Упражнение 04
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 *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. Изпълнението приключва, когато условието се оцени като лъжа. Ако в тялото на цикъла трябва да бъде включен повече от един оператор, то трябва да бъде оформено като *блок*.

Цикъл do-while

Операторът, подобно на *while*, може да реализира произволен цикличен процес.

```
do
  <оператор>;
while(<условие>);
```

Първо се преминава веднъж през *тялото* на цикъла. Ако тялото на цикъла съдържа повече от един оператор, то трябва да бъде оформено като блок.

Следва проверка на *условието*. Ако стойността на условието е истина, отново се изпълнява *тялото* на цикъла. Цикълът приключва, когато условието се оцени като лъжа.

Каква е разликата между *while* и *do-while*?

Цикълът *while* се нарича още цикъл с пред-условие. След като условието се проверява преди да се влезе в тялото на цикъла, то е възможно тялото на цикъла да не бъде изпълнено нито веднъж.

Тялото на цикъла *do-while* се изпълнява поне веднъж.

Оператори *break* и *continue*

Използването на оператора *break* в тялото на цикъл води до прекъсването му. Изпълнението продължава с първия оператор след тялото на цикъла.

Използването на оператора *continue* в тялото на цикъл *while* прехвърля изпълнението към проверката на *условието*.

Внимание!

Ако операторът *continue* бъде използван преди промяната на променливата/ите, използвана в проверката на условието, може да се предизвика безкраен цикъл.

В посочения по-долу пример ще бъдат изведени цифрите от 0 до 4. Когато number достигне стойност 5, условието на оператора if е изпълнено и среща оператора *continue*. Изпълнението се прехвърля към проверката на условието на цикъла, като корекцията се пропуска. Програмата изпада в състояние на безкраен цикъл.

```
int number = 0;
while(number < 10)
{
  if(number == 5)
    continue;

  cout << number << ' ';
  ++number;
}
```

Използване на операторите while и do-while

Задача 0
(Използва се постигане на коректен вход)

Да се напише програма, която въвежда цели числа до въвеждането на положително.

Задача 1
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.

Задача 2
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 3
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 4
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 5
Да се напише програма, която проверява дали дадено естествено число е просто.

Задача 6
Едно положително цяло 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 проверява дали дадено естествено число е съвършено.

Задача 7
Да се напише програма, която намира цялото число, което се получава от положителното цяло число n, като се задраска k-тата ($k \leq$ броя на цифрите на n) му цифра отдясно наляво.

Например:

Ако n = 12345, а k = 2, резултатът е 1235. Ако n = 12345, а k = 5, резултатът е 2345. Ако n = 9, k = 1, резултатът е 0.

Задача 8*
Да се напише програма, която променя дадено положително цяло число като заменя всяко срещане на дадена цифра с друга.

Задача 9
Да се напише програма, която проверява дали в записа на дадено положително цяло число има повтарящи се цифри.

Задача 10
Едно положително цяло число се нарича автоморфно, ако се съдържа в края на своя квадрат.

Например: 5 се съдържа в квадрата си 25.

Да се напише програма, която проверява дали дадено положително цяло число е автоморфно.

Задача 11
Да се напише функция, която пресмята приближено

$$e^x = 1 + \frac{1}{1!}x + \frac{1}{2!}x^2 + \cdots + \frac{1}{n!}x^n$$

за дадени стойности *x* и точност ε. Сумирането се прекратява, когато абсолютната стойност на поредният член на редицата стане по-малък от предварително зададено ε, достатъчно малко положително число. Да се сравни резултата със стойността, пресметната от вградената функция *exp*.

Да се напише програма, която пресмята горната функция, но сумирането се прекратява, когато към сумата се добавят две поредни събираеми, чиято разлика по абсолютна стойност не надвишава ε.

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 (,). 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*;
- 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*.

Използването на оператора *break* в тялото на цикъла води до прекъсването му. Изпълнението продължава с първия оператор след тялото на цикъла.

Използването на оператора *continue* в тялото на цикъл *for* прехвърля изпълнението в частта *корекция*.

Използване на оператор *for*

Задача 0
Да се напише програма, която по дадено естествено число n намира неговия факториел.

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

Да се напише програма, която намира сумата на нечетните числа в интервала [1; n], които не се делят на 7. Да се използва оператор `continue`.

Задача 1
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
Да се напише програма, която намира средно-аритметичното на първите n наброй числа на Фибоначи.

$$Fib(0) = 0$$
$$Fib(1) = 1$$
$$Fib(n) = Fib(n-1) + Fib(n-2)$$

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
Да се напише програма, която извежда на екрана всички трицифрени числа *abc*, за които е в сила $a + c = b$. Например: 121, 341, 891, 770…

Бележка: b е цифра, т.е. принадлежи на интервала 0…9. $a + c \leq 9 \rightarrow c \leq 9 - a$.

Задача 7
Да се напише функция, която по дадено естествено число *n*, пресмята сумата:

$$S = 1.2 + 2.3.4 + 3.4.5.6 + \dots + n.(n+1) \dots 2n.$$

Задача 8
Да се напише функция, която пресмята сумата:

어어쾌

a) $S = \frac{1!}{\frac{1}{2}} + \frac{2!}{\frac{1}{2}+\frac{1}{3}} + \cdots + \frac{n!}{\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{n+1}}$, където *n* е естествено число;

b) $S = \frac{2}{3.4!} + \frac{4}{5.6!} + \cdots + \frac{2n}{(2n+1).(2n+2)!}$, където *n* е естествено число.

Литература

Тодорова, М., Армянов, П., Петкова, Д., & Георгиев, К. (2008). *Сборник от задачи по програмиране на C++: част първа Увод в програмирането*. София: ТехноЛогика.