

Лабораторная работа №4

«Циклы»

Теоретическое введение

Язык C++ поддерживает 3 вида циклов:

- while — цикл с предусловием;
- do-while — цикл с постусловием;
- for — цикл со счетчиком.

Синтаксис и блок-схемы этих циклов представлены в таблице ниже.

Цикл	Синтаксис	Блок-схема
while	<pre>while (условие) { блок операторов }</pre>	<pre> graph TD Start(()) --> Cond{условие} Cond -- да --> Block[блок операторов] Block --> Start Cond -- нет --> End(()) </pre>
do-while	<pre>do{ блок операторов }while (условие);</pre>	<pre> graph TD Start(()) --> Block[блок операторов] Block --> Cond{условие} Cond -- да --> Start Cond -- нет --> End(()) </pre>
for	<pre>for (иниц-ия; условие; инкремент) { блок операторов }</pre>	<pre> graph TD Start(()) --> Init[иниц-ия] Init --> Cond{условие} Cond -- да --> Block[блок операторов] Block --> Incr[инкремент] Incr --> Start Cond -- нет --> End(()) </pre>

Пример. Составим блок-схему программы, которая выводит на экран таблицу степеней двойки:

n	2 ⁿ
0	1
1	2
2	4
3	8
4	16
5	32
6	64

Для этого заведем две переменных:

- n , которая отвечает за показатель степени;
- val , которая содержит текущее значение 2^n .

Блок-схема будет выглядеть следующим образом:

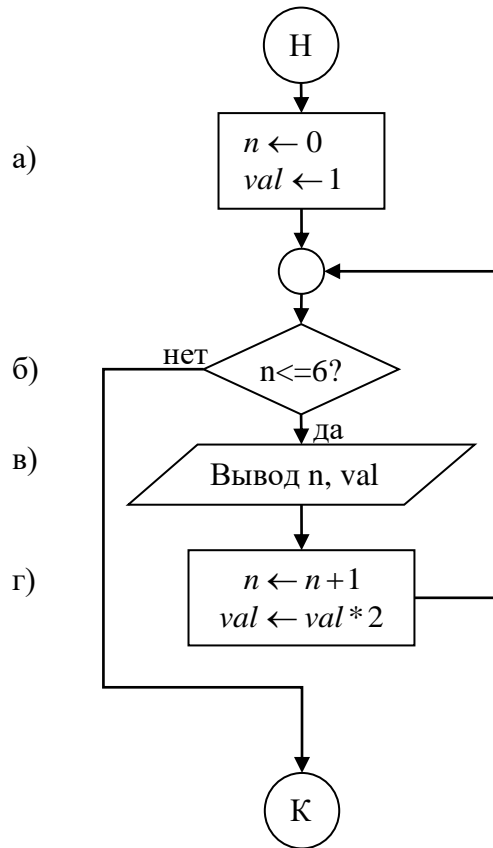


Рис. 1. Блок-схема алгоритма вывода на экран таблицы степеней числа 2

Управление выполнением цикла

При разработке циклических алгоритмов нередко возникает необходимость как либо «вмешаться» в стандартный ход работы цикла `for`, `while` или `do-while`. Например, бывает нужно *досрочно прервать* цикл или досрочно перейти к *следующей итерации*, прервав текущую. Для этих целей в C++ имеются 2 специальных оператора: **break** и **continue**. Подробнее см. главу 2, п. 2.18 книги Дейтела.

Оператор **break** осуществляет немедленный выход из цикла.

Оператор **continue** — пропуск оставшейся части текущей итерации и переход к началу следующей.

Пример. Напишем программу, которая накапливает сумму элементов входной последовательности чисел до тех пор, пока на вход не поступит отрицательное число.

```
int sum = 0;
cout << "Введите числа:"
while(1){                                     // «бесконечный» цикл: 1 всегда истинно
    int n;
    cin >> n;                                // вводим число
    if(n < 0)                                 // если n < 0 - конец цикла
        break;
    sum = sum + n;                            // положить n «в копилку»
}
cout << "Сумма равна: " << sum << "\n";
```

Рис. 2. Программа нахождения суммы элементов входной последовательности до 1-го отрицательного

Если оператор **break** в этом примере заменить оператором **continue** — то получится программа, которая складывает положительные (а также 0) элементы входной последовательности, игнорируя отрицательные. Что Вы можете сказать о конечности данного алгоритма?

Флаговые переменные

В приведенном на рис. 2 фрагменте кода управление выполнением цикла осуществляется полностью «вручную»: сам цикл устроен, по сути, как бесконечный, а внутри его тела уже принимается решение о его завершении в нужный момент.

Иногда бывает удобно скомбинировать данный подход с «обычным»: заводится специальная **переменная-флаг** которая сигнализирует о необходимости завершения цикла. Изначально она равна нулю, но при наступлении определенных условий она «взводится» — устанавливается в 1. Так как в условии цикла ее значение постоянно проверяется, то при установке ее в 1 цикл завершается.

Удобство данного подхода состоит в том, что он дает возможность *идентифицировать причину* завершения цикла (если таких причин может быть несколько).

Например, нам нужно написать программу, которая считывает с клавиатуры некоторое заранее определенное количество положительных чисел и находит их сумму. В случае обнаружения отрицательного числа — подсчет прекращается с соответствующим уведомлением.

```
int N;
cout << "Сколько чисел? ";
cin >> N;

cout << "Введите числа: ";
int sum = 0;
int flag = 0;                                // флаг выхода, пока 0
for(int i=0; flag==0 && i<N; i++){           // цикл на N итераций
    int n;                                    // или пока не взведен флаг
    cin >> n;
    if(n < 0){
        flag = 1;                            // взводим флаг!
        continue;
    } // if negative
    sum = sum + n;
}
cout << "Сумма равна: " << sum << "\n";
if(flag==0)
    cout << "Все " << N << " чисел успешно обработаны.\n";
else
    cout << "Досрочное завершение. Обработаны не все числа!\n";
```

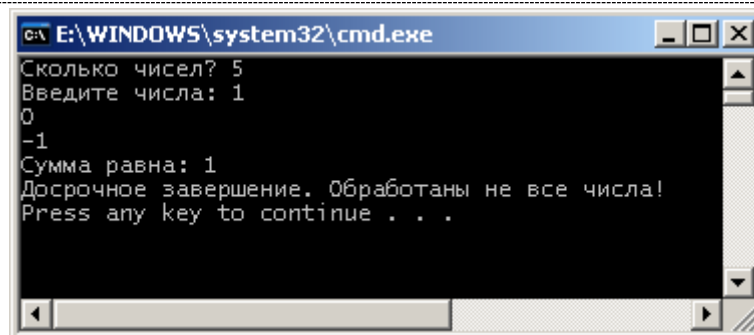


Рис. 2. Программа нахождения суммы элементов входной последовательности до 1-го отрицательного

Задания для самостоятельного решения

Задание 1. Составьте по блок-схеме на рис. 1 программу и протестируйте ее.

Задание 2. Перепишите получившуюся программу еще двумя различными способами: с использованием цикла `for` и цикла `do-while`. По поводу синтаксиса см. главу 2 книги Дейтела.

Задание 3. Используя цикл `while`, напишите программу, которая считывает с клавиатуры натуральное число M и натуральное «основание» k , после чего находит максимальную степень, в которую надо возвести k , чтобы результат все еще не превышал M . Например, если $M=87$, $k=5$, результат будет равен 2, т.к. $5^2 < 87$, а $5^3 > 87$ (этот алгоритм нам еще пригодится в программе перевода числа в произвольную систему счисления). Нарисуйте блок-схему данной программы в RAPTOR'е и проверьте правильность ее работы. Включите в отчет диаграмму потоков данных.

Задание 4. Составьте программу вычисления факториала целого неотрицательного числа N . Проверьте корректность ее работы при $N=0$. Включите в отчет диаграмму потоков данных и блок-схему данной программы.

Задание 5. Миллионер мистер Блэк очень не любит инфляцию, которая ежегодно «съедает» 12% его состояния. Поэтому он вложил большую часть своих денег в акции компаний на Нью-Йоркской фондовой бирже, которые, как известно, в среднем растут быстрее инфляции и с каждым годом делают Блэка еще богаче. Но на самом деле курсы акций — вещь нестабильная, и поэтому иногда могут и падать (хотя в среднем растут). Мистера Блэка это категорически не устраивает, и он хочет написать программу, которая каждый день бы отслеживала текущую стоимость его инвестиционного портфеля и выдавала сигнал тревоги, если «сегодняшняя» стоимость оказывалась ниже «вчерашней». Собственно задание: программа считывает с клавиатуры, одно за другим, последовательность чисел i , как только обнаруживает, что последовательность перестала возрастать (или начала убывать), выводит на экран предупредительное сообщение и завершается. Указание: здесь удобно использовать оператор `break`.

Задание 6. Разработать программу, которая вычисляет сумму бесконечного ряда $\sum_{i=1}^{\infty} \frac{1}{i^2}$ с

заданной точностью $\varepsilon = 10^{-4}$. Считать, что требуемая точность достигнута, если вычислена сумма нескольких первых слагаемых и очередное слагаемое оказалось по модулю меньше, чем ε .

Задание 7. Напечатать числа в виде следующей таблицы. Указание: использовать вложенные циклы.

1	5 5 5 5 5
2 2	6 6 6 6
3 3 3	7 7 7
4 4 4 4	8 8
5 5 5 5 5	9

а)

б)

Задание 8. Выполнить свой вариант индивидуального задания №4 (см. Приложение).

Примеры выполнения

В качестве справочника по организации циклов в C/C++ можно использовать приведенные ниже программы. Первые три из них вычисляют сумму первых N натуральных чисел с помощью трех различных видов циклов.

1 Пример цикла *while*

```
#include <iostream>
using namespace std;

// препроцессорная обработка: везде в тексте программы вместо N
// подставляется значение 6
#define N 5
int main()
{
    // хранит текущее число
    int i = 1;
    // хранит текущее значение суммы
    int sum = 0;
    // пока не все числа просмотрены
    while (i <= N){
        // значение переменной sum увеличивается на текущее значение i
        sum = sum + i;
        // наращивание значений переменной i на 1
        i=i+1;
    }
    cout << sum << endl;
    return 0;
}
```

2 Пример цикла *do-while*

```
#include <iostream>
using namespace std;

int main()
{
    //описание целой константы вместо директивы препроцессора
    const int N = 5;
    int i = 1;
    int sum = 0;
    do{
        sum += i;
        i++;
    }while(i <= N);
    cout << sum << endl;
    return 0;
}
```

3 Пример цикла *for*

```
#include <iostream>
using namespace std;

#define N 5
```

```

int main()
{
    int i;
    int sum=0;
    for (i=1; i<=N; i++)
        sum+=i;
    cout << sum << endl;
    return 0;
}

```

4 Вложенные циклы

«Блок операторов», который составляет тело цикла, может быть любого, в том числе, довольно большого, размера. В частности, нет никаких ограничений на *использование внутри тела цикла еще одного или нескольких циклов*. Такие конструкции называются *вложенными циклами*.

В качестве примера рассмотрим фрагмент программы, выводящий на экран все возможные показания электронных часов вида «ЧЧ:ММ» в течение суток.

```

for(int hh=0; hh<24; hh++){
    for(int mm=0; mm<60; mm++){
        cout << hh << ":" << mm << "\n";
    } // mm
} // hh

```



Рис. 3. Пример вложенных циклов

Контрольные вопросы

1. Напишите общую форму записи операторов цикла (3 шт.) в языке C/C++.
2. Цикл — это многократное повторное выполнение некоторого фрагмента кода. А может ли цикл выполниться всего 1 раз? Ноль раз?
3. Возможно ли произвольный фрагмент программы, использующий цикл for, переписать с использованием цикла while?
4. Найдите ошибку в следующем фрагменте программы:

```
int x = 1;
int sum = 0;
while(x <= 4);
{
    sum += x;
    x++;
}
```

Если исправить ошибку: чему равно значение переменной sum после завершения цикла?

Как будет меняться значение этой переменной во время выполнения цикла?

5. При каком условии закончит выполняться цикл из программы на рис. 2? Как нужно изменить данную программу, чтобы цикл в ней завершался при вводе некоторого «магического» числа — например, 42?
6. Сколько *всего* раз выполнится тело внутреннего цикла на рис. 3?
7. Как изменится работа некоторой программы, если имеющиеся в ней вложенные циклы разместить не один внутри другого, а друг за другом?

Оценивание

Содержание отчета

1. Условия, исходные коды и скриншоты выполнения заданий 1-8.
2. Диаграмма потоков данных и нарисованная в RAPTOR'е блок-схема к заданию 3.
3. Диаграмма потоков данных и нарисованная *вручную* блок-схема к заданию 4.

Примечание: блок-схемы и диаграммы потоков данных можно рисовать в уже распечатанном отчете от руки. Не надо мучать Ворд непроизводительной тратой времени.

Баллы за задания

Задание	Баллы	
1	обязательное	1
2	обязательное	1
3	обязательное	1
4	обязательное	1
5	1	
6	1	
7	1	
8	1	
Всего	8	

Бонусы

Досрочная сдача: +1 балла.

Несвоевременная сдача: макс. балл уменьшается на 1 за каждую просроченную неделю.

Индивидуальное задание №4а

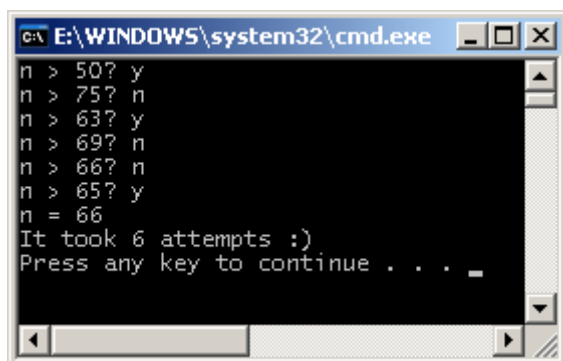
Если составление блок-схем алгоритмов не вызывает у Вас затруднений, и если Вы достаточно свободно обращаетесь с циклами, то вместо заданий 3-7 можно сделать индивидуальное задание №4а (см. Приложение). В этом случае оценивание будет следующее:

Задание	Баллы	
1	обязательное	1
2	обязательное	1
инд. задание 4а	6	
8	2	
Всего	10	

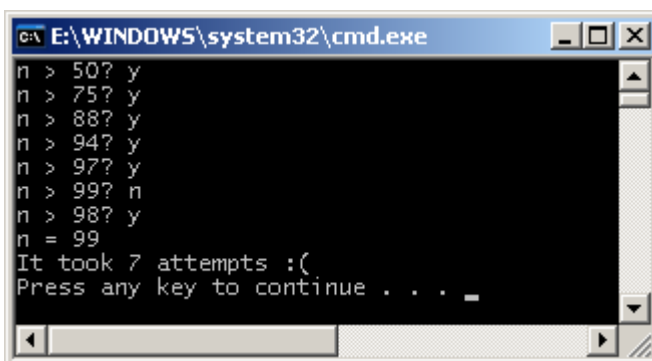
По поводу оценивания см. также «Критерии оценивания.doc».

Бинарный поиск

Напишите программу, которая задает пользователю серию несложных вопросов и «угадывает» таким образом задуманное им целое число от 1 до 100. Например:



```
C:\E:\WINDOWS\system32\cmd.exe
n > 50? y
n > 75? n
n > 63? y
n > 69? n
n > 66? n
n > 65? y
n = 66
It took 6 attempts :)
Press any key to continue . . . _
```



```
C:\E:\WINDOWS\system32\cmd.exe
n > 50? y
n > 75? y
n > 88? y
n > 94? y
n > 97? y
n > 99? n
n > 98? y
n = 99
It took 7 attempts :(
Press any key to continue . . . _
```


Цена задания — 1 балл.

Дополнительно: промоделируйте вышеуказанный процесс угадывания, переделав программу так, чтобы она «играла» сама с собой. Случайное число «загадывайте» с помощью функции `rand()` из библиотеки `stdlib`. Прodelав 1000 (можно больше) экспериментов, вычислите среднее число попыток, необходимых для успешного угадывания. Какое отношение это число имеет к величине $\log_2 100$? Цена задания — договорная.

Приложение: индивидуальные задания

Индивидуальное задание №4

1. Вводится последовательность из n целых чисел (n задается с клавиатуры). Найти, сколько в ней нулей.
2. Вводится последовательность из n целых чисел (n задается с клавиатуры). Найти наибольшее из отрицательных чисел.
3. Вводится последовательность из n целых чисел (n задается с клавиатуры). Найти два наименьших числа.
4. Вводится последовательность из n целых чисел (n задается с клавиатуры). Определить, сохраняют ли числа в последовательности знак.
5. Вводится последовательность из n целых чисел (n задается с клавиатуры). Проверить, имеются ли в последовательности два идущих подряд нулевых члена.
6. Вводится последовательность из n целых чисел (n задается с клавиатуры). Найти сумму чисел, меньших заданного числа A и их количество.
7. Вводится последовательность из n целых чисел (n задается с клавиатуры). Найти среднее арифметическое всех членов последовательности, кроме i -го (i — задается с клавиатуры).
8. Вводится последовательность из n целых чисел (n задается с клавиатуры). Верно ли, что отрицательных членов в последовательности больше, чем положительных?
9. Вводится последовательность из n целых чисел (n задается с клавиатуры). Верно ли, что наибольший член последовательности больше 1?
10. Вводится последовательность из n целых чисел (n задается с клавиатуры). Найти наименьшее число.
11. Вводится последовательность из n целых чисел (n задается с клавиатуры). Определить, содержит ли последовательность хотя бы два равных соседних числа.
12. Вводится последовательность из n целых чисел (n задается с клавиатуры). Определить, является ли последовательность знакопеременной (т.е. любые два соседних элемента имеют разный знак).
13. Вводится последовательность из n целых чисел (n задается с клавиатуры). Определить, сколько в последовательности пар равных соседних элементов.
14. Вводится последовательность из n целых чисел (n задается с клавиатуры). Найти количество чисел, больших обоих своих соседей (локальных максимумов).
15. Вводится последовательность из n целых чисел (n задается с клавиатуры). Проверить, упорядочена ли последовательность. По возрастанию или по убыванию?
16. Вводится последовательность из n целых чисел (n задается с клавиатуры). Посчитать, сколько в последовательности отрицательных чисел и найти сумму положительных чисел.
17. Дано натуральное число n . Переставить местами первую и последнюю цифры числа n .
18. Дано натуральное число n . Приписать по 1 цифре в начало и в конец записи числа n .
19. Дано натуральное число n . Проверить, упорядочены ли цифры по возрастанию.
20. Дано натуральное число n . Напечатать в обратном порядке цифры числа n .
21. Дано натуральное число n . Перенести в конец записи числа n его первую цифру.
22. Дано натуральное число n . Переставить i -ю и k -ю цифры числа n .
23. Дано натуральное число n . Включить новое число $x1$ в запись числа n перед k -ой цифрой. $x1$ может содержать более одной цифры.
24. Дано натуральное число n . Удалить из записи максимальную цифру.
25. Дано натуральное число n . Удалить k -ую цифру из записи числа n .
26. Дано натуральное число n . Удалить из записи числа n все цифры, меньшие среднего арифметического.
27. Дано натуральное число n . Перенести в начало записи числа n его последнюю цифру.
28. Дано натуральное число n . Проверить, упорядочены ли цифры по убыванию.
29. Дано натуральное число n . Удалить из записи минимальную цифру.
30. Дано натуральное число n . Получить все его натуральные делители.

*Индивидуальное задание №4а

(бонусное)

1. Найти все натуральные числа, не превосходящие заданного N и делящиеся на каждую из своих цифр.
2. Найти все натуральные числа, не превосходящие заданного N и равные сумме кубов своих цифр.
3. Даны натуральные числа m и n . Получить все меньшие n натуральные числа, квадрат суммы цифр которых равен m .
4. Дано натуральное число n . Среди чисел $1, 2, \dots, n$, найти автоморфные, т. е. такие, запись которых совпадает с последними цифрами записи их квадрата ($6^2 = 36, 25^2 = 625$).
5. Найти натуральное число из интервала $[1, n]$ с максимальной суммой делителей.
6. Натуральное число называется совершенным, если оно равно сумме всех своих делителей, за исключением самого себя.
 - а. Число 6 совершенное, т.к. $6 = 1 + 2 + 3$
 - б. Число 8 не совершенное, т.к. $8 \neq 1 + 2 + 4$
- Дано натуральное n . Получить все совершенные числа, меньше n .
7. Два натуральных числа называют дружественными, если каждое из них равно сумме всех делителей другого, кроме самого этого числа. Найти все пары дружественных чисел, лежащих в диапазоне от A до B .
8. Дано натуральное число n . Выяснить, имеются ли среди чисел $n, n+1, \dots, 2n$ близнецы, т. е. простые числа, разность между которыми равна двум.
9. Напечатать все представления натурального числа N суммой двух натуральных чисел. Перестановка слагаемых нового способа не дает.
10. Дано натуральное число N ($N \geq 3$). Получить все тройки натуральных чисел x_1, x_2, x_3 такие, что $x_1 \geq x_2 \geq x_3$ и $x_1 + x_2 + x_3 = n$.
11. Дано натуральное число N . Получить все Пифагоровы тройки натуральных чисел, каждое из которых не превосходит n , т. е. все такие тройки натуральных чисел a, b, c , что $a^2 + b^2 = c^2$ ($a \leq b \leq c \leq N$).
12. Найти все натуральные числа, не превосходящие заданного N и представимые в виде суммы квадратов двух каких-либо различных натуральных чисел.
13. Найти наименьшее натуральное число N , представимое двумя различными способами в виде суммы кубов двух натуральных чисел: $N = x^3 + y^3$ ($x \geq y$).
14. Дано натуральное число n . Найти все меньшие n числа Мерсена. (Простое число называется числом Мерсена, если оно может быть представлено в виде $2^p - 1$, где p – тоже простое число).
15. Натуральное число называется палиндромом, если его запись читается одинаково с начала и с конца (как, например, 4884, 393, 1). Найти все меньшие 100 натуральных числа, которые при возведении в квадрат дают палиндром.