Упражнение 05
Циклични процеси

Изчислителен процес, при който оператор или група от оператори, които се изпълняват **многократно** за различни стойности на параметрите, се нарича **цикличен**.

Когато броят на повторенията е известен предварително, цикличният процес е **индуктивен**.

Цикличен процес, при който броят на повторенията не е известен се нарича **итеративен**.

Цикъл *for*

Цикъл ***for*** обикновено се използва главно за реализиране на индуктивни циклични процеси.

```
for(<инициализация>; <условие>; <корекция>) <оператор>;
```

Изпълнението започва от *инициализацията*, която включва дефиниция с инициализация на една или повече променливи или няколко операции за присвояване, които са отделени с оператор запетая (,) и не завършва с ;. Следва проверка на *условието*, което е зададено с булев израз. Ако *условието* не е изпълнено, цикълът приключва, без *тялото* да бъде изпълнено нито веднъж. Ако *условието* е изпълнено, последователно се изпълняват следните действия: изпълнение на *тялото*, изпълнение на операциите в частта *корекция* и оценка на *условието*, докато условието има стойност истина.

Областта на действие на променливите, дефинирани в частта инициализация или в тялото на цикъла, е от дефиницията до края на цикъла.

Ако частта *условие* е празна, се подразбира *истина*. Частта *корекция* може да бъде преместена в тялото на цикъла. Инциализацията може да се извърши преди тялото на цикъла.

Ако в тялото на цикъла трябва да бъдат включени повече от един оператори, те трябва да бъдат обединени в *блок*.

Цикъл *while*

С този оператор за цикъл може да се реализира произволен цикличен процес.

```
while(<условие>) <оператор>;
```

Оценява се *условието*. Ако стойността му е лъжа, изпълнението на цикъла се прекратява и тялото не се изпълнява нито веднъж. Ако стойността на условието е истина, се изпълнява *тялото* на цикъла и се преминава към следваща проверка на *условието*. Цикълът се изпълнява, докато условието е истина.

Верифициране на входа

За да се направи пълна проверка за коректност на входните данни, ако те принадлежат на даден числов интервал, трябва да се провери:

1) дали е било прочетено число;
2) дали това число е в искания интервал.

Пример: Дадена е стойността на лицето на квадрат. Да се намери дължината на страната му.

Лицето на квадрата трябва да бъде положително число. Но какво се случва, ако не е въведено число?

```
double area;
cout << "Area = "; cin >> area;
```

Последователността, в която се изпълняват двете проверки е от съществено значение. Ако от клавиатурата не е въведено число, в променливата `area` ще се запише стойност 0. Каква е вероятността тази стойност да отговаря на условията за входните данни?

Изразът `cin >> area;` има стойност и тя е потокът `cin`. Ако след операцията, потокът е в добро състояние, то операцията е била успешна. Променлива от тип поток може да се постави в условието на оператор *if*.

```
if(!cin) // cin.fail()
{
    cout << "Number is expected!\n";
    return 1;
}

if(area < 0)
{
    cout << "The area should be positive number!\n";
    return 1;
}
```

Задачи
Цикъл for

Задача 1
Да се напише програма, която по зададено естествено число намира факториел.

$$n! = 1.2.3 \dots (n-1).n$$

Ако задачата е решавана, да се замени със следната...

Задача
Да се напише програма, която пресмята сумата от всяко *n*-то число, като се започне от зададено цяло число *start* и се достигне до зададено цяло число *end*.

Задача 2
Да се напише програма, която намира средно-аритметичното на всички числа в интервала $[start, end], start < end$. Да се добави проверка за валидност на входните данни.

Задача 3
Да се напише програма, която по дадено реално число x намира стойността на израза

$$\Big(\dots\Big(\big((x+2)x+3\big)x+4\Big)+\dots+10\Big)x+11.$$

Задача 4
Дадено е цяло число *n*, $n \geq 0$. Да се напише програма, която намира сумата:

a) $S = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \dots + (-1)^{n-1}\frac{1}{n}$;

b) $S = 1 + \frac{1}{2} - \frac{2}{3} + \frac{1}{4} + \frac{1}{5} - \frac{2}{6} + \dots + \frac{a}{n}$, където a е -2, ако n се дели на 3 и a е 1, в противен случай.

Резултатът да бъде изведен до 3 символ след десетичната запетая.

Задача 5
Нека m и n са естествени числа, $n \geq 1, m \geq 1$. Да се напише програма, която определя броя на елементите от серията числа $i^3 + 7.i^2 + n^3, i = 1, \dots, n$, които са кратни на m. Да се направи проверка за коректност на входните данни.

Задача 6
Да се напише програма, която намира всички трицифрени числа от интервала $[m, n], m < n$, на които като се задраска цифрата на десетиците, намаляват цяло число пъти.

Задача 7
Едно естествено число се нарича съвършено, ако е равно на сбора от всички свои делители без самото число. Например $28 = 1 + 2 + 4 + 7 + 14$. Да се напише програма, която намира всички съвършени числа в даден интервал. Да се направи проверка за коректност на входните данни.

Цилъл while

Задача 8
Да се напише програма, която въвежда от клавиатурата редица от цели числа и намира средно-аритметично на четните числа. Въвеждането продължава до въвеждане на 0.

Може да се замени със следната:

Задача
Да се напише програма, която въвежда цели числа до въвеждането на положително. Използва се постигане на коректен вход.

Задача 9
Да се напише програма, която намира НОД по алгоритъма на Евклид.

Нека a и b са естествени числа. Редицата $r_1 > r_2 > \cdots > r_n$ е определена по следния начин:

$a = bq_0 + r_1$

$b = r_1q_1 + r_2$

$r_1 = r_2q_2 + r_3$

...

$r_{n-1} = r_nq_n$, където r_n е последният ненулев член на редицата. НОД$(a, b) = r_n$.

Задача 10
Да се напише програма, която намира най-голяма цифра в записа на дадено естествено число. Да се намери броя на цифрите в записа на естественото число.

Задача 11
Да се напише програма, която проверява дали дадена цифра се съдържа в записа на дадено естествено число. Да се направи валидалия на входните данни.

Задача 12
Да напише програма, която намира симетричното число на дадено цяло число. Симетрично на дадено цяло число се нарича число със същия знак и същите цифри, но записани в обратен ред.

Да се провери дали цялото число е палиндром.

Задача 13
Да се напише програма, която проверява дали дадено естествено число е просто.