kodzie programu.

Każda funkcja w kodzie programu, poza `main()` powinna być najpierw deklarowana. Dzięki temu niezależnie od kolejności w kodzie każdą z funkcji będzie można wywołać w dowolnym miejscu programu. Deklaracja funkcji składa się i informacji o typie przez nią zwracanym, nazwie funkcji oraz liście typów zmiennych wejściowych.

```
typ nazwa(typ,...);
```

Deklaracje funkcji umieszcza się w obszarze nagłówkowy programu, przed funkcją `main()`. Kod zadeklarowanych funkcji umieszcza się po kodzie funkcji `main()`.

Instrukcje z obszaru nagłówkowego można wydzielać do osobnego pliku, zwanego nagłówkowym. Plik ten ma zazwyczaj nazwę taką samą jak plik z kodem. Nadaje mu się rozszerzenie `.h`. Oba pliki łączy się umieszczając w pliku źródłowym (`.c`) dyrektywę `#include` z nazwą pliku nagłówkowego zamkniętego w cudzysłowach.

```
#include "nazwa.h"
```

Dzięki temu informacje o podpiętych bibliotekach, liście funkcji programu, zmiennych globalnych itp jest czytelniejsza.

7 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>`.

Zadania

Na podstawie zebranych w instrukcji informacji napisać programy realizujące poniższe zadania. Programy napisać wykorzystując programowanie funkcyjne z podziałem na plik nagłówkowy i źródłowy.

- Napisać program przeliczający temperaturę w stopniach Fahrenheita na stopnie Celsjusza i na odwrót. Przeliczenie realizuje się zgodnie z zależnością:
  $T_F = \left(T_C \cdot \frac{9}{5}\right) + 32$ lub $T_F = 1.8 \cdot T_C + 32$.
- Napisać program generujący liczbę losową w zadanym przez użytkownika przedziale.
- Zmodyfikować program generujący liczby losowe, aby zwracał wartości rzeczywiste z przedziale (0,1) z dokładnością o zadanej przez użytkownika liczbą miejsc po przecinku.