Методические указания к контрольной работе

Создание простейших программ на языке программирования С++

Цель контрольной работы

1. Освоение структуры программы на языке С++ с использованием управляющих конструкций.
2. Получение навыков в организации ввода-вывода значений стандартных типов данных
3. Приобретение навыков в записи выражений на языке С++ и использовании стандартных функций.

Постановка задачи

1. Написать на языке С++ программу циклической обработки данных.

Содержание отчета

1. Постановка задачи для конкретного варианта.
2. Алгоритм решения задачи.
3. Текст программы и результаты ее выполнения.

Методические указания

1. Ввод-вывод данных

В языке С++ производится ввод-вывод *потоков* байтов. Поток - это просто последовательность байтов. В операциях ввода байты пересылаются от устройства (например, от клавиатуры, дисковода или соединения сети) в оперативную память. При выводе байты пересылаются из оперативной памяти на устройства (например, на экран дисплея, принтер, дисковод или в соединение сети).

Вывод данных.

Вывод в потоки может быть выполнен с помощью операции поместить в поток, т.е. перегруженной операции <<. Операция << перегружена для вывода элементов данных встроенных типов, для вывода строк и вывода значений указателей. В двух программах показан вывод строки, использующий одну операцию поместить в поток, также многократное использование операции поместить в поток. Выполнение обеих программ даст аналогичный результат.

```
// Вывод строки операцией поместить в поток
# include <iostream>
using namespace std;
int main ( )
{
    cout << "Добро пожаловать в мир С++!\n";
    return 0;
}

// Вывод строки двумя операциями поместить в поток
# include <iostream>
using namespace std;
int main( )
{
    cout << "Добро пожаловать в ";
    cout << "мир С++  !\n";
    return 0;
}
```

Переход на новую строку, осуществленный в этих программах с помощью управляющей последовательности \n, можно осуществить и с помощью *манипулятора потока* endl (end line – конец строки). Манипулятор потока endl вызывает переход на новую строку и кроме того приводит к сбросу буфера вывода (т.е. заставляет буфер немедленно вывести данные, даже если он полностью не заполнен). Сброс буфера вывода может быть также выполнен оператором: cout << flush;

```
// Использование манипулятора потока endl
# include <iostream>
using namespace std;
int main ( )
{
    cout << "Добро пожаловать в ";
    cout << "мир C++!";
    cout << endl;        // манипулятор потока конец строки
    return 0;
}
// Вывод значения выражения
# include <iostream>
using namespace std;
int main ( )
{
    cout << "47 плюс 53 равняется ";
    cout <<(47 + 53);  // выражение
    cout << endl;
    return 0;
}
```

Печать целых и дробных чисел

Операция поместить в поток (<<) языка C++ является достаточно «интеллектуальной», чтобы определить тип переменной (полагая, что она должным образом объявлена); поэтому для использования операции поместить в поток никакой дополнительной информации не требуется.

```
# include <iostream>
using namespace std;
int main ( )
{
    cout<<"Целое число - "<< 3<<endl;
    cout<<"Дробное число - "<< 3.1415<<endl;
    return 0;
}
```

Сцепление операций поместить в поток и взять из потока

Каждая из перегруженных операций << и >> может быть использована в *сцепленной форме*.

Многократные операции поместить в поток выполняются в той последовательности, в которой они записаны:

(((cout <<"47 плюс 53 равняется ")<<(47 + 53))<< endl);

т.е. операция поместить в поток << имеет ассоциативность слева направо. Такой способ сцепления операций поместить в поток возможен, поскольку перегруженная операция << возвращает ссылку на объект своего левого операнда, т.е. на объект cout. Таким образом, самое левое выражение в скобках

(cout << "47 плюс 53 равно")

выводит заданную строку символов и возвращает ссылку на cout. Это приводит к тому, что среднее выражение в скобках должно выполняться как

(cout<<(47 + 53))

выводить целое значение 100 и возвращать ссылку на cout. Затем выполняется самое правое выражение в скобках как

cout << endl;

которое переводит строку, сбрасывает cout и возвращает ссылку на cout. Эта ссылка в данном случае не используется.

```
// Сцепление перегруженных операций поместить в поток <<.
# include <iostream>
using namespace std;
int main ( )
{
    cout << "47 плюс 53 равняется " << (47 + 53) << endl;
    return 0;
}
```

Печать строк и символов

При написании программы в стиле языка С при вводе-выводе программист должен давать информацию о типе данных. В С++ типы данных определяются автоматически и это является замечательным достижением по сравнению с языком С.

```
# include <iostream>
using namespace std;
int main ( )
{
    char c='A';
    char string1[ ] = "проверка первой строки";
    char *string2 = "проверка второй строки";
    cout << "Значение символа равно " << c<<endl;
    cout << "Значение первой строки равно " << string1<<endl;
    cout << "Значение второй строки равно " << string2<<endl;
    return 0;
}
```

Ввод потоков

Операция ввода потока может быть выполнена с помощью операции взять из потока, т.е. перегруженной операции>>. Эта операция обычно игнорирует во входном потоке так называемые *символы разделители* или *пробельные символы* (пробелы, знаки табуляции, знак новой строки).

Операция взять из потока

Для того чтобы прочитать два целых числа, необходимо использовать объект cin и перегруженную операцию взять из потока >>. Операции взять из потока можно сцеплять.

Операция взять из потока также является достаточно «интеллектуальной», чтобы определить тип используемых данных. Если переменная была должным образом объявлена, то не требуется никакой дополнительной информации для использования операции взять из потока (в случае использования языка С такая информация требуется).

```
/* Вычисление суммы двух целых чисел, которые вводятся с клавиатуры с помощью cin и операции взять из потока */
# include <iostream>
using namespace std;
int main ( )
{
    int x,  y;
    cout << "Введите два целых числа:";
    cin >> x >>y;
    cout << "Сумма чисел " <<x<<" и "<<y<< " равна " <<(x+y)<<endl;
    return 0;
}
```

Манипуляторы потоков

В языке С++ имеется возможность использовать *манипуляторы потоков,* которые решают задачи форматирования. Манипуляторы потоков позволяют выполнять следующие операции:

- ✓ задание ширины полей;

- ✓ задание точности;
- ✓ установку и сброс флагов формата;
- ✓ задание заполняющего символа полей;
- ✓ сброс потоков;
- ✓ вставку в выходной поток символа новой строки и сброс потока;
- ✓ вставку нулевого символа в выходной поток и пропуск символов разделителей во входном потоке.

Манипуляторы потоков dec, oct, hex и setbase, задающие основание чисел

Целые числа обычно интерпретируются как десятичные (с основанием 10). Для изменения основания интерпретации целых чисел в потоке используются манипуляторы hex, чтобы установить шестнадцатеричный формат представления элементов данных (с основанием 16), oct, чтобы установить восьмеричный формат представления данных (с основанием 8). Манипулятор dec используется для возврата к основанию потока 10.

Основание потока может быть также изменено с помощью манипулятора потока setbase, который принимает один целый параметр со значениями 10, 8 или 16, задающими соответствующие основания системы счисления. Поскольку манипулятор setbase принимает параметр, он называется *параметризованным манипулятором потока.* Использование манипулятора setbase или любого другого параметризованного манипулятора требует включение заголовочного файла iomanip.h. Основание потока остается установленным до тех пор, пока оно не будет изменено явным образом.

```
// Использование манипуляторов потока hex, oct, dec и setbase.
#include<iomanip>
#include <iostream>
using namespace std;
int main (){
    int n;
    cout << " Введите десятичное число:" << endl;
    cin>>n;
    cout << n <<" в шестнадцатеричном формате равно " << hex << n <<endl;
    cout << n <<" в восьмеричном формате равно " << oct << n <<endl;
    cout << setbase(10) <<" в десятичном формате равно " << n <<endl;
    return 0;
}
```

Точность чисел с плавающей запятой (precision, setprecision)

Мы можем управлять точностью печатаемых чисел с плавающей запятой, т.е. числом разрядов справа от десятичной точки, используя манипулятор потока setprecision или функцию-элемент precision. Вызов любой из этих установок точности действует для всех последующих операций вывода до тех пор, пока не будет произведена следующая установка точности. Функция-элемент precision не имеет никаких аргументов и возвращает текущее значение точности.

```
// Управление точностью печати значений с плавающей запятой
#include<iomanip>
#include<iostream>
using namespace std;
int main ()
const float pi=3.14159265;
main ()
{
    cout << "PI with precision 1, 2, 3"<<endl;
    cout.precision(1);
    cout<<pi<<endl;
    cout.precision(2);
    cout<<pi<<endl;
    cout.precision(3);
```

```
    cout<<pi<<endl;
    cout << "PI with precision 3 = "<<setprecision(3)<< pi<<endl;
    cout << "PI with precision 4 = "<<setprecision(4)<< pi<<endl;
    cout << "PI with precision 5 = "<<setprecision(5)<< pi<<endl;
    return 0;
}
```

2. Типы данных и их объявление.

Объявления переменной в языке C/C++ имеют следующий формат:

[спецификатор_класса_памяти] спецификатор_типа описатель [=инициатор].

Спецификатор класса памяти – определяется одним из четырех ключевых слов языка C/C++: **auto, extern, register, static**, и указывает, каким образом будет распределяться память под объявляемую переменную, с одной стороны, а с другой, область видимости этой переменной, т.е., из каких частей программы можно к ней обратиться.

Спецификатор типа – одно или несколько ключевых слов, определяющие тип объявляемой переменной. В языке C/C++ имеется стандартный набор типов данных, используя который можно сконструировать новые (уникальные) типы данных.

Описатель – идентификатор простой переменной либо более сложная конструкция с квадратными скобками, круглыми скобками или звездочкой (набором звездочек).

Инициатор - задает начальное значение или список начальных значений, которые (которое) присваивается переменной при объявлении.

Переменная любого типа может быть объявлена как немодифицируемая. Это достигается добавлением ключевого слова **const** к спецификатору-типа. Объекты с типом **const** представляют собой данные используемые только для чтения, т.е. этой переменной не может быть присвоено новое значение. Отметим, что если после слова **const** отсутствует спецификатор-типа, то подразумевается спецификатор типа **int**. Если ключевое слово **const** стоит перед объявлением составных типов (массив, структура, смесь, перечисление), то это приводит к тому, что каждый элемент также должен являться немодифицируемым, т.е. значение ему может быть присвоено только один раз.

Примеры:

const double A=2.128E-2;

const B=286; (подразумевается const int B=286)

Целый тип данных

Для определения данных целого типа используются различные ключевые слова, которые определяют диапазон значений и размер области памяти, выделяемой под переменные.

Таблица 1 Целочисленные типы данных

Тип	Размер памяти в байтах	Диапазон значений
char	1	от -128 до 127
int	2	от -32768 до 32767
short	2	от -32768 до 32767
long	4	от -2 147 483 648 до 2 147 483 647
unsigned char	1	от 0 до 255
unsigned int	2	от 0 до 65535
unsigned short	2	от 0 до 65535
unsigned long	4	от 0 до 4 294 967 295

Ключевые слова **signed** и **unsigned** необязательны. Они указывают, как интерпретируется нулевой бит объявляемой переменной, т.е., если указано ключевое слово **unsigned**, то нулевой бит интерпретируется как часть числа, в противном случае нулевой бит интерпретируется как знаковый. В случае отсутствия ключевого слова **unsigned** целая переменная считается знаковой. В том случае, если спецификатор типа состоит из ключевого типа **signed** или **unsigned** и далее следует идентификатор переменной, то она будет рассматриваться как переменная типа **int**.

Например:

unsigned int n;

unsigned int b;

int c; (подразумевается signed int c);

unsigned d; (подразумевается unsigned int d);

signed f; (подразумевается signed int f).

Отметим, что модификатор-типа **char** используется для представления символа (из массива представление символов) или для объявления строковых литералов. Значением объекта типа **char** является код (размером 1 байт), соответствующий представляемому символу. Для представления символов русского алфавита, модификатор типа идентификатора данных имеет вид **unsigned char**, так как коды русских букв превышают величину 127.

Следует сделать следующее замечание: в языке C/C++ не определено представление в памяти и диапазон значений для идентификаторов с модификаторами-типа **int** и **unsigned int**. Размер памяти для переменной с модификатором типа **signed int** определяется длиной машинного слова, которое имеет различный размер на разных машинах.

Для определения длины памяти занимаемой переменной можно использовать операцию **sizeof** языка C/C++, возвращающую значение длины указанного модификатора-типа.

Например:

```
a = sizeof(int);
b = sizeof(long int);
c = sizeof(unsigned long);
d = sizeof(short);
```

3. Составление алгоритма

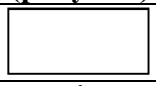
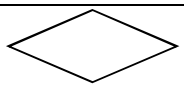
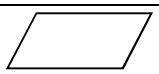
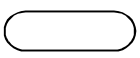
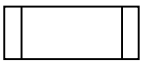
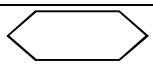
Перед написанием программы на алгоритмическом языке следует составить алгоритм решения задачи. Под алгоритмом понимают детально описанную последовательность действий (операций), приводящую к решению поставленной задачи. Основными формами представления алгоритма являются: словесное описание, блок- схемы, структурограммы.

Составление алгоритмов графическим способом подчиняется двум ГОСТам:

ГОСТ 19.002-80, соответствует международному стандарту ИСО 2636-73. Регламентирует правила составления блок-схем.

ГОСТ 19.003-80, соответствует международному стандарту ИСО 1028-73. Регламентирует использование графических примитивов.

Таблица 2 Графические примитивы

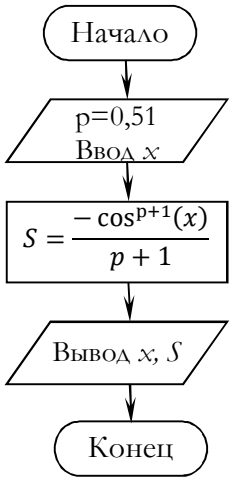
Название	Символ (рисунок)	Выполняемая функция (пояснение)
1. Блок вычислений		Выполняет вычислительное действие или группу действий
2. Логический блок		Выбор направления выполнения алгоритма в зависимости от условия
3. Блоки ввода/вывода		Ввод или вывод данных вне зависимости от физического носителя
		Вывод данных на печатающее устройство
4. Начало/конец (вход/выход)		Начало или конец программы, вход или выход в подпрограмму
5. Предопределенный процесс		Вычисления по стандартной или пользовательской подпрограмме
6. Блок модификации		Выполнение действий, изменяющих пункты алгоритма
7. Соединитель		Указание связи между прерванными линиями в пределах одной страницы
8. Межстраничный соединитель		Указание связи между частями схемы, расположенной на разных страницах

Пример вычислительной программы

Составить алгоритм и программу вычисления значения функции

$$S = \frac{-\cos(x)^{p+1}}{p+1}$$, где p=0,51=const

Алгоритм решения этой задачи можно представить в виде следующей последовательности действий:



Программа на языке С++, соответствующая этому алгоритму, может иметь следующий вид:

```
//Составление простейшей линейной программы
#include <iostream>
#include <Cmath>
using namespace std;
float const p=0.51;
int main( )
{
    setlocale(LC_ALL, "Russian");
    int x; float S;
    cout << "Ввести x" << endl;
    cin >> x;
    S=(-pow((cos(x)),(p+1)))/(p+1);
    cout << "x=" << x << "  S=" << S << endl;
    system("pause");
    return 0;
}
```

Математические библиотечные функции

Математические библиотечные функции позволяют программисту выполнять определенные типовые математические вычисления.

Обычно функция вызывается путем записи имени функции, после которого записывается левая круглая скобка, затем *аргумент* функции (или список аргументов, разделенных запятыми), а завершает запись правая круглая скобка.

При использовании функций математической библиотеки необходимо подключать соответствующий заголовочный файл с помощью директивы препроцессора #include <Cmath>.

Таблица 3 Базовые математические функции стандарта С++

Имя	**Описание**
abs(x)	$\|x\|$–возвращает абсолютную величину целого числа *x*.
acos(x)	$\arccos(x)$ – вычисляет главное значение арккосинуса аргумента x. Если заданный аргумент не попадает в диапазон значений [-1,+1], происходит ошибка выхода за допустимые пределы области определения. Функция **acos** возвращает значение арккосинуса в диапазоне $[0, \pi]$ в радианах.

asin(x)	$\arcsin(x)$ – вычисляет главное значение арксинуса аргумента x. Если заданный аргумент не попадает в диапазон значений [-1, + 1], происходит ошибка выхода за пределы области определения. Функция **asin** возвращает значение арксинуса в диапазоне $\left[-\frac{\pi}{2},\frac{\pi}{2}\right]$ в радианах.
atan(x)	$\operatorname{arctg}(x)$ – вычисляет главное значение арктангенса аргумента x. Функция **atan** возвращает значение арктангенса в диапазоне $\left[-\frac{\pi}{2},\frac{\pi}{2}\right]$ в радианах.
atan2(y,x)	$\operatorname{arctg}\left(\frac{x}{y}\right)$ вычисляет главное значение арктангенса выражения $\frac{x}{y}$, используя знаки обоих аргументов для определения квадранта, в котором выбирается возвращаемое значение. Функция **atan2** возвращает значение арктангенса $\frac{x}{y}$ в диапазоне $\left[-\frac{\pi}{2},\frac{\pi}{2}\right]$ в радианах.
ceil(x)	округление до ближайшего большего целого числа
cos(x)	$\cos(x)$ – вычисляет косинус аргумента *x*, значение которого задается в радианах.
cosh(x)	$\operatorname{ch}(x)$ – вычисляет гиперболический косинус аргумента x
cbrt(x)	$\sqrt[3]{x}$ – вычисляет значение кубического корня из аргумента *x*.
exp(x)	e^x – возведение экспоненты в степень ***x***.
exp2(x)	2^x – возведение числа 2 в степень ***x***.
fabs(x)	$\lvert x\rvert$ – абсолютная величина числа с плавающей точкой
floor(x)	округление до ближайшего меньшего целого числа
fmod(x)	вычисление остатка от деления нацело для чисел с плавающей точкой
frexp(x, *p)	разбивает число с плавающей точкой на мантиссу и показатель степени.
ldexp(x, p)	умножает число с плавающей точкой на число 2, возведенное в целую степень. Может возникнуть ошибка выхода за диапазон допустимых значений. Функция возвращает значение, равное произведению $x \cdot 2^p$.
log(x)	$\ln(x)$ – вычисляет натуральный логарифм аргумента ***x***.
log2(x)	$\log_2(x)$ – вычисляет логарифм по основанию 2 аргумента ***x***.
log10(x)	$\lg(x)$ – вычисляет десятичный логарифм аргумента x.
modf(x, *p)	извлекает целую и дробную части (с учетом знака) из числа с плавающей точкой
pow(x,y)	x^y – вычисляет результат возведения *x* в степень *y*
sin(x)	$\sin(x)$ – вычисляет синус аргумента *x*, значение которого задается в радианах.
sinh(x)	$\operatorname{sh}(x)$ – вычисляет гиперболический синус аргумента x.
sqrt(x)	$\sqrt{x}$ – вычисляет положительное значение квадратного корня из аргумента x . Если аргумент имеет отрицательное значение, происходит ошибка выхода за пределы области определения
tan(x)	$\operatorname{tg}(x)$ – вычисляет тангенс аргумента x, значение которого задается в радианах
tanh(x)	$\operatorname{th}(x)$ – вычисляет гиперболический тангенс аргумента *x*

4. Структура выбора if /else (ЕСЛИ-ИНАЧЕ)

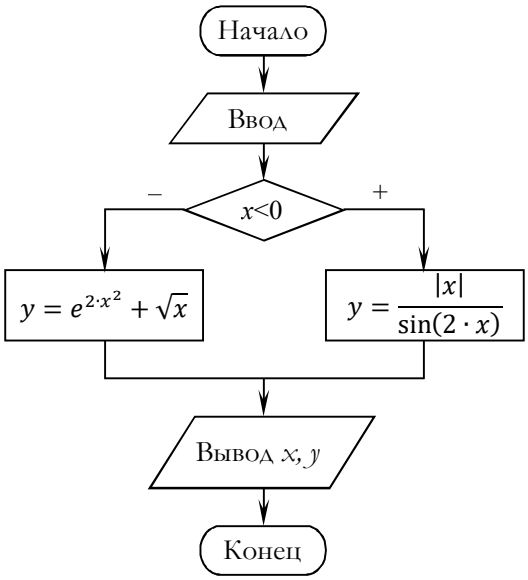
Структура выбора используется для выбора среди альтернативных путей обработки информации.

Структура выбора if/else позволяет программисту определить различные действия, которые должны выполняться в случаях, если условие истинно или ложно.

Пример. Составить алгоритм и программу вычислений значения функции.

$$y = \begin{cases} \dfrac{|x|}{\sin(2x)}, & \text{если } x < 0 \\ e^{2x^2} + \sqrt{x}, & \text{если } x \geq 0 \end{cases}$$

Алгоритм решения поставленной задачи можно составить следующим образом:



1. Задается значение аргумента *x*.

2. Проверяется условие $x<0$. Если это условие выполняется, то вычисляется $y = \frac{|x|}{\sin(2 \cdot x)}$, в противном случае значение функции вычисляется по формуле $y = e^{2 \cdot x^2} + \sqrt{x}$.

3. Выводится на экран дисплея значение *x* и соответствующее ему значение функции.

Текст программы на языке C++, реализующей приведенный алгоритм, может иметь следующий вид:

```
#include <iostream>
#include <Cmath>

using namespace std;

int main( )
{
        setlocale(LC_ALL, "Russian");
        int x, float y;
        cout << “Ввести x” << endl;
        cin >> x;
        if (x<0)
                y=abs(x)/sin(2*x);
        else y=exp(2*x*x)+sqrt(x);
        cout << “y=” << y<<endl;
        system(“Pause”);
        return 0;
}
```

C++ имеет еще *условную операцию* (?:), которая близка к структуре if/else. Условная операция – единственная *трехчленная (тернарная) операция* в C++, имеющая три операнда. Эти операнды вместе с условной операцией формируют *условное выражение.* Первый операнд является условием, второй операнд содержит значение условного выражения в случае, если условие истинно, а третий операнд равен значению условного выражения, если условие ложно.

```
x<0 ? y=abs(x)/sin(2*x) : y=exp(2*x*x)+sqrt(x);
```

5. Циклические конструкции

Цикл с предусловием – while

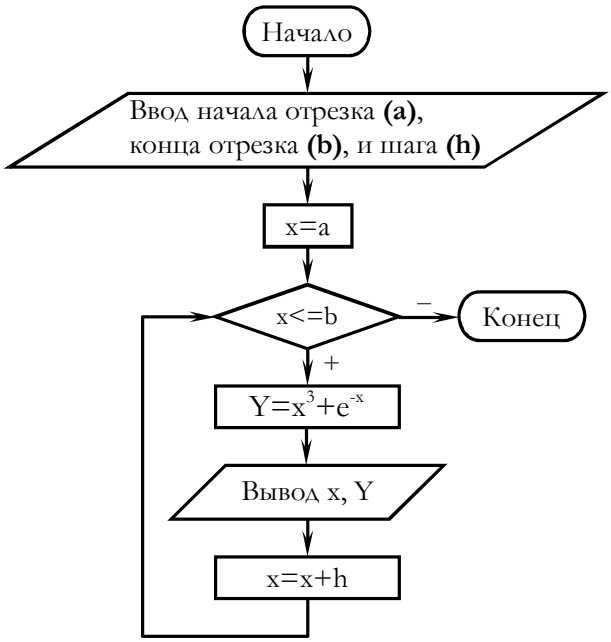
Структура повторения позволяет программисту определить действие, которое должно повторяться, пока некоторое условие остается истинным.

Оператор (или операторы), записанные в теле структуры повторения **while** составляют тело **while**, которое может быть отдельным или составным оператором. Очевидно, что когда-нибудь условие станет ложным. Начиная с этого момента повторение прерывается и выполняется первое предложение

псевдокода, следующее за структурой повторения.

Пример. Составить программу вычисления значения функции $Y=x^3+e^{-x}$ при изменении x на отрезке [a;b] с шагом h.

Алгоритм решения поставленной задачи можно составить следующим образом:



Текст программы на языке С++, реализующей приведенный алгоритм, может иметь следующий вид:

```
#include <iostream> // Подключение модуля для работы с клавиатурой и монитором
#include <Cmath> // Подключение модуля для использования математических функций

using namespace std;

int main( ) {
                setlocale(LC_ALL, "Russian");
                float a,b,h,Y,x;
                cout << "Ввести начало, конец и шаг отрезка"<< endl;
                cin>>a>>b>>h;
                x=a;  // присвоение переменной x значения соответствующего началу отрезка
                while (x<=b) {  /* описание цикла с предусловием. Цикл продолжается до тех пор,
                                   пока x≤b */
                    Y=pow(x,3)+exp(-x); // вычисление выражения на очередной итерации цикла.
                    cout << "x="<< x <<"  Y="<<Y<< endl; // вывод значения x и Y.
                    x=x+h; // Увеличение переменной на шаг цикла.
                }
                System("Pause");
                return 0;
}
```

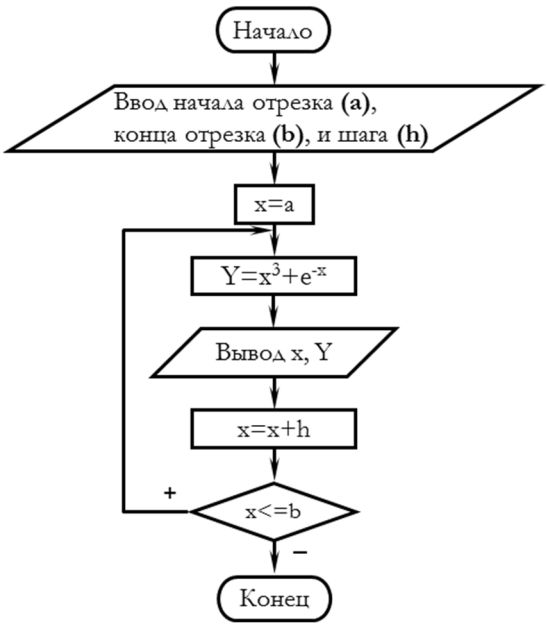
Структура цикла с постусловием do/while

Структура повторения **do/while** похожа на структуру **while**. В структуре **while** условие продолжения циклов проверяется в начале цикла, до того, как выполняется тело цикла. В структуре **do/while** проверка условия продолжения циклов производится *после* того, как тело цикла выполнено, следовательно, тело цикла будет выполнено по крайней мере один раз. Когда **do/while** завершается, выполнение программы продолжается с оператора, следующего за предложением **while**. Отметим, что в структуре **do/while** нет необходимости использовать фигурные скобки, если тело состоит только из одного оператора. Но фигурные скобки обычно все же ставят, чтобы избежать путаницы между структурами while и do/while.

Пример. Составить программу вычисления значения функции $Y=x^3+e^{-x}$ при изменении x на

отрезке [a;b] с шагом h.

Алгоритм решения поставленной задачи можно составить следующим образом:



Текст программы на языке С++, реализующей приведенный алгоритм, может иметь следующий вид:

```
#include <iostream> // Подключение модуля для работы с клавиатурой и монитором
#include <Cmath> // Подключение модуля для использования математических функций

using namespace std;

int main( ) {
                setlocale(LC_ALL, "Russian");
                float a,b,h,Y,x;
                cout << "Ввести начало, конец и шаг отрезка"<< endl;
                cin>>a>>b>>h;
                x=a;  // присвоение переменной x значения соответствующего началу отрезка
                do {  /* описание цикла с предусловием. Цикл продолжается до тех пор, пока x≤b */
                    Y=pow(x,3)+exp(-x); // вычисление выражения на очередной итерации цикла.
                    cout << "x="<< x <<"  Y="<<Y<< endl; // вывод значения x и Y.
                    x=x+h; // Увеличение переменной на шаг цикла.
                }
                while (x<=b)
                System("Pause");
                return 0;
}
```

Структура цикла с параметром for.

Структура повторения **for** содержит все элементы, необходимые для повторения, управляемого счетчиком. В общем виде структура параметрического цикла выглядит следующим образом:

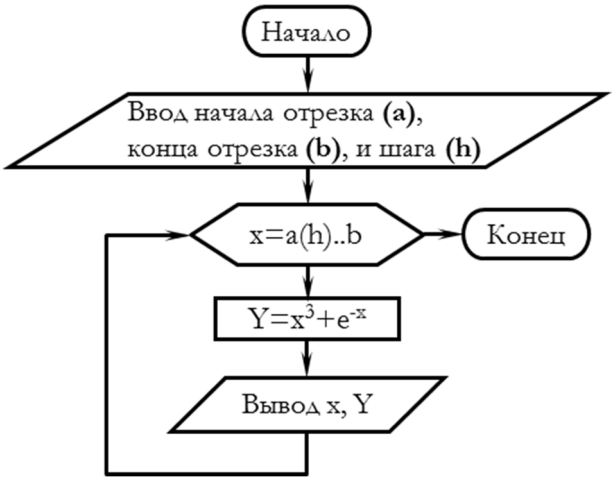
for (имя управляющей переменной**=**начальное значение управляющей переменной**;** конечное значение управляющей переменной; приращение управляющей переменной)

Когда структура **for** начинает выполняться, управляющей переменной задается начальное значение. Затем проверяется условие продолжения цикла. Если условие удовлетворяется, выполняется тело цикла. Затем управляющая переменная увеличивается на величину приращения и цикл опять начинается с проверки условия его продолжения. Этот процесс продолжается, пока управляющая

переменная не увеличится до конечного значения – это приведет к тому, что условие продолжения цикла нарушится и повторение прекратится. Выполнение программы продолжится с первого оператора, расположенного после структуры **for**.

Пример. Составить программу вычисления значения функции $Y=x^3+e^{-x}$ при изменении x на отрезке [a;b] с шагом h.

Алгоритм решения поставленной задачи можно составить следующим образом:



Текст программы на языке С++, реализующей приведенный алгоритм, может иметь следующий вид:

```
#include <iostream> // Подключение модуля для работы с клавиатурой и монитором
#include <Cmath> // Подключение модуля для использования математических функций

using namespace std;

int main( ) {
                setlocale(LC_ALL, "Russian");
                float a,b,h,Y,x;
                cout << "Ввести начало, конец и шаг отрезка"<< endl;
                cin>>a>>b>>h;
                x=a;  // присвоение переменной x значения соответствующего началу отрезка
                for (x=a; x<=b; x=x+h)
                    Y=pow(x,3)+exp(-x); // вычисление выражения на очередной итерации цикла.
                    cout << "x="<< x <<"  Y="<<Y<< endl; // вывод значения x и Y.
                }
                System("Pause");
                return 0;
}
```

Выражения в структуре **for** являются необязательными. Если конечное значение управляющей переменной отсутствует, С++ предполагает, что условие продолжения цикла всегда истинно и таким образом создается бесконечно повторяющийся цикл. Иногда может отсутствовать начальное значение управляющей, если начальное значение управляющей переменной задано где-то в другом месте программы. Может отсутствовать и приращение управляющей переменной, если приращение переменной осуществляется операторами в теле структуры **for** или если приращение не требуется. Выражение для приращения переменной в структуре **for** действует так же, как автономный оператор в конце тела **for**.

«Приращение» структуры **for** может быть отрицательным (в этом случае в действительности происходит не приращение, а уменьшение переменной, управляющей циклом). Приращение цикла может и дробным. Для этого управляющая переменная должна иметь соответствующий тип

Если условие продолжения цикла с самого начала не удовлетворяется, то операторы тела структуры **for** не выполняются и управление передается оператору, следующему за **for**.

Варианты заданий

Таблица 1

Номер	Функция	Отрезок	Шаг	Тип цикла
1	$y=\begin{cases}\dfrac{\sqrt{\lvert x\rvert}}{\sin(2-x)}, & \text{если } x<0\\ e^{\sqrt{x}}+x^2, & \text{если } x\geq 0\end{cases}$;	[-2;1]	0,5	С предусловием
2	$y=\begin{cases}x^2+1, & \text{если } x\leq\dfrac{\pi}{2}\\ x-2, & \text{если } x>\dfrac{\pi}{2}\end{cases}$	[0;π]	0,2	С постусловием
3	$y=\begin{cases}0.25\cdot x^3, & \text{если } x\leq 0\\ 3\cdot\sin(\sqrt{x}), & \text{если } x>0\end{cases}$	[-1;1]	0,4	С параметром
4	$y=\begin{cases}\sin(x+y), & \text{если } \lvert x\rvert\leq 1\\ e^{x^2}, & \text{если } \lvert x\rvert>1\end{cases}$	[-5;5]	1	С предусловием
5	$A=\begin{cases}e^{x+1}, & \text{если } \sin(x)>0.4\\ \operatorname{ctg}(x), & \text{если } \sin(x)\leq 0.4\end{cases}$;	[1;7]	1	С постусловием
6	$y=\begin{cases}\sqrt{k}, & \text{если } \lvert\sin(k)\rvert\leq 0.5\\ \sqrt[3]{\sqrt{k}} & \text{если } \lvert\sin(k)\rvert>0.5\end{cases}$	[0;3]	0,5	С параметром
7	$x=\begin{cases}\dfrac{z}{2}, & \text{если } \sin(z)<0.68\\ z-1, & \text{если } \sin(z)\geq 0.68\end{cases}$;	[0;1]	0,1	С предусловием
8	$V=\begin{cases}\operatorname{arctg}\left(\dfrac{\sqrt{i}+2}{n+3}\right), & \text{если } \operatorname{tg} i<3\\ e^{i+\cos(n)}, & \text{если } \operatorname{tg} i\geq 3\end{cases}$ при $n=2$;	[1;10]	2	С постусловием
9	$W=\begin{cases}\sin(\sqrt{x}), & \text{если } x<-\dfrac{\pi}{2}\\ \cos(x)-\sin(x), & \text{если } -\dfrac{\pi}{2}\leq x\leq\dfrac{\pi}{2}\\ \cos(\sqrt{x}), & \text{если } x>\dfrac{\pi}{2}\end{cases}$;	[-1.5;1.5]	0,2	С параметром
10	$A=\begin{cases}\sin\left(\dfrac{y^2+1}{n}\right), & \text{если } \sin\left(\dfrac{y^2+1}{n}\right)>0\\ \cos\left(y+\dfrac{1}{n}\right), & \text{если } \sin\left(\dfrac{y^2+1}{n}\right)\leq 0\end{cases}$;	[1;10]	2	С предусловием
11	$y=\begin{cases}x^2+b, & \text{если } x>5\\ x^3-4b, & \text{если } x\leq 5\end{cases}$ при $b=0.5$;	[4;6]	0,5	С постусловием
12	$y=\begin{cases}x^z, & \text{если } z\leq 5x\\ \dfrac{1}{\cos(x\cdot z)}, & \text{если } z>5x\end{cases}$ при $x=0.5$;	[1;2]	0,1	С параметром
13	$Z=\begin{cases}\sin(y), & \text{если } \cos(y)<0.5\\ \operatorname{ctg}(y), & \text{если } \cos(y)\geq 0.5\end{cases}$;	[1;9]	1	С предусловием
14	$M=\begin{cases}\sqrt{x^2+y^2}, & \text{если } x+y<0\\ x^3, & \text{если } x+y\geq 0\end{cases}$	[-1,5;1,5]	0,5	С постусловием
15	$z=\begin{cases}e^{x^2}+\lvert y\rvert, & \text{если } \lvert x\rvert+\lvert y\rvert>1\\ \sin(x+y), & \text{если } \lvert x\rvert+\lvert y\rvert\leq 1\end{cases}$	[-2;2]	0,5	С параметром
16	$y=\begin{cases}\dfrac{a\cdot x^2-b\cdot x-c}{10^{-6}}, & \text{если } x=10\\ \dfrac{a\cdot x^2+b\cdot x+c}{10^6}, & \text{если } x\neq 10\end{cases}$	[1;2]	0,1	С предусловием

17	$y=\begin{cases}0.1\cdot x^2-x\cdot\ln(x), & \text{если } \sqrt{x}<1.2\\ \cos(x)+e^{-\frac{x^2}{2}}, & \text{если } \sqrt{x}\geq 1.2\end{cases}$;	[1;2]	0,2	С постусловием
18	$T=\begin{cases}a\cdot\sin(x), & \text{если } \cos(x)<0.3\\ a^2\cdot\operatorname{tg}(\sqrt{x}), & \text{если } \cos(x)\geq 0.3\end{cases}$;	[0,5;1]	0,1	С параметром
19	$y=\begin{cases}\frac{x}{2}\cdot\sin(x), & \text{если } x<0\\ \ln\left(\frac{1}{2\cdot x}\right), & \text{если } x\geq 0\end{cases}$	[-1;1]	0,2	С предусловием
20	$Y=\begin{cases}e^{x+2}, & \text{если } \sin(x)>0.4\\ 2\cdot\operatorname{tg}(x), & \text{если } \sin(x)\leq 0.4\end{cases}$;	[1;5]	1	С постусловием
21	$y=\begin{cases}\ln\left\lvert 2\cdot\sin\left(\frac{x}{2}\right)\right\rvert, & \text{если } x<0\\ e^{\cos(x)}, & \text{если } x\geq 0\end{cases}$;	[-10;10]	2	С параметром
22	$F=\begin{cases}x+\sqrt{x}+\sqrt[3]{x}, & \text{если } x>0\\ \sqrt{1+e^{2\cdot\pi}}, & \text{если } x\leq 0\end{cases}$;	[-0.5;1]	0,1	С предусловием
23	$k=\begin{cases}1-\sin(x), & \text{если } 5\leq x<0\\ \frac{1}{2}\cdot(1+\cos(x)), & \text{если } 10\leq x\leq 15\end{cases}$;	[0;50]	5	С постусловием
24	$F=\begin{cases}-\ln\lvert\sin(x)\rvert, & \text{если } x<0\\ \operatorname{sh}(x), & \text{если } x\geq 0\end{cases}$;	[0,1;0,2]	0,01	С параметром
25	$S=\begin{cases}\operatorname{tg}^3(x), & \text{если } x\geq 0\\ \sqrt{1+e^x}, & \text{если } x<0\end{cases}$;	[-1;1]	0,2	С предусловием
26	$y=\begin{cases}\operatorname{arctg}(\sqrt{x}), & \text{если } \operatorname{tg}(x)\leq 2\\ e^{x+\cos(x)}, & \text{если } \operatorname{tg}(x)\geq 2\end{cases}$;	[0;1]	0,1	С постусловием
27	$Y=\begin{cases}e^{\frac{x}{2}}, & \text{если } \sin(x)>0.5\\ \operatorname{tg}(x)+\cos(x), & \text{если } \sin(x)\leq 0.5\end{cases}$;	[0;10]	1	С параметром
28	$y=\begin{cases}\sqrt[3]{x}, & \text{если } x-\text{четное}\\ x^3, & \text{если } x-\text{нечетное}\end{cases}$;	[0;10]	1	С предусловием
29	$z=\begin{cases}e^{x^2}+\lvert y\rvert, & \text{если } \lvert x\rvert+\lvert y\rvert>1\\ \sin(x+y), & \text{если } \lvert x\rvert+\lvert y\rvert\leq 1\end{cases}$	[0;1]	0,1	С постусловием
30	$W=\begin{cases}(x^2+3)^2-\sqrt{0.5\cdot\pi+y}, & \text{если } x\cdot y<0\\ x^2+3\cdot\sqrt[3]{(\pi-y)}, & \text{если } x\cdot y\geq 0\end{cases}$ при $y=0.5$	[-10;0]	0,5	С параметром
31	$S=\begin{cases}\operatorname{ch}^3(x), & \text{если } x\geq 0\\ \sqrt{\frac{1}{e^x}}, & \text{если } x<0\end{cases}$;	[-1;1]	0,2	С предусловием
32	$z=\begin{cases}e^{i+1}, & \text{если } \sin(i)>0.3\\ \arcsin(i), & \text{если } \sin(i)\leq 0.3\end{cases}$;	[-5;4]	0,4	С постусловием
33	$F=\begin{cases}\frac{\sqrt{x}+x^2}{\sqrt[4]{x}+x^4}, & \text{если } x>0\\ \frac{\sqrt[3]{x}+x^3}{\sqrt[3]{x}+x^2}, & \text{если } x\leq 0\end{cases}$;	[-20;30]	7	С параметром
34	$y=\begin{cases}\cos(x)+\tan(x), & \text{если } x-\text{кратно 3}\\ \sin(x)+\operatorname{ctg}(x), & \text{если } x-\text{не кратно 3}\end{cases}$;	[0;30]	2	С предусловием
35	$W=\begin{cases}\sqrt{x^3}, & \text{если } -\frac{\pi}{2}\leq\lvert\arcsin(k)\rvert\\ \sqrt[3]{x}, & \text{если } \lvert\arcsin(k)\rvert>\frac{\pi}{2}\end{cases}$	[-3;3]	0.2	С постусловием

36	$F=\begin{cases}\ln(x), & \text{если } x>3\\ \lg(x), & \text{если } x<3\\ \log_2(x), & \text{в остальных случаях}\end{cases}$;	[-6;6]	0,5	С параметром
37	$W=\begin{cases}\sin(2x)-\cos(2x), & \text{если } x<0\\ tg(x)\cdot ctg(x), & \text{если } x>0\\ \dfrac{1}{2\cdot\sin^2(x)}, & \text{если } x=0\end{cases}$;	[-3;3]	0.2	С предусловием
38	$W=\begin{cases}x^3-2\cdot\dfrac{y}{3}, & \text{если } (x+y)<0\\ \dfrac{x-y}{\sqrt{x+y}}, & \text{если } (x+y)\geq 0\end{cases}$ при $x=2$	[-1;1]	0,2	С постусловием
39	$y=\begin{cases}2\cdot\sin(x), & \text{если } x\leq -\dfrac{\pi}{2}\\ a\cdot\sin(x)+b, & \text{если } -\dfrac{\pi}{2}<x<\dfrac{\pi}{2}\\ \cos(x), & \text{если } x\geq\dfrac{\pi}{2}\end{cases}$	[-3;3]	0.2	С параметром
40	$W=\begin{cases}\sin(\sqrt{x}), & \text{если } x<-\dfrac{\pi}{2}\\ \cos(x)-\sin(x), & \text{если } -\dfrac{\pi}{2}\leq x\leq\dfrac{\pi}{2}\\ \cos(\sqrt{x}), & \text{если } x>\dfrac{\pi}{2}\end{cases}$;	[-3;3]	0.2	С предусловием