Lekcja 5
Pętle logiczne

schematy pętli –algorytmy: min-max – Euklides – Fibbonaci

Duża część algorytmów działa na dowolnych danych i nie jesteśmy w stanie przewidzieć, ile razy powinna wykonać się pętla (jak w przypadku pętli FOR). Może to być zależne na przykład od wpisywanego z klawiatury tekstu, podawania poprawnych lub błędnych danych, itp. W ogólnym przypadku możemy stwierdzić, że **czas działania pętli logicznej będzie zależał od spełnienia (lub nie) określonego warunku logicznego**.

Rodzaje pętli WHILE

- *warunek logiczny jest sprawdzany przed wykonaniem instrukcji w pętli*

while (warunek) instrukcja;

albo

while (warunek) { instrukcje;…; }

```
int i=1;
while (i<=10){
  cout << i << endl;
  i=i+1;
}
```

- *najpierw wykonywana jest instrukcja, a potem sprawdzany jest warunek logiczny*

do instrukcja; while (warunek);

albo

do { instrukcje;…; } while (warunek);

```
int j=1;
do {
  cout << j << endl;
  j=j+1;
} while (j<=10);
```

Zadanie - SUMA i ŚREDNIA

Podsumuj i wylicz średnią arytmetyczną ocen wpisywanych z klawiatury. Kończymy obliczenia, gdy użytkownik wpisze z klawiatury 0. Sprawdź czy oceny mieszczą się w zakresie od 1 do 6.

```
#include <stdlib.h>  //system()
#include <iostream>  //cout i cin
#include <iomanip>   //fixed i setprecision()
using namespace std;
int main() {
  setlocale(LC_ALL, "");
  cout << "SUMA i ŚREDNIA OCEN" << endl << endl;
  //zerujemy aby program nie liczył glupot
  int ile=0;
  float suma=0;
  float srednia=0;
  //nie musimy inicjowac
  //bo za pierwszym razem wpisana wartosc z klawiatury
  float ocena;
  do {
    cout << "Wpisz ocenę (0-koniec): ";
    cin >> ocena;
    if (ocena>=1 && ocena<=6){
       ile=ile+1;
       suma=suma+ocena;
       srednia=suma/ile;
    }
  } while (ocena!=0);

  cout << "Ilość ocen:    " << ile << endl;
  cout << "Suma ocen:     " << suma << endl;
  cout << fixed << setprecision(2);
  cout << "Średnia ocen: " << srednia << endl;
  system("pause");
}
```

- zawsze dawaj wskazówki użytkownikowi: „co ma robić”
- stałe i zmienne deklaruj w jednym miejscu, na początku
- inicjuj zmienne, które będą coś wyliczać
- nie musisz inicjować zmiennych, które za chwilę otrzymają wartość (np. z klawiatury)
- stosuj komentarze, aby ułatwić zrozumienie
- stosuj wcięcia i wyrównuj bloki

Przykładowy zrzut ekranu

```
SUMA i ŚREDNIA OCEN

Wpisz ocenę (0-koniec): 2
Wpisz ocenę (0-koniec): 5
Wpisz ocenę (0-koniec): 3.5
Wpisz ocenę (0-koniec): 4.33
Wpisz ocenę (0-koniec): 6
Wpisz ocenę (0-koniec): 0
Ilość ocen:    5
Suma ocen:     20.83
Średnia ocen: 4.17
```

Kolejne algorytmy zostaną opisane zgodnie ze specyfikacją wymaganą na maturze.

Zadanie - Wyszukiwanie minimum (maksimum)

Wpisujemy z klawiatury liczby i wyszukujemy minimalną (maksymalną).

Jeden z najbardziej elementarnych algorytmów, stosowany przede wszystkim podczas operacji sortowania.

Specyfikacja

Zadanie: wybierz minimalną z liczb <1..n> wpisywanych z klawiatury. Zero kończy proces.

Dane: liczba naturalna N <1..maxint>

Wyniki: liczba minimalna

Lista kroków:

1. wprowadź liczbę N
2. zmiennej MIN przypisz N
3. wprowadź liczbę N
4. jeżeli N > 0 i N < MIN przypisz MIN=N
5. jeżeli N>0 wróć do kroku 3
6. Wypisz liczbę minimalną MIN

```
int N;
cout << "wpisz liczbę: ";
cin >> N;
int MIN=N;
do {
  cout << "wpisz liczbę: ";
  cin >> N;
  if (N>0 && N<MIN) MIN=N;
} while (N>0);
cout<<"minimalna: "<<MIN<<endl;
```

Zadanie - Euklides

Napisz program (funkcję), który policzy największy wspólny dzielnik dwóch liczb wpisanych z klawiatury.

Największy wspólny dzielnik liczb wykorzystywany jest do skracania ułamków (licznik i mianownik dzieli się przez NWD) oraz do obliczania najmniejszej wspólnej wielokrotności NWW (sprowadzanie ułamków do wspólnego mianownika)

Specyfikacja

Dane: Dwie liczby naturalne A i B

Wynik: wartość NWD

Lista kroków:

1. Wprowadź A i B
2. Czy A jest różne od B
3. Dopóki A nie jest równe B
   powtarzaj punkty 4 i 5
   4. Od liczby większej odejmij mniejszą
   5. Do liczby większej przypisz różnicę
6. Wyprowadź NWD równe pierwszej liczbie (lub drugiej bo i tak są równe)

```
cout << "EUKLIDES NWD" << endl;
cout << "NWD Wpisz dwie liczby: ";
int A,B;
cin >> A >> B;
while (A != B) if (A > B) A=A-B; else B=B-A;
cout<<"NWD(" << A<<","<<B<<")="<<NWD(A,B)<<endl;
```

```
NWD wpisz dwie liczby:
45 27
NWD(45,27)=9
```

Zadanie - Fibonacci

Początkowo mamy jedną parę królików. Po miesiącu osiągają dojrzałość płciową i po kolejnym rodzi się nowa para królików. W kolejnym miesiącu znów urodzi się nowa para królików od rodziców, a młode króliki osiągną dojrzałość płciową, by za miesiąc wydać nowe pokolenie. I tak w nieskończoność. Zadaniem programu (funkcji) będzie policzenie, ile par królików będzie w stadzie po N miesiącach?

W kolejnych miesiącach będzie następująca ilość par:
1,1,2,3,5,8,13,21,34,55,89,144
i jest to suma dwóch poprzednich ciągów.

Dane: Liczba naturalna N – liczba kroków

Wynik: Ilość par królików po N krokach

Lista kroków:

1. F1=0 F2=1 - pierwsze dwa wyrazy
2. pętla od 0 do N-1
   3. wylicz F2=F2+F1
   4. wylicz F1=F2-F1
5. F2 zawiera wynik

```
int FIB(int N){
  int F1=0, F2=1, i=0;
  while (i<N){
    F2=F2+F1;
    F1=F2-F1;
    i++;
  };
return F2;
}
...
cout << "FIBONACCI Ile kroków: " << endl;
int krok;
cin >> krok;
cout<<"FIBONACCI("<<krok<<")="<<FIB(krok)<<endl;
```

Zadanie – SINUS

Wartość funkcji trygonometrycznej sin(x) można wyznaczyć za pomocą szeregu Taylora:

$$sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!}$$

Wykorzystując tą definicję napisz funkcję, który policzy wartość sinusa dla argumentu x z dokładnością n, gdzie n oznacza ilość wyrazów w szeregu Taylora. W programie głównym wylicz wartość funkcji dla kolejnych kątów od 0° do 360°, co 30°

```
#include <cmath>
...
long long SILNIA(int N){
  long long s=1, i=1;
  while (i<=N) {s=s*i;i++;} return s;
}
double RADIANY(double sto){
  return sto*M_PI/180;
}
double SIN(double x, int n){
  double s=0;
  double rad=RADIANY(x);
  int i=0;
  while (i<=n){
    s=s+pow(-1,i)*pow(rad,2*i+1)/SILNIA(2*i+1);
    i++;
  }
  return s;
}
...
cout << "SINUS - Szereg Taylora" << endl;
cout << "Wpisz kąt (stopnie): ";
double kat,krok;
cin >> kat;
cout << "Wpisz krok (0-N): ";
cin >> krok;
cout<<"SIN(" <<kat<<")="<<SIN(kat,krok)<<endl;
...
int k=0;
while (k<=360){
  cout.width(5);
  cout << k;
  cout.width(10);
  cout<<fixed<<setprecision(5);
  cout<<SIN(k,8)<<endl;
  k=k+30;
}
```

- funkcja Taylora liczy kąty dla x podawanych w tzw. mierze łukowej, w radianach. Aby funkcja przyjmowała kąty w stopniach, dopiszemy dodatkową funkcję RADIANY, która będzie zamieniać radiany na stopnie, Dodatkowo w funkcji SIN wykorzystano funkcję SILNIA oraz funkcję POW i M_PI z biblioteki CMATH.
- zwróć uwagę na pojawiające się niedokładności, coraz większe dla większych kątów.
- typ **long long** $-2^{63} \div 2^{63} - 1$, czyli przedział [-9223372036854775808, 9223372036854775807]
- wszystkie pętle za pomocą WHILE, choć można za pomocą FOR, bo zawsze znamy koniec

```
300   -0.86567
330   -0.49787
360    0.01098
```

Zadania – while i funkcje

1) *Wczytuj z klawiatury oceny, aż do wpisania znaku zera. Na ekranie wyświetlamy za każdym razem sumę i średnią ocen.*
2) *Na wykonanie szkieletu prostopadłościanu zużyto 56 cm drutu. Sprawdź, jakie mogą być poszczególne długości krawędzi.*
   - napisz funkcję, która liczy długość drutu potrzebnego na wykonanie prostopadłościanu
   - w trzech pętlach WHILE zmieniaj krawędzie 1.. 56 cm
3) *Napisz funkcję, która wyliczy sumę cyfr podanej jako parametr liczby całkowitej. Wykorzystaj dzielenie całkowite i modulo oraz pętlę while.*
   - zmodyfikuj funkcję tak, aby potrafiła obliczać sumę dla liczb rzeczywistych
4) *Napisz funkcję PIERWSZA, która sprawdza, czy podana jako parametr liczba jest pierwsza. Jako wynik otrzymujemy zero-gdy pierwsza lub dzielnik. Z jej pomocą wypisz na ekranie liczby pierwsze w zakresie 1..100*

```
wpisz ocenę: 1
   suma=1
srednia=1

wpisz ocenę: 2
   suma=3
srednia=1.5

wpisz ocenę: 3
   suma=6
srednia=2

wpisz ocenę: 4
   suma=10
srednia=2.5

wpisz ocenę: 5
   suma=15
srednia=3

wpisz ocenę: 5.33
   suma=20.33
srednia=3.38833

wpisz ocenę: 0
```

```
SZKIELET PROSTOPADŁOŚCIANU
długość krawędzi = 56
a=1 b=1 c=12
a=1 b=2 c=11
a=1 b=3 c=10
a=1 b=4 c=9
a=1 b=5 c=8
a=1 b=6 c=7
a=1 b=7 c=6
a=1 b=8 c=5
a=1 b=9 c=4
a=1 b=10 c=3
a=1 b=11 c=2
a=1 b=12 c=1
a=2 b=1 c=11
a=2 b=2 c=10
a=2 b=3 c=9
a=2 b=4 c=8
a=2 b=5 c=7
a=2 b=6 c=6
```

```
1  2  3  5  7  11  13  17  19  23  29  31  37  41  43  47  53  59  61  67  71  73  79  83  89  97
```

5) *Co zostanie wydrukowane po wykonaniu pętli:*

a) *a = 1; b = 3; while (a < b) { a = 3 * a – 1; b = 2 * b + 1; } cout << a << endl << b;*

b) *a = 21; b = 3; while (a != b) { a = a – 1; b = b + 1; } cout << a << endl << b;*

c) *a = 1000; b = 1; while (a > b) { a /= 2; b *= 2; } cout << a << endl << b;*

d) *a = 81; b = 9; while (a != b) { if (a > b) a -= b; else b -= a; } cout << a << endl << b;*