Wrocław, 25 października 2021

Podstawy Programowania

Laboratorium 2

Standardowe algorytmy iteracyjne

prowadzący: *mgr inż. Radosław Idzikowski*

1 Wprowadzenie

Celem zajęć jest poznanie podstaw programowania w języku `C/C++`, w szczególności ćwiczenia z tworzeniem programów ilustrujących zastosowanie instrukcji pętlowych (`while`, `do while`, `for`). Standardowe algorytmy iteracyjne: zliczanie, sumowanie, szukanie maksimum i minimum. Ćwiczenia z tworzeniem własnych funkcji. Funkcje bezparametrowe i zmienne lokalne. Przekazywanie parametrów przez zmienne globalne.

Podczas programowania zawsze spotykamy się z operacjami, które należy wykonywać wielokrotnie. W celu powtórzenia danego ciągu instrukcji należy skorzystać z pętli. Istnieją trzy typy pętli. Każda pętla składa się zawsze z góry określonego warunku końca oraz powtarzanych instrukcji. Poniżej pokazano w jaki sposób można wypisać ciąg liczb od `1` do `9` włącznie przy użyciu różnych pętli. Jak widać pętle można stosować zamiennie.

```cpp
#include <iostream>

int main()
{
    for (int a = 1; a < 10; a++)
        std::cout << a << " ";
    std::cout << "\n";

    int b = 1;
    while (b < 10)
        std::cout << b++ << " ";
    std::cout << "\n";

    int c = 1;
    do
        std::cout << c++ << " ";
    while (c < 10);
    std::cout << "\n";
}
```

Jednak każdą pętle charakteryzuje inna cecha/zastosowanie. Pętli `for` używa się najczęściej w przypadku kiedy pętla ma się wykonać z góry określoną liczbę razy (dokładnie tak jak w pokazanym przypadku). W przeciwnym razie, kiedy nie wiemy ile dokładnie razy ma wykonać się pętla (np.: aż do podania konkretniej wartości), korzystamy z pętli `while` lub `do while`. Jeśli chcemy zagwarantować, aby pętla wykonała się chociaż raz to należy użyć `do while`, ponieważ warunek jest sprawdziany na końcu.

W celu uporządkowania oraz poprawienia czytelności programu częstym zabiegiem jest wydzielenia fragmentów programu do funkcji. W szczególności wydzielanymi fragmentami są instrukcje, które chcemy powtarzać. Należy pamiętać, że podobnie jak w przypadku definiowania zmiennych należy zawsze określić typ funkcji (`void`, `int`, `bool` itp.). Ponadto cechą charakterystyczną dla funkcji są nawiasy "`(`", "`)`" umieszczone bezpośrednio po nazwie funkcji. Ciało funkcji (instrukcje wewnątrz funkcji) są zgrupowane wewnątrz nawiasów "{", "}". Poprzez polecenie `return` można wyjść z funkcji wraz z określoną wartością. Wartość zwracana przez funkcje musi być zgodna z jej typem! W jednej funkcji polecenie `return` można pojawiać się więcej niż jeden w zależności od rozgałęzienia funkcji warunkowych.

```cpp
#include <iostream>

int a;

bool odd() {
    return a % 2;
}

int main()
{
    a = 1;
    for (; a < 10; a++) {
        if (odd())
            std::cout <<a<< " - nieparzysta\n";
        else
            std::cout <<a<< " - parzysta\n";
    }
}
```

Powyżej przedstawiono kompletny kod programy, który ma zadanie określić parzystość liczb całkowitych z zakresu $[1, 10)$. Proszę zwrócić uwagę, że pole inicjalizacyjne w pętli `for` zostało nieuzupełnione. Pole może pozostać puste, jednak trzeba pamiętać o wszystkich średnikach.

2 Zadania

1. Napisz program z wykorzystaniem funkcji, która będzie sumować wprowadzane przez użytkownika kolejne liczby do momentu podania liczby `99`. Ponadto należy sumować liczby wyłącznie z zakresu od `-15` do `15` włącznie. Należy przechowywać jedynie ostatnią liczbę. Jeśli użytkownik wprowadzi następujący ciąg liczb: `20 -7 -30 2 15 3 99` to suma wczytanych liczb wynosi: `13`.

2. Napisz program z wykorzystaniem funkcji, która wśród 10-ciu podanych kolejno liczbach przez użytkownika zliczy liczby pierwsze. Do sprawdzania czy dana liczba jest pierwsza zalecane jest napisanie osobnej funkcji.

3. Napisz program z wykorzystaniem funkcji, która zwróci największy wspólny dzielnik dwóch liczb. Należy zastosować algorytm Euklidesa. Należy przyjąć: $a, b \in C_+$

Algorithm 1: algorytm Euklidesa

```
function NWD(a,b):
    while a ≠ b do
        if a > b then
            a ← a − b;
        else
            b ← b − a;
    return a;
```

4. Napisz program z wbudowanym `MENU` do obsługi wcześniej napisanych podprogramów. Program główny powinien działać w pętli i pozwalać na wielokrotne włączanie podprogramów. Program powinien kończyć pracę poprzez wybór odpowiedniej opcji.

Uwaga

Przypominam o przesłaniu programów na koniec zajęć, według wcześniej podanego wzoru. Proszę o niedołączanie plików innych niż rozszerzeniu `*.cpp`.