Podstawy Programowania
Zajęcia laboratoryjne 2
I rok Bioinformatyki Politechniki Poznańskiej

Sterowanie wykonywaniem programu

1 Pętle

Pętle pozwalają na wykonanie konkretnych instrukcji określoną liczbę razy. Wyróżniamy 3 różne pętle: *for*, *while* i *do while*, które różnią się składnią. Przetestuj poniższe przykłady:

- Składnia pętli *for* (kod 1 dostępny w materiałach na stronie). Aby pętla wykonała zawarte w niej instrukcje odpowiednią ilość razy potrzebna jest pewna zmienna (iterator) do której zostanie przypisana wartość początkowa, powinien być zdefiniowany warunek do jakiego momentu pętla ma się wykonywać i należy określić krok. Pętla wykonuje się od wartości początkowej ($i = 0$) do momentu aż spełniony zostanie zadany warunek ($i < 10$) i wykonuje się z odpowiednio zdefiniowanym krokiem ($i++$ czyli z krokiem jeden). Zwróć uwagę, że argumenty od, do i krok są zapisywane po średniku.

```
1 for(int i = 0; i < 10; i++){
2     printf("%d\n", i);
3 }
```

- Składnia pętli *while* (kod 2 dostępny w materiałach na stronie). Różnica między pętlą *for* a pętlą *while* wynika przede wszystkim ze składni, w której warunek początkowy jest zdefiniowany przed pętlą, a krok z jakim ma się wykonywać pętla jest zapisany wewnątrz pętli. Pętla wykonuje instrukcje znajdujące się w ciele funkcji dopóki warunek pętli jest spełniony.

```
1 int i = 0;
2 while(i < 10){
3     printf("%d\n", i);
4     i++;
5 }
```

- Składnia pętli *do while* (kod 3 dostępny w materiałach na stronie). Różnica między pętlą *while* a *do while* jest taka, że pętla *do while* wykona się przynajmniej jeden raz, nawet jeśli warunek nie będzie spełniony. Zobacz co się stanie gdy w poniższym kodzie zmienisz warunek: while(i < 10), na while(i == 10).

```
1 int i = 0;
2 do{
3     printf("%d\n", i);
4     i++;
5 }while(i < 10);
```

2 Zadania

Zad 1 - Uruchom poniższy kod, przeanalizuj jego działanie (kod 1 dostępny w materiałach). Następnie wykonaj poniższe modyfikacje:

- Zauważ, że poniższy kod wyświetla 10 elementów o wartości od 0 do 9. Co się stanie gdy zmienisz warunek limitu z $i < 10$ na $i <= 10$, ile zostanie wyświetlonych wartości?
- Zmień poniższy kod tak aby wyświetlał elementy z zakresu $[5, 7]$ oraz $[5, 7)$.
- Zmień poniższy kod tak wyświetlał co drugi element z zakresu $[0, 10]$, czyli zmień krok pętli z 1 na 2. Zauważ, że to wartość iteratora (zmiennej i) powinna się odpowiednio zmieniać.
- Zmień poniższy kod tak aby wyświetlał wartości w odwróconej kolejności, czyli od 10 do 0.

```c
#include<stdio.h>

int main(){

    for(int i = 0; i < 10; i++){
        printf("%d\n", i);
    }
    return 0;
}
```

Zad 2 - Uruchom poniższy kod i przeanalizuj jego działanie (kod 4 dostępny w materiałach)

```c
#include <stdio.h>
int main()
{
    int fahr, celsius; //zmienne typu int
    int start, limit, krok; //zmienne typu int

    start = 0; //przypisz 0 do start
    limit = 200; //przypisz 200 do limit
    krok = 20; //przypisz 20 do krok

    fahr = start; //przypisz wartosc start do zmiennej fahr, czyli fahr = 0
    while(fahr <= limit) //wykonuj petle dopoki wartosc fahr jest <= od zmiennej limit
    {
        celsius = 5 * (fahr - 32) / 9; //oblicz stopnie C i przypisz wynik do celsius
        printf("%d\t%d\n", fahr, celsius); //wypisz zmienne na ekran
        fahr = fahr + krok; //zwieksz wartosc zmiennej fahr o wartosc zmiennej krok
    }
}
```

Zad 3. Wykorzystaj powyższy program i zamień pętlę *while* na pętlę *for*. Zwróć uwagę jak tym razem został napisany identyfikator typu i co zmienia taki zapis?

```c
for(fahr = 0.0; fahr <= limit; fahr = fahr + krok)
{
    float celsius = (5.0/9.0)*(fahr - 32.0);
    printf("%3.0f \t %6.1f \n", fahr, celsius);
}
```

Zad 4. Zaobserwuj jaki jest efekt błędu w wyświetlaniu wartości zmiennej fahr w funkcji printf

```
for(fahr = 0.0; fahr <= limit; fahr = fahr + krok)
{
    float celsius = (5.0/9.0)*(fahr - 32.0);
    printf("%d %6.1f \n", fahr, celsius);
}
```

Zad 5 Zaobserwuj co się stanie jeśli zmienimy zapis ułamka z 5.0/9.0 na 5/9

```
for(fahr = 0.0; fahr <= limit; fahr = fahr + krok)
{
    float celsius = (5/9)*(fahr - 32.0);
    printf("%3.0f %6.1f \n", fahr, celsius);
}
```

Zad 6. Wykorzystaj funkcję scanf() i dodaj do programu wczytywanie wartości zmiennych limit i krok.

3 Instrukcje warunkowe *if*, *else* oraz operacje logiczne

Instrukcje warunkowe określają konkretne warunki do wykonania poszczególnych fragmentów kodu. Jeśli zadany warunek jest spełniony to wykonaj konkretne instrukcje, natomiast jeśli warunek nie jest spełniony to wykonaj inne instrukcje. Składnia instrukcji warunkowej *if else*, została przedstawiona na poniższym przykładzie (kod 5 dostępny w materiałach), pamiętaj że konieczne jest określenie warunku!

```
if(liczba % 2 == 0)
{
   printf("liczba %d jest parzysta!", liczba);
}
else
{
    printf("liczba %d jest nieparzysta!", liczba);
}
```

Jeżeli warunek w instrukcji warunkowej nie zostanie spełniony, to program pominie instrukcje w niej zawarte i wykona instrukcje zawarte w else, chyba, że nie zostaną one określone. Zobacz co się stanie jeśli nie zostanie napisana część „w innym przypadku”, czyli else.

Operatory często wykorzystane przy tworzeniu instrukcjach warunkowych:

- operator przyrównania: == (zwróć uwagę, że operator przyrównania == (podwójny znak równości) różni się od operatora przypisania = (pojedynczy znak równości))
- operator różne od: !=
- operator lub: ||
- operator oraz: &&
- operator modulo (zwraca resztę z dzielenia): %

Instrukcje warunkowe mogą być złożone z kilku warunków, przetestuj poniższy przykład (kod 6 dostępny w materiałach):

```
1
2  if(znak == 'a' || znak == 'b')
3  {
4      printf("instrukcja wykona sie jesli znak = a lub jesli znak = b\n");
5  }
6
7  if(znak == 'b' && liczba == 13)
8  {
9      printf("instrukcja wykona sie jesli znak = b i jesli liczba = 13\n");
10 }
11
12 if(znak != 'A' )
13 {
14     printf("instrukcja wykona sie jesli znak bedzie rozny od wielkiej litery A\n");
15 }
16
17 if (znak == 'a' || (liczba == 3 && inna == 7))
18 {
19     printf("TRUE\n");
20     //jesli znak = a lub jesli liczba = 3 i inna liczba = 7 to wyswietli sie: "TRUE"
21 }
22 else
23 {
24     printf("NOT TRUE\n");
25     //jesli warunek nie jest spelniony to wyswietli sie: "NOT TRUE"
26 }
```

4 Zadania

Wykonaj poniższe zadania, **używaj pętli** i instrukcji warunkowych.

Zad 1. Napisz program, który 10-krotnie poprosi użytkownika o wprowadzenie z klawiatury liczby rzeczywistej. Jeśli wprowadzona liczba jest dodatnia to program powinien ją wyświetlić.

Zad 2. Napisz program, który wyświetli wszystkie parzyste liczby całkowite z przedziału $[0, 100]$.

Zad 3. Napisz program, który wyświetli na ekranie kolejne liczby całkowite, które są podzielne bez reszty przez n oraz należą do przedziału z zakresu $[0, 100]$. Wartość zmiennej n powinna być wprowadzona z klawiatury.

Zad 4. Napisz program, który policzy ile jest liczb podzielnych przez n w przedziale $[0, 100]$. Wartość zmiennej n powinna być wprowadzona z klawiatury.

Zad 5. Napisz program, który poprosi użytkownika o podanie dwóch wartości całkowitych a i b. Następnie sprawdzi która z nich jest większa i utworzy odpowiedni przedział [mniejsza wartość, większa wartość]. Oznacz to, że jeśli wartość zmiennej a jest mniejsza, to program tworzy pętlę dla liczb całkowitych z przedziału $[a, b]$ i powinien wyświetlić tylko te liczby z podanego zakresu, które są podzielne bez reszty przez 3. W innym wypadku (jeśli mniejsza jest wartość zmiennej b) to program tworzy pętlę dla liczb całkowitych z przedziału $[b, a]$ i powinien wyświetlić tylko te liczby z podanego zakresu, które są podzielne bez reszty przez 3.

Zad 6. Napisz program wyznaczający średnią arytmetyczną dla n liczb rzeczywistych. Liczba n powinna być wprowadzona z klawiatury.

Zad 7. Napisz program, który wyświetli liczby z przedziału $[0, 100]$ w kolejności malejącej.

Zad 8. Napisz program, który wczyta 3 liczb całkowite z klawiatury, a następnie znajdzie najmniejszą z nich.

Zad 9. Napisz program, który wylosuje n liczb z przedziału $[0, m]$. Wartości zmiennych n i m powinny być wprowadzone z klawiatury. Aby wylosować wartości wykorzystaj funkcję rand(). Dowiedz się samodzielnie jaką bibliotekę należy dołączyć do programu oraz jak wygląda użycie gotowej funkcji rand(). Skorzystaj ze strony https://stackoverflow.com lub innej dowolnej.

Zad 10. Napisz program, który poprosi użytkownika o podanie szczęśliwego numerka (liczby całkowitej) z przedziału $[1, 10]$. Następie wykonaj losowanie 10 liczb całkowitych z tego samego przedziału i policz ile razy podczas tego losowania pojawił się szczęśliwy numerek.