Wydział: **Zarządzania i Modelowania Komputerowego**
Kierunek**: Inżynieria Danych**
Przedmiot: **Programowanie w środowisku RAD-C++**
Rok **2** Semestr **3**

Ćwiczenie 2 – Instrukcje pętli, proste funkcje

Instrukcje pętli

```
for (<wyr1> ; <wyr2> ; <wyr3>) <instrukcja>
```

wyr1 - wyrażenie inicjujące, najczęściej służy do zainicjowania zmiennej, która steruje pętlą
wyr2 - warunek logiczny, kontroluje liczbę powtórzeń pętli
wyr3 - wyrażenie kroku, najczęściej modyfikuje zmienną sterującą pętlą

Instrukcja jest wykonywana w następujący sposób:

- wykonywane jest wyrażenie inicjujące **wyr1**,
- dopóki **wyr2** jest prawdziwe wykonywana jest **instrukcja**, a następnie **wyr3**.

```
while (<wyrazenie>) <instrukcja>
```

Dopóki wyrażenie **wyrazenie** jest prawdziwe wykonywana jest **instrukcja.** Warunek sprawdzany jest przed pierwszym wykonaniem instrukcji.

```
do <instrukcja> while (<wyrazenie >)
```

Dopóki wyrażenie **wyrazenie** jest prawdziwe wykonywana jest **instrukcja** ale warunek sprawdzany jest dopiero po pierwszym wykonaniu instrukcji.

1. Poniższy program wyznacza wartość ln(2) wg wzoru:

$$1 - 1/2 + 1/3 - 1/4 + \mathbf{...} = \ln(2)$$

```
#include <iostream.h>
#include <conio.h>
#include <math.h>
void main()
{
  float suma, znak;
  int i;
  znak = -1;
  suma = 0;
  i = 1;
  while (i < 100)
   {
   znak = - znak;
   suma = suma + znak * 1 / i;
   i = i + 1;
   };
  cout << "LN(2)="  << suma << "    Dokladnie=" << log(2.0);
  getch();
  return;
}
```

Wpisz i uruchom program. Zobacz, że program daje niepoprawne wyniki po wymianie instrukcji obliczającej sumę na (dlaczego?):

```
suma = suma + 1 / i * znak;
```

Napisz wersję programu, która przeprowadza obliczenia ln(2) instrukcją do, a potem wersję wykorzystującą instrukcję for.

2. Napisz program, który po wprowadzeniu dwu liczb:

   x – wartość argumentu

   eps – dokładność obliczeń

   wyznacza wartość exp(x) obliczoną z dokładnością *eps* z wykorzystaniem wzoru:

$$\exp(x) = \sum_{i=0}^{\infty} \frac{x^i}{i!}$$

Wczytywanie danych zorganizuj przy pomocy instrukcji cin. Kolejne wartości wyrazów szeregu obliczaj na podstawie poprzednich stosując wzór

```
wyraz = wyraz * x / i
```

gdzie i przyjmuje kolejne wartości liczbowe. Sumuj wyrazy do chwili, aż kolejny okaże się mniejszy od *eps*.
Napisz wersję programu w której obliczenia exp(x) prowadzone są w funkcji. Zastosuj zapis:

```
#include <iostream.h>
#include <conio.h>
#include <math.h>
float fexp(float x, float eps);  // prototyp funkcji
void main()
{
  float x, eps;
  cin >> x >> eps;
  cout << "Wyznaczono=" << fexp(x, eps) << "  Dokladnie=" << exp(x);
  getch();
  return;
}
float fexp(float x, float eps)
{
//  tu wstaw zapis obliczen sumy szeregu
  return suma;
  }
```

3. Zdefiniować funkcję, która dla zadanej liczby a>0 wyznacza przybliżoną wartość jej pierwiastka korzystając ze wzoru:

$$x_{n+1} = 0.5\left(x_n + \frac{a}{x_n}\right) \qquad n = 0, 1, 2....$$

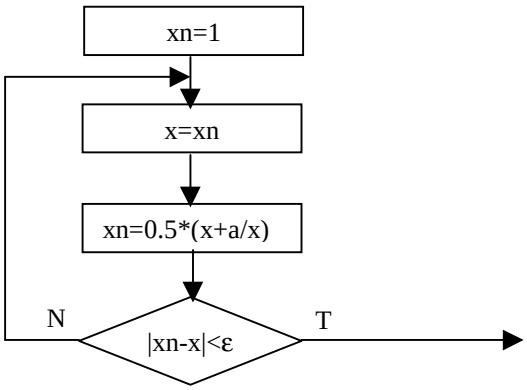
Obliczenia należy zakończyć, gdy: $|x_{n+1} - x_n| < \varepsilon$

Liczba ε oznacza dokładność obliczeń, np. ε=0,001. Przyjąć x_0=1.

Funkcję wykorzystać do policzenia poniższych wartości:

$$\sqrt{2} \quad \sqrt{5} \quad \sqrt{2a} \quad \sqrt{a+3}$$

Wskazówka: Wartość bezwzględną wyznaczać za pomocą funkcji fabs(x) zawartej w standardowej bibliotece math.h. Do obliczeń wykorzystać poniższy algorytm.



4. Napisać program, w którym zdefiniowana zostanie funkcja wyznaczająca k-ty element ciągu:

$$a_k = \frac{2(k+1)(k+2)}{3k}$$

W programie wykorzystać tę funkcję do policzenia
a) n pierwszych elementów ciągu,
b) sumy elementów o indeksach parzystych z zakresu <2, n>.

Zadania domowe:

1. Napisać program wczytujący współrzędne N punktów (N – dana liczba) i wyświetlający ile z nich leży w trzeciej ćwiartce układu współrzędnych. Sprawdzenie czy dany punkt leży w odpowiedniej ćwiartce zrealizować w postaci funkcji.

2. Napisać program wyznaczający:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Zdefiniować i wykorzystać do obliczeń funkcję o nazwie **silnia**, która dla zadanego n wyznacza n!

3. Napisać funkcję o prototypie:

```
void PiO(float r, float &P, float &V);    // parametry P i V przekazywane
                                          // przez referencję
```

obliczającą pole powierzchni P i objętość V kuli o danym promieniu r. Parametry wynikowe funkcji dla obliczonych wartości powinny być przekazywane przez referencję. Sprawdź działanie funkcji wywołując ją w programie dla kilku przykładowych wartości. Jak działa podobna funkcja o wszystkich parametrach przekazywanych przez wartość?