Laboratorium Informatyki II
Ćwiczenie 3

1 Funkcje

Funkcje są podstawowym blokiem budującym każdy program pisany w języku C/C++. Funkcja to wydzielony fragment kodu realizujący określone zadanie. Mogą być osadzane w kodzie na cztery sposoby:

a) w głównym pliku źródłowym (`*.cpp`) w obszarze nagłówkowym - istotna jest kolejność osadzania funkcji

b) w głównym pliku źródłowym (`*.cpp`) w obszarze nagłówkowym deklaracje funkcji, poniżej funkcji `main()` kod funkcji - kolejność osadzania funkcji nie jest istotna

c) w pliku nagłówkowym głównego pliku źródłowego (`*.hpp`) deklaracje funkcji, a kod funkcji w głównym pliku źródłowym (`*.cpp`)

d) W dedykowanym funkcji(iom) zestawie plików nagłówkowego i źródłowego (odpowiednio deklaracja i kod funkcji), w obszarze nagłówkowym lub pliku nagłówkowym głównego pliku źródłowego umieszcza się dyrektywą `#include "nazwa.hpp"` podpinającą pliki z kodem funkcji do programu

W trakcie kompilacji projektu wieloplikowego w miejscu osadzenia dyrektywy `#include` wstawiany jest kod wskazanego w niej pliku. W zależności od stosowanego środowiska programistycznego procedura tworzenia i kompilacji programu wieloplikowego będzie przebiegała odmiennie. Opcja d) spośród wymienionych powyżej wydaje się najbardziej uniwersalna. W sposób znaczący ułatwia wielokrotne wykorzystanie kodu, dzięki możliwości podpinania plików `*.hpp` i `*.cpp` z kodem funkcji do kolejnych projektów.

Ze względu na sposób osadzania kodu z pliku nagłówkowego w trakcie kompilacji należy jego treść zabezpieczyć przed wielokrotnym wykonaniem. Wynika to z tego że plik nagłówkowy ze względu na konieczność przeprowadzania weryfikacji kodu przez narzędzia wspomagające programowanie może być w projekcie osadzony kilkukrotnie. Może się to więc wiązać z wielokrotnym deklarowaniem typów, zmienny lub funkcji co prowadziło by do błędów kompilacji. Kod pliku nagłówkowego powinien być zamknięty w układzie warunkowym sprawdzającym czy dany plik został już wcześniej załadowany do projektu. Jeżeli tak to pomijany jest jego kod, jeżeli nie to jest on wykonywany. Realizuje się to dzięki dyrektywom preprocesora warunkowej i deklaracyjnej, jak pokazano poniżej. W edytorze **code::blocks** odpowiedni szablon jest automatycznie generowany.

```
#ifndef NAZWA_HPP
#define NAZWA_HPP
    kod;
#endif
```

Instrukcja z pierwszego wiersza sprawdza czy znana jest nazwa podana po spacji. Zgodnie z wytycznymi standardu języka nazwę tą zapisuje się dużymi literami, a sama nazwa jest tożsama z nazwą pliku nagłówkowego z kropką zastąpioną kreską dolną. Jeżeli nazwa jest nieznana to wykonywana jest następna linia kodu, w przeciwnym wypadku następuje przeskok do instrukcji zamykającej układ warunkowy (`#endif`).

Kolejny wiersz definiuje `NAZWA_HPP` dzięki czemu przy kolejnej próbie załadowania pliku nagłówkowego będzie już znana i kod nie zostanie wykonany. Po dyrektywie `#define` umieszcza się właściwy kod pliku nagłówkowego.

program.cpp

```
#include <iostream>
#include "input.hpp"
#include "output.hpp"
int main()
{
    int x = liczba("x");
    int y = liczba("y");
    drukuj("x+y",x+y);
    drukuj("x*y",x*y);
    drukuj("4*x-2*y",4*x-2*y);
}
```

input.hpp

```
#ifndef INPUT_HPP
#define INPUT_HPP
#include <iostream>
int liczba(const char*);
#endif
```

output.hpp

```
#ifndef OUTPUT_HPP
#define OUTPUT_HPP
#include <iostream>
void drukuj(const char*,int);
#endif
```

input.cpp

```
#include "input.hpp"
int liczba(const char *nazwa)
{
    int L;
    std::cout << "Podaj " << nazwa << ": ";
    std::cin >> L;
    return L;
}
```

output.cpp

```
#include "output.hpp"
void drukuj(const char *nazwa, int L)
{
    std::cout << "Działanie " << nazwa << " wynosi: " << L << std::endl;
}
```

2 Funkcja main()

Funkcja main() tak jak każda funkcja może mieć atrybuty wejściowe. Są one przypisywane w momencie wywołania programu w konsoli. Parametry te są umieszczane po nazwie programu za pomocą listy elementów rozdzielonych spacją. Jeżeli parametr jest ciągiem tekstowym zawierającym spacje lub tabulacje należy go zamknąć w cudzysłowach. Funkcja `main()` ma dwa parametry, jeden przechowuje informacje o liczbie parametrów a drugi w tablicy 2D typu char przechowuje poszczególne parametry. Należy pamiętać że parametrem zerowym jest nazwa programu.

```
#include <iostream>
using namespace std;
int main(int argv, char **argc) //lub *argc[]
{
   if(argv>1){
      for(int i=0;i<argv;i++)
         cout << "Parametr " << i+1 << ": --> " << argc[i] << endl;
   }else
      cout << "Nie podano parametrów wywołania." << endl;
}
```

3 Liczby losowe

W języku C++ można korzystać z generatora z języka C dostępnego w bibliotece `<cstdlib>`. Zalecane jest jednak stosowanie dedykowanego narzędzi z biblioteki `<random>`. Opiera się na zmiennej odpowiadającej uprzędzeniu losującemu typu `random_device`. Dla utworzonego urządzenia należy przypisać wybrany typ generatora liczb pseudolosowych. Biblioteka dostarcza dziesięć silników na których można oprzeć generator.

- `default_random_engine` - silnik domyślny
- `minstd_rand` - minimalny standardowy generator
- `minstd_rand0` - minimalny standardowy generator
- `mt19937` - generator Mersenne Twister 19937
- `mt19937_64` - generator Mersenne Twister 19937
- `ranlux24_base` - generator Ranlux o podstawie 24
- `ranlux48_base` - generator Ranlux o podstawie 48
- `ranlux24` - generator Ranlux 24
- `ranlux48` - generator Ranlux 48
- `knuth_b` - generator Knuth-B

Samo losowanie można przeprowadzić w oparciu o jeden z powyższych generatorów lub dodatkowo określić rozkład w jakim ma być przeprowadzone. Dostępne są dwa typu rozkładów, na wartościach całkowitych (`int`) lub rzeczywistych (`float` lub `double`).

```
uniform_int_distribution<int> nazwa(zakres start,zakres koniec);
uniform_real_distribution<float> nazwa(zakres start,zakres koniec);
uniform_real_distribution<double> nazwa(zakres start,zakres koniec);
```

Poniższy kod prezentuje zastosowanie generatora liczb losowych języka C++ do losowania liczb całkowitych i rzeczywistych.

```
#include <iostream>
#include <random>

int main()
{
    std::random_device silnik;
    std::default_random_engine generator(silnik());
    std::uniform_int_distribution<int> rozklad_int(-25,25);
    std::uniform_real_distribution<float> rozklad_float(-25.0,25.0);
    std::uniform_real_distribution<double> rozklad_double(-25.0,25.0);
    for(int i=0;i<10;i++){
        std::cout << "Losowanie " << i+1 << " --> ";
        std::cout << generator() << " : ";
        std::cout << rozklad_int(generator) << " : ";
        std::cout << rozklad_float(generator) << " : ";
        std::cout << rozklad_double(generator) << " : " << std::endl;
    }
```

Zadania

W oparciu o informacje wykładowe i treść niniejszej instrukcji napisać programy. Napisać programy z oparciu o programowanie funkcyjne, przekazując parametry zadania jako atrybuty funkcji `main()`:

a) Napisać program losujący n liczb rzeczywistych z określonego przez użytkownika przedziału <A,B> bez powtórzeń. Liczby wyświetlane są w kolejności rosnącej lub malejącej. Wykorzystać sortowanie bąbelkowe.

b) Napisać program konwertujący dowolną liczbę rzeczywistą na liczbę binarną zmiennopozycyjną (32 lub 64 bitową). Program kontrolnie przeprowadza konwersję odwrotną w celu sprawdzenia poprawności działania.