Лабораторная работа № 4. Разработка программ циклической структуры

Цель работы: научиться разрабатывать программы алгоритмов циклической структуры

Теоретическое обоснование

Цикл - это многократное выполнение одинаковых действий. Доказано, что любой алгоритм может быть записан с помощью трех алгоритмических конструкций: циклов, условных операторов и последовательного выполнения команд (линейных алгоритмов). Вся суть цикла - это повторять до тех пор, пока некоторое условие не становится ложно.

Цикл с условием

Рассмотрим следующую задачу: определить количество цифр в десятичной записи целого положительного числа. Будем предполагать, что исходное число записано в переменную n целого типа. Сначала составим алгоритм решения этой задачи. Чтобы считать что-то в программе, нужно использовать переменную, которую называют счетчиком. Для подсчета количества цифр необходимо как-то отсекать эти цифры 20 по одной, с начала или с конца, каждый раз увеличивая счетчик. Начальное значение счетчика должно быть равно нулю, так как до выполнения алгоритма еще не найдено ни одно цифры. Отсечь последнюю цифру проще - достаточно разделить число нацело на 10 (поскольку речь идет о десятичной системе). Операции отсечения и увеличения счетчика нужно выполнять столько раз, сколько цифр в числе. Как же поймать момент, когда цифры кончатся? Несложно понять, что в этом случае результат очередного деления на 10 будет равен нулю, это и говорит о том, что отброшена последняя оставшаяся цифра. Изменение переменной n и счетчика для начального значения 1234 можно записать в виде таблицы. Запись такого цикла на языке C++ выглядит так:

```
count =0 ;
```

```
while ( n>0 )
{
        n=n \ 1 0 ;
        count ++ ;
}
```

Цикл с переменной.

Здесь целочисленная переменная-счетчик имеет имя count. Слово while переводится с английского как пока, за ним в скобках записывается условие работы цикла (в данном случае - пока n>0). Фигурные скобки ограничивают составной оператор. Если в теле цикла нужно выполнить только один оператор, эти скобки можно не ставить. Если проверка условия выполняется в начале очередного шага цикла. Такой цикл называется циклом с предусловием (то есть с предварительной проверкой условия) или циклом пока.

Если в начальный момент значение переменной n будет нулевой или отрицательное, цикл не выполнится ни одного раза. В данном случае количество шагов цикла пока неизвестно, оно равно количеству цифр введенного числа, то есть зависит от исходных данных. Кроме того, этот же цикл может быть использован и в том случае, когда число шагов известно заранее или может быть вычислено:

```
k =0 ;
 while ( k<10)
{
        cout<< привет \n " ;
        k ++ ;
}
```

Если условие в заголовке цикла никогда не нарушится, цикл будет работать бесконечно долго. В этом случае говорят, что программа зациклилась. Например, если забыть увеличить переменную k в предыдущем цикле, программа зациклится:

```
k =0 ;
 while ( k<10)
{
        cout<< привет \n " ;
}
```

Во многих языках программирования существует цикл с постусловием, в котором условие проверяется после завершения очередного шага цикла. Это полезно в том случае, когда нужно обязательно выполнить цикл хотя бы один раз. Например, пользователь должен ввести с клавиатуры положительное число. Для того, чтобы защитить программу от неверных входных данных, можно использовать такой цикл с постусловием:

```
do {
 cout << " Введите n>0 : " ;
 cin >>n ;
}
while ( n < = 0 ) ;
```

Этот цикл закончится тогда, когда условие n<=0 нарушится, то есть станет истинным условие n>0. А это значит, что пользователь ввел допустимое (положительное) значение. Обратите внимание на важную особенность этого вида цикла: при входе в цикл условие не проверяется, поэтому цикл всегда выполняется хотя бы один раз.

Вложенные циклы

В более сложных задачах часто бывает так, что на каждом шаге цикла нужно выполнять обработку данных, которая также представляет собой циклический алгоритм. В этом случае получается конструкция цикл в цикле или вложенный цикл.

Предположим, что нужно найти все простые числа в интервале от 2 до 1000. Простейший (но не самый быстрый) алгоритм решения такой задачи на

псевдокоде выглядит так: сделать для n от 1 до 1000 если число n простое то вывод n Чтобы проверить делимость числа n на некоторое число k, нужно взять остаток от деления n на k. Если этот остаток равен нулю, то n делится на k. Таким образом, программу можно записать так (здесь n, k и count - целочисленные переменные, count обозначает счетчик делителей):

```
for ( n=2 ; n<=1000 ; n++ )
{
    count =0 ;
    for ( k =2 ; k<n; k++)
    if (n%k==0) count++;
    if (count==0) count<<n<<endl;
}
```

Преинкремент и постинкремент

Разберём различия инструкций преинкремента (++i) и постинкремента (i++).

Приведём два примера:

```
int a, b;
b = 10;
a = b++;
cout << a << " " << b << endl;
```

На экран будет выведено два числа — 10 и 11.

```
int a, b;
b = 10;
a = ++b;
cout << a << " " << b << endl;
```

На экран будет выведено два числа - 11 и 11.

Таким образом, преинкремент сначала прибавляет единицу к переменной b, а затем возвращает её значение. Постинкремент, наоборот, сначала возвращает значение переменной b, а затем прибавляет к ней единицу.

Такое поведение называется **побочным эффектом операций** и часто позволяет сократить длину кода. Однако, мы не рекомендуем пользоваться данным приёмом, так как это может привести к ошибкам в программе

и усложнить чтение кода. Советуем не использовать преинкремент и постинкремент в присваиваниях.

Методика выполнения работы

1. Разберите и выполните следующие задачи

Задача 1. Задача про сумму чисел от 1 до 100

Вывести на экран все числа от 11 до 100100 и их сумму.

Решение

```
#include <iostream>
using namespace std;
int main()
{
    int i, s;
    i = 1;
    s = 0;
    while (i <= 100){
        cout << i << " ";
        s = s + i;
        i = i + 1;
    }
    cout << s << endl;
    return 0;
}
```

В данном примере мы выполняем действия внутри цикла while до тех пор, пока логическое выражение i<=100 истинно.

Важно при написании цикла всегда изменять переменную, которая используется в условии продолжения цикла, так как в противном случае цикл будет выполняться бесконечно долго.

Задача 2. Задача про степень двойки

Найдём наибольшую степень двойки, которая не превосходит 1000.

Решение

```
#include <iostream>
using namespace std;
int main()
{
    int ans = 1;
    while (ans * 2 < 1000){
        ans = ans * 2;
```

```
    }
    cout << ans;
    return 0;
}
```

Заметим, что если мы напишем в условии выхода из цикла ans<1000, то мы получим первую степень двойки, большую или равную 1000.

Задача 3. Задача про наибольшее число в последовательности

Дана последовательность чисел, необходимо найти самое большое число в последовательности. Признаком завершения последовательности является число 0.

Решение

```
#include <iostream>
using namespace std;
int main()
{
    int x, ma = 0;
    cin >> x;
    while (x != 0){
        if (x > ma){
            ma = x;
        }
        cin >> x;
    }
    cout << ma << endl;
    return 0;
}
```

Задача 4. Задача про цифры числа

Дано число x. Необходимо посчитать количество и сумму его цифр.

Решение

```
#include <iostream>
using namespace std;
int main()
{
    int x, cnt = 0, s = 0;
    cin >> x;
    while (x > 0){
        cnt = cnt + 1;
        s = s + x % 10;
        x = x / 10;
    }
    cout << cnt << " " << s << endl;
    return 0;
}
```

Заметим, что программа даст неправильный ответ для числа 00 (в его записи присутствует одна цифра). Этот случай нужно рассмотреть отдельно.

Задача 5. Задача про наибольшее число в последовательности

Дана последовательнось чисел. Необходимо найти самое большое число в последовательности. Признаком завершения последовательности является число 0.

Мы уже рассматривали похожую задачу ранее, однако сейчас мы решим её иначе, не опираясь на информацию о том, что нам известны ограничения на значения чисел в последовательности.

Решение

```
#include <iostream>
using namespace std;
int main()
{
    int maxx, now;
    cin >> now;
    maxx = now;
    while (now > 0){
        if (now > maxx){
            maxx = now;
        }
        cin >> now;
    }
    cout << maxx << endl;
    return 0;
}
```

Задача 6. Задача про количество максимумов в последовательности

Дана последовательнось чисел. Необходимо найти максимум в последовательности и количество элементов, значение которых равно максимуму. Признаком завершения последовательности является число 0.

```
#include <iostream>
using namespace std;
int main()
{
    int maxx, now, cntmax = 0;
    cin >> now;
    maxx = now;
    while (now > 0){
        if (now > maxx){
            maxx = now;
            cntmax = 1;
```

```
        } else if (now == maxx){
            cntmax = cntmax + 1;
        }
        cin >> now;
    }
    cout << maxx << " " << cntmax << endl;
    return 0;
}
```

Задача 7. Бесконечный цикл

Если в реализации программы допустить ошибку, то цикл может работать бесконечно долго. Рассмотрим это на примере задачи, в которой требуется вывести все числа от 1 до n.

Решение

```
#include <iostream>
using namespace std;
int main()
{
    int n;
    cin >> n;
    int i = 1;
    while (i <= n){
        cout << i << " ";
    }
    return 0;
}
```

Данный код будет бесконечно долго выводить единицы. Несложно заметить, что в данной программе мы забываем изменять значение счётчика i внутри цикла. Добавим необходимую команду и получим верное решение:

Решение

```
#include <iostream>
using namespace std;
int main()
{
    int n;
    cin >> n;
    int i = 1;
    while (i <= n){
        cout << i << " ";
        i = i + 1;
    }
    return 0;
}
```

Задача 8. Задача о локальных максимумах

Элемент последовательности называется локальным максимумом, если он строго больше предыдущего и последующего элементов последовательности. Первый и последний элементы последовательности не являются локальными максимумами.

Дана последовательность натуральных чисел, завершающаяся числом 0. Определите количество строгих локальных максимумов в этой последовательности.

Решение

```
#include <iostream>
using namespace std;
int main()
{
    int prev, now, next, cnt = 0;
    cin >> prev;
    cin >> now;
    cin >> next;
    while (next > 0){
        if (now > prev && now > next){
            cnt = cnt + 1;
        }
        prev = now;
        now = next;
        cin >> next;
    }
    cout << cnt << endl;
    return 0;
}
```

Познакомимся с инструкциями *break* и *continue*.

Задача 9. Задача о поиске числа

Дана последовательность и число x. Необходимо вывести номер позиции, в которой число x первый раз встречается в последовательности. Если число x не присутствует в последовательности, то необходимо вывести количество элементов в ней. Признаком завершения последовательности является число 0.

Решение

Для решения задачи используем команду *break*. Данная инструкция применяется только внутри цикла. Если она запустилась, то выполнение цикла немедленно останавливается, и программа переходит на первую инструкцию после цикла.

```
#include <iostream>
using namespace std;
int main()
{
    int x, now, i = 0;
    cin >> x;
    cin >> now;
    while (now > 0){
        if (now == x){
            break;
        }
        i = i + 1;
        cin >> now;
    }
    cout << i;
    return 0;
}
```

Инструкция *break* сильно затрудняет чтение кода, поэтому использовать её не рекомендуется.

Задача 10. Задача о чётных числах

Дана последовательность. Необходимо вывести сумму всех чётных элементов последовательности. Признаком завершения последовательности является число 0.

Решение

Для решения данной задачи будем использовать инструкцию *continue*. Данная инструкция применяется только внутри цикла. Если она запустилась, то выполнение цикла немедленно переходит на начало следующей итерации.

```
#include <iostream>
using namespace std;
int main()
{
    int now, summ = 0;
    cin >> now;
    while (now > 0){
        if (now % 2 != 0){
            cin >> now;
            continue;
        }
```

```
        summ = summ + now;
        cin >> now;
    }
    cout << summ << endl;
    return 0;
}
```

Заметим, что в данной задаче можно написать более понятное и простое решение без инструкции *continue*. Инструкция *continue* сильно затрудняет чтение кода, поэтому использовать её не рекомендуется.

Некоторые операции в языке C++ имеют сокращённую запись. Например, увеличить x на единицу можно несколькими способами:

x=x+1;

x+=1;

++x;

x++;

Последние два варианта самые компактные, поэтому рекомендуем использовать именно их. Операции ++x и x++ имеют небольшое различие, но пока мы его рассматривать не будем.

Задача 11. Задача про числа от 11 до 100100

Выведите на экран все числа от 11 до 100100.

Решение с циклом while

```
while (i <= 100){
    cout << i << " ";
    i++;
}
```

Данную задачу можно решить иначе с помощью цикла for.

Решение с помощью цикла for

```
for (i = 1; i <= 100; ++i)
    cout << i << " ";
```

Данное решение лучше, так как все инструкции управления циклом находятся в одном месте, что упрощает чтение программы.

Цикл for содержит три инструкции управления, разделённые точкой с запятой:

1. Инициализация переменной.

2. Условие продолжения цикла.

3. Инструкция, выполняемая после завершения всех операций внутри цикла (обычно изменение счётчика).

12. Задача про таблицу умножения

Выведите на экран таблицу умножения для чисел от 1 до 10.

Решение

Решим данную задачу с помощью вложенного цикла for.

```
for (i = 1; i <= 10; ++i){
    for (int j = 1; j <= 10; ++j){
        cout << i * j << " ";
    }
    cout << endl;
}
```

Иногда может быть важно не выводить пробел после последнего числа в строке. Модифицируем код, чтобы учесть это требование:

```
for (i = 1; i <= 10; ++i){
    for (int j = 1; j <= 9; ++j){
        cout << i * j << " ";
    }
    cout << i * 10 << endl;
}
```

Модифицируем программу так, чтобы не выводить перевод строки после вывода таблицы:

```
for (i = 1; i <= 10; ++i){
    for (int j = 1; j <= 10; ++j){
        cout << i * j << " ";
    }
    if (i != 10){
        cout << endl;
    }
}
```

13. Задача

Дана последовательность из n натуральных чисел. Найдите сумму чётных чисел в последовательности. Значения всех чисел не превышают $2^{31}-1$.

Решение

Несмотря на то, что для хранения элементов последовательности хватит типа *int*, для хранения ответа необходимо использовать больший тип данных — *long long*.

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

int main() {
    long long sum = 0;
    int now, n;
    cin >> n;
    for (int i = 0; i < n; ++i){
        cin >> now;
        if  (now % 2 == 0){
            sum += now;
        }
    }
    cout << sum << endl;
    return 0;
}
```

14. Задача

Даны два натуральных числа - A и B. Необходимо вывести все нечётные числа на отрезке [A;B]. Гарантируется, что A - нечётное и A<B.

Решение

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

int main() {
    int a, b;
    cin >> a >> b;
    for (int i = a; i <= b; i += 2){
        cout << i << " ";
    }
    return 0;
}
```

15. Задача

Выведите все n-значные натуральные числа.

Решение

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

int main()
{
    int d;
    cin >> d;
    int mind = 1, maxd;
    for (int i = 1; i < d ; ++i){
        mind *= 10;
    }
    maxd = mind * 10 - 1;
```

```
    for (int i = mind; i <= maxd; ++i){
        cout << i << " ";
    }
    return 0;
}
```

15. Задача

Даны три натуральных числа - a, b, c. Выведите все числа на отрезке от [a;b], делящиеся на c.

Решение

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

int main()
{
    int a, b, c;
    cin >> a >> b >> c;
    int m = a % c;
    if (m != 0){
        a += c - m;
    }
    for (int i = a; i <= b; i += c){
        cout << i << " ";
    }
    cout << endl;
    return 0;
}
```

4. Самостоятельно разработать программы к следующим задачам по вариантам

Задание 1. Разработать программу на языке программирования C++ по вариантам

Вариант	Условие задачи	Входные данные	Выходные данные
1	**Цифры числа** Дано 10-значное число. Выведите все цифры этого числа в обратном порядке по одной. **Входные данные** На вход подаётся натуральное 10-значное число. **Выходные данные** Выведите ответ на задачу. В качестве разделителя между цифрами можно использовать переводы строки и пробелы.	1234567890	0 9 8 7 6 5 4 3 2 1
2	**Диофантово уравнение**	1 -1	1

	Даны числа a, b, c, d. Выведите в порядке возрастания все целые числа от 0 до 1000 включительно, которые являются корнями уравнения $a*x^3+b*x^2+c*x+d=0$. **Входные данные** Вводятся целые числа a, b, c и d. Все числа не превосходят по модулю 30000. **Выходные данные** Выведите ответ на задачу. Если в указанном промежутке нет корней уравнения, то ничего выводить не нужно.	1 -1 1 1 1 1	
3	**Остатки**. Даны целые неотрицательные числа a, b, c, d, при этом 0≤c< и т.д.), должен быть только один цикл. **Входные данные** На вход подаются четыре строки, в каждой из которых написано по одному неотрицательному целому числу - a, b, c, d. Все числа не превосходят $2*10^9$, 0≤c<d. **Выходные данные** Выведите ответ на задачу.	2 5 0 2 5 5 0 5	2 4 5
4	**Четные числа** По данным двум натуральным числам A и B (A≤B) выведите все чётные числа на отрезке от A до B. В этой задаче нельзя использовать инструкцию if. **Входные данные** Вводятся два натуральных числа A и B. **Выходные данные** Выведите ответ на задачу	1 10	2 4 6 8 10
5	**Делители** По данному натуральному числу выведите все его натуральные делители в порядке возрастания. **Входные данные** На вход подаётся единственное натуральное число n (n≤1000). **Выходные данные** Выведите все делители числа в порядке возрастания. Делители можно выводить на одной строке, разделяя пробелом, или на разных строках.	10	1 2 5 10
6	**Лесенка** По данному натуральному числу n выведите лесенку из n ступенек, i-я ступенька состоит из чисел от 1 до i без пробелов. **Входные данные** На вход подаётся натуральное число n (n≤9). **Выходные данные** Выведите ответ на задачу	3	1 12 123
7	**Сумма произведений соседних чисел** По заданной последовательности a1, a2, ……, an чисел вычислите сумму $a1*a2+a2*a3+\cdots+an-1*an$ **Входные данные** Первая строка входных данных содержит число n≥2. В следующих n строках вводится по одному числу. В i+1 строке содержится значение i-того элемента последовательности. Все числа во входном файле натуральные, не превосходящие 100. **Выходные данные** Выведите ответ на задачу.	4 2 3 1 5	14
8	**Среднее значение последовательности** Определите среднее значение всех элементов последовательности, завершающейся числом 0. Сам ноль в последовательность не входит. Использовать массивы в данной задаче нельзя. **Входные данные** Вводится последовательность целых чисел, оканчивающаяся числом 0 (само число 0 в последовательность не входит, а служит как признак её окончания. **Выходные данные** Выведите ответ на задачу.	1 7 9 0	5.66666666667
9	**Количество локальных максимумов**	1	2

	Элемент последовательности называется строгим локальным максимумом, если он строго больше предыдущего и последующего элементов последовательности. Первый и последний элемент последовательности не являются локальными максимумами. **Входные данные** Дана последовательность натуральных чисел, завершающаяся числом 0. Гарантируется, что все числа не превосходят 100, а также, что в последовательности есть хотя бы три элемента. **Выходные данные** Определите количество строгих локальных максимумов в этой последовательности.	2 1 2 1 0	
10	**Количество элементов, которые больше предыдущего** Последовательность состоит из натуральных чисел и завершается числом 0. Определите, сколько элементов этой последовательности больше предыдущего элемента. **Входные данные** Вводится последовательность натуральных чисел, оканчивающаяся числом 0 (само число 0 в последовательность не входит, а служит как признак её окончания). **Выходные данные** Выведите ответ на задачу.	1 2 1 2 1 0	2

Задание 2.

Вариант	Условие задачи
1	Составить программу расчета таблицы значений функции: Y = 5(1-e-0,5t)cos(2π t), t >= 0 в интервале a$\leq$t$\leq$b в n равностоящих точках. Границы интервала [a,b] и количество точек n ввести с клавиатуры. Результаты вывести на экран.
2	Составить программу расчета таблицы значений функции Y=e-2tsin22pt, t$\geq$0 в интервале a$\leq$t$\leq$b в n равностоящих точках. Границы интервала [a,b] и количество точек n ввести с клавиатуры. Результаты вывести на экран.
3	Составить программу расчета таблицы значений функции y= cos(2t)+e^{t}, t$\geq$0 в интервале a$\leq$t$\leq$b в n равностоящих точках. Границы интервала [a,b] и количество точек n ввести с клавиатуры. Результаты вывести на экран.
4	Составить программу расчета таблицы значений функции Y=2(3-$e^{-0,5t}$)sin(2π t), t>0, a<=t<=b в интервале a$\leq$t$\leq$b в n равностоящих точках. Границы интервала [a,b] и количество точек n ввести с клавиатуры

	Найти количество всех отрицательных значений функции в расчетных точках. Результаты вывести на экран.
5	Составить программу расчета таблицы значений функции y=(1-sin 2π t)e^t, t>=0 в интервале a$\leq$t$\leq$b в n равностоящих точках. Границы интервала [a,b] и количество точек n ввести с клавиатуры. Результаты вывести на экран.
6	Составить программу расчета таблицы значений функции $Y = 4e^{-0.5t} * \cos t, t \geq 0$ в интервале a$\leq$t$\leq$b в n равностоящих точках. Границы интервала [a,b] и количество точек n ввести с клавиатуры. Результаты вывести на экран.
7	Составить программу расчета таблицы значений функции y= 5cos(2t)+e^t, t$\geq$0 в интервале a$\leq$t$\leq$b в n равностоящих точках. Границы интервала [a,b] и количество точек n ввести с клавиатуры. Результаты вывести на экран.
8	Составить программу расчета таблицы значений функции f(x)=$\sqrt{x}$ +e^{-x} cos x; на интервале a<=x<=b в n равностоящих точках. Границы интервала [a,b] и количество точек ввести с клавиатуры. Результаты вывести на экран.
9	Составить программу расчета таблицы значений функции $Y = 1 + \frac{\sin 2\pi t}{1+t}; t >= 0$ в интервале a$\leq$t$\leq$b в n равностоящих точках. Границы интервала [a,b] и количество точек n ввести с клавиатуры. Результаты вывести на экран.
10	Составить программу расчета таблицы значений функции $y = t * \sin 2t$, t>=0 в интервале a$\leq$t$\leq$b в n равностоящих точках. Границы интервала [a,b] и количество точек n ввести с клавиатуры. Результаты вывести на экран.

Задание 3. (Задание повышенной трудности)

Вариант	Условие задачи	Входные данные	Выходные данные

1	Определите наименьшее расстояние между двумя локальными максимумами последовательности натуральных чисел, завершающейся числом 0. Если в последовательности нет двух локальных максимумов, выведите число 0. Начальное и конечное значение при этом локальными максимумами не считаются. Расстоянием считается количество пробелов между элементами. **Формат входных данных** Вводится последовательность целых чисел, оканчивающаяся числом 0 (само число 0 в последовательность не входит, а служит как признак ее окончания). **Формат выходных данных** Выведите ответ на задачу.	Sample Input 1: 1 2 1 1 2 1 2 1 0 Sample Input 2: 1 2 3 0	Sample Output 1: 2
2	Элемент последовательности называется локальным максимумом, если он строго больше предыдущего и последующего элемента последовательности. Первый и последний элемент последовательности не являются локальными максимумами. Дана последовательность натуральных чисел, завершающаяся числом 0. Определите количество строгих локальных максимумов в этой последовательности. **Формат входных данных** Вводится последовательность натуральных чисел, оканчивающаяся числом 0 (само число 0 в последовательность не входит, а служит как признак ее окончания). **Формат выходных данных** Выведите ответ на задачу.	Sample Input: 1 2 1 2 1 0	Sample Output: 2
3	Дана последовательность натуральных чисел, завершающаяся числом 0. Определите, какое	Sample Input:	Sample Output:

	наибольшее число подряд идущих элементов этой последовательности равны друг другу. Если не нашлось ни одной пары, тройки и т.д. элементов, равных друг другу, то программа должна вывести число 1. **Формат входных данных** Вводится последовательность натуральных чисел, оканчивающаяся числом 0 (само число 0 в последовательность не входит, а служит как признак ее окончания). **Формат выходных данных** Выведите ответ на задачу.	1 7 7 9 1 0	2
4	Последовательность Фибоначчи определяется так: F(0) = 0, F(1) = 1, …, F(n) = F(n−1) + F(n−2). Дано натуральное число A. Определите, каким по счету числом Фибоначчи оно является, то есть выведите такое число N, что F(N) = A. Если A не является числом Фибоначчи, выведите число -1. **Формат входных данных** Вводится натуральное число A > 1. **Формат выходных данных** Выведите ответ на задачу.	Sample Input: 8	Sample Output: 6
5	Последовательность Фибоначчи определяется так: F(0) = 0, F(1) = 1, …, F(n) = F(n−1) + F(n−2). По данному числу N определите N-е число Фибоначчи F(N). **Формат входных данных** Вводится натуральное число N. **Формат выходных данных** Выведите ответ на задачу.	Sample Input: 6	Sample Output: 8
6	Последовательность состоит из натуральных чисел и завершается числом 0. Определите значение второго по величине элемента в этой последовательности, то есть элемента, который будет наибольшим, если из	Sample Input 1: 4 4 2 3	Sample Output 1: 4

	последовательности удалить наибольший элемент. **Формат входных данных** Вводится последовательность целых чисел, оканчивающаяся числом 0 (само число 0 в последовательность не входит, а служит как признак ее окончания). **Формат выходных данных** Выведите ответ на задачу.	0 Sample Input 2: 2 1 0	Sample Output 2: 1
7	Последовательность состоит из натуральных чисел и завершается числом 0. Определите, какое количество элементов этой последовательности, равны ее наибольшему элементу. **Формат входных данных** Вводится непустая последовательность целых чисел, оканчивающаяся числом 0 (само число 0 в последовательность не входит, а служит как признак ее окончания). **Формат выходных данных** Выведите ответ на задачу.	Sample Input 1: 1 7 9 0 Sample Input 2: 1 3 3 1 0	Sample Output 1: 1 Sample Output 2: 2
8	Последовательность состоит из натуральных чисел и завершается числом 0. Определите значение второго минимального по величине элемента в этой последовательности, то есть элемента, который будет наименьшим, если из последовательности удалить наименьший элемент. Последнее число 0 не учитывается. Гарантируется, что в последовательности есть хотя бы два элемента (кроме завершающего числа 0). **Входные данные** На вход подаётся последовательность целых неотрицательных чисел, заканчивающаяся нулём. Все числа в последовательности неотрицательные, по значению не превосходящие 10^9. **Выходные данные** Выведите ответ на задачу.	Sample Input 1: 1 7 9 0 Sample Input 2: 3 2 2 1 1 0	Sample Output 1: 7 Sample Output 2: 1

9	Дано натуральное число N. Выведите слово YES, если число N является точной степенью двойки, или слово NO в противном случае. **Формат входных данных** Вводится натуральное число. **Формат выходных данных** Выведите ответ на задачу.	Sample Input 1: 1 YES Sample Input 2: 2	Sample Output 1: Sample Output 2:
10	Программа получает на вход последовательность целых неотрицательных чисел, каждое число записано в отдельной строке. Последовательность завершается числом 0, при считывании которого программа должна закончить свою работу и вывести количество членов последовательности (не считая завершающего числа 0). Числа, следующие за числом 0, считывать не нужно. **Формат входных данных** Вводится последовательность целых чисел, заканчивающаяся числом 0. **Формат выходных данных** Выведите ответ на задачу.	Sample Input: 1 7 9 0 5	Sample Output: 3

Контрольные вопросы

1. Какой алгоритм называется циклическим?
2. Какие типы алгоритмов существуют? Опишите принцип их работы?
3. Опишите операторы цикла на C++
4. Чем преинкремент отличается от постинкремента?