Laboratorium Informatyki II
Ćwiczenie 1

1 Wstęp

Język C++ powstał jako nadzbiór języka C. Składniowo jest on w znacznym stopniu zgodny. Język C jest językiem nastawionym na programowanie strukturalne, natomiast C++ jest językiem wspierającym programowanie obiektowe. Kurs programowania w C++ w ramach laboratorium informatyki rozpocząć należy od podstaw czyli programowania strukturalnego w C++.

Każdy program w C++ analogicznie jak w C rozpoczyna się od funkcji `main()`. Programowanie strukturalne polega na podziale zadaniowym algorytmu na pojedyncze, możliwie wąskie zadania i implementacja ich w funkcjach, które następnie są wywoływane bezpośrednio lub pośrednio z funkcji `main()`.

2 Stałe, zmienne i tablice

Zmienna stałe i tablice statyczne deklaruje się w języku C++ analogicznie jak w języku C. Do typów podstawowych dochodzi jeszcze typ logiczny `bool`. Zmienne można deklarować oraz inicjować, w sposób identyczny jak w języku C. W C++ dochodzi możliwość inicjacji za pomocą identyfikatora `auto` który identyfikuje typ przypisanej wartości i automatyczne przypisuje optymalny typ danych:

```
auto x =1;
auto y = true;
auto z = 'A';
auto w = 3.1415;
```

3 Operacje wejścia wyjścia

Podstawową biblioteką dostarczającą narzędzi do obsługi strumieni wejścia i wyjścia jest biblioteka `<iostream>`. W języku C++ można także korzystać z wszystkich bibliotek języka C. Zgodnie z przyjętą nomenklaturą w kodzie C++ nazwy bibliotek z języka C poprzedza się literą `c` i pomija rozszerzenie `.h`: zamiast `<stdio.h>` należy zapisać `<cstdio>`.

W języku C++ za operacje wejścia i wyjścia odpowiadają obiekty obsługi strumienia, wejściowego `cin` i wyjściowego `cout` oraz operatory wstawienia określające kierunek przepływu danych `>>` i `<<`. Stosowana jest także funkcja przeniesienia kursora do nowego wiersza `endl`. W języku C++ stosowane jest pojęcie przestrzeni nazw, pozwalające na grupowanie zmiennych, funkcji, klas. Większość podstawowych narzędzi dostępna jest w standardowej przestrzeni nazw `std`. Powoduje to że wywołanie określonego polecenia możliwe jest po wcześniejszym odwołaniu się do właściwej przestrzeni nazw. Obiekty obsługi strumienia wejścia i wyjścia przynależą do standardowej przestrzeni nazw.

```
std::cout << Podaj wartość x:"
int x;
std::cin >> x;
```

Możliwe jest zadeklarowanie domyślnego stosowania określonej przestrzeni nazw lub określenia domyślnego stosowania określonego elementu z podanej przestrzeni nazw. W przypadku stosowania w kodzie rożnych przestrzeni nazw zalecane jest to drugie rozwiązanie.

```
#include <iostream>
using std::cout;
using std::cin;
using std::endl;
int main()
{
cout << "Podaj wartość x:";
    int x;
    cin >> x;
    cout << "x=" << x << endl;
}
```

```
#include <iostream>
using namespace std;
int main()
{
    cout << "Podaj wartość x:";
    int x;
    cin >> x;
    cout << "x=" << x << endl;
}
```

Strumień wyjściowy zawiera szereg metod umożliwiających sterowanie sposobem wyświetlania danych na ekranie. metoda `.put` umożliwia wysłanie pojedynczego znaku na ekran poprzez przypisanie do niej argumentu w postaci znaku lub wartości kodowej znaku. Natomiast metoda `.write` umożliwia wysłanie ciągu tekstowego o określonej długości.

```
#include <iostream>
using namespace std;
int main()
{
    int tab1[] = {45,55,66,77,88};
    char tab2[] = "Przykładowy ciąg tekstowy";
    cout.put('A').put('\t').put(134).put('\n');
    cout.write(tab2,sizeof(tab2)).put('\n');
    for(int i=0;i<5;i++)
        cout.write((char*)&tab1[i],sizeof(int)).put('\t');
}
```

Obiekt strumienia wyjściowego pozwala na określenie systemu liczbowego za pomocą którego wyświetlane będą liczby całkowite. Wyboru można dokonać pomiędzy systemami dziesiętnym `dec`, ósemkowym `oct` i szesnastkowym `hex`. Określenie wybranego systemu może być realizowane na dwa sposoby: `cout << hex;` lub `hex(cout)`. Aktywowany system wyświetlania liczb całkowitych obowiązuje do końca programu lub aktywacji innego systemu.

```
#include <iostream>
using namespace std;
int main()
{
    int tab[] = {45,55,66,77,88};
    cout << "DEC\tHEX\tOCT" << endl;
    for(int i=0;i<5;i++){
        cout << dec << tab[i] << "\t";
        hex(cout);
        cout << tab[i] << "\t";
        cout << oct << tab[i] << "\n";
```

```
    }
}
```

Dla danych wyświetlanych za pomocą obiektu strumienia wyjściowego `cout` można określić minimalną liczbę znaków za pomocą których dana zmienna lub wartość będzie wyświetlona. Realizuje się to za pomocą metody `.width(int)`, podając jako parametr minimalną liczbę znaków. Niestety ustawienie aktywne jest tylko dla pierwszego odwołania do obiektu `cout`. W przypadku wartości rzeczywistych (typy `float` i `double`) liczbę znaków określa się za pomocą metody `.precision(int)`, o analogicznym jak metoda `.width(int)` zakresie działania.

```
#include <iostream>
using namespace std;
int main()
{
    cout << "ułamek zwykły | ułamek dziesiętny" << endl;
    for(int i=2;i<10;i++){
        cout.width(12);
        cout << "1/" << i << " | ";
        cout.width(17);
        cout.precision(3);
        cout << 1/(float)i << endl;
    }
}
```

4 Instrukcje sterujące

Instrukcje sterujące są jednymi z najważniejszych konstrukcji w każdym języku programowania. Dzięki nim możliwe jest zaprogramowanie kodu realizującego podjęcie decyzji i wielokrotne wykonanie określonego fragmentu kodu. W języku C++ dostępne są te same instrukcje co w języku C. Ich składnia i zasady użycia są identyczne. Jedynie w przypadku pętli iteracyjnej `for()` dochodzi dodatkowa składnia realizująca iteracje po elementach tablicy. W składni wykorzystuje się identyfikator `auto`. W tak zdefiniowanej pętli zmienną iteracyjną pętli stają się po kolei elementy tablicy. W odróżnieniu od klasycznej pętli nie ma konieczności określania liczby elementów. Pętla tego typu może operować wyłącznie na tablicach statycznych.

```
#include <iostream>
using namespace std;
int main()
{
    int tab1[] = {3,8,12,36,45,77,99};
    for(auto i:tab1)
        cout << "tab1 = " << i << endl;
    char tab2[] = " Testowy ciąg znaków";
    cout << "Tekst: ";
    for(auto i:tab2)
        cout << i;
    cout << endl;
}
```

Zadania

W oparciu o informacje wykładowe i treść niniejszej instrukcji wykonać poniższe zadania:

a) Napisać program wyznaczający pierwiastki podanego przez użytkownika równania kwadratowego

b) Napisać program obliczający średnia arytmetyczną i geometryczną Liczb całkowitych z określonego przez użytkownika przedziału liczb.

c) Napisać program obliczający sumę liczb parzystych i iloczyn liczb nieparzystych dla liczb z podanego przez użytkownika przedziału

d) Napisać program wyświetlający na ekranie tablę przeliczania temperatury w stopniach Celcjusza w zadanym przez użytkownika przedziale na wartości odpowiadających temperatur w Kelwinach i stopniach Farenheita.

e) Napisać program kalkulator wykonujący wprowadzone przez użytkownika działanie matematyczne (np.: `3+3.15` lub `4^12`).