## Алгоритмизация

### Кувшины

Имеются два кувшина емкостью 3 и 8 л. Необходимо составить алгоритм, с помощью которого, пользуясь только этими двумя кувшинами, можно набрать 7 л воды

[**R0001**](#R0001)

### Вычисление наибольшего общего делителя (НОД). Алгоритм Евклида.

Вычислить наибольший общий делитель (НОД) двух натуральных чисел А и В с применением алгоритма Евклида. Основная идея алгоритма в том, что НОД А и В есть также и НОД (А - В), т.е. последовательное вычитание из большего числа меньшего до тех пор, пока числа не сравняются, должно привести к искомому значению НОД

[**R0002**](#R0002)

### Числа Фибоначчи

Вывести все числа ряда Фибоначчи (1, 1, 2, 3, 5, 8 ...) до заданного натурального N. Очередной член ряда F, определяется как сумма двух предыдущих (Fj = Fj-1 + Fj-2). Первые два члена ряда равны 1.

[**R0003**](#R0003)

## Данные

### #1

Что выведет программа?

#include "iostream"

using namespace std;

int digit(char ch)

{

return(!(ch >= '0' && ch <= '9'));

}

int main()

{

cout << "digit('a')=" << digit('a') << endl;

cout << "digit('1')=" << digit('1') << endl;

return 0;

}

[**R0004**](#R0004)

### #2

Что в результате появится на экране?

#include <stdio.h>

void main()

{

printf("(%-\*.\*f)", 10, 2, 13.567);

printf("\n");

}

[**R0046**](#R0046)

### #3

Каковы в результате будут значения i и str, при входном потоке «15 Belarus»?

#include <stdio.h>

void main()

{

char str[50];

int i;

scanf("%d%5s", &i, str);

printf("i = %d\nstr = %s\n", i, str);

}

[**R0047**](#R0047)

### #4

Чему будет равна строка str при вводе «Belarus»?

#include <stdio.h>

void main()

{

char str[50];

scanf("%[a-z]", str);

printf("str = %s\n", str);

}

[**R0048**](#R0048)

### #5

Каков будет результат при использовании функций sscanf() и sprintf()?

#include <stdio.h>

int main()

{

char str1[10]="1.14", str2[10]="2.14";

float fvalue1=3.14, fvalue2=4.14;

sprintf(str1, "%f", fvalue1);

sscanf(str2, "%f", &fvalue2);

printf("str1 = %s\nstr2 = %s\nfvalue1 = %f\nfvalue2 = %f\n", str1, str2, fvalue1, fvalue2);

return 0;

}

[**R0049**](#R0049)

## Операции

### #1

Чему равно x и y?

#include "iostream"

using namespace std;

int main()

{

int x=2, y=2;

if((x==2) && (y+=++x))

--x,y++;

cout << "x=" << x << endl << "y=" << y << endl;

return 0;

}

[**R0005**](#R0005)

### #2

Выполнится ли данная программа без ошибок? Если да, то какой результат? Если нет, то почему?

int main()

{

http://www.google.ru/

return 0;

}

[**R0006**](#R0006)

### #3

Чему будет равен Х в результате выполнения данной программы?

#include <iostream>

using namespace std;

int main()

{

int x=2;

x=x\*x++;

cout << x << endl;

return 0;

}

[**R0029**](#R0029)

### #4

Чему будет равен Н в результате выполнения данной программы?

#include <iostream>

using namespace std;

int main()

{

int y=2;

y=y\*++y;

cout << y << endl;

return 0;

}

[**R0030**](#R0030)

### #5

Что будет выведено на экран в результате выполнения операции cout?

#include <iostream>

using namespace std;

int main()

{

cout << 8/3 << endl;

return 0;

}

[**R0032**](#R0032)

### #6

Чему в результате окажется равна переменная X?

#include <iostream>

using namespace std;

int main()

{

int x=2;

while(!x)

x--;

cout << "x = " << x << endl;

return 0;

}

[**R0035**](#R0035)

### #7

Допустимо ли такое использование макроопределения? И если да то, чему будет равен Х?

#include <iostream>

using namespace std;

#define Y 5

#define X !Y

int main()

{

cout << "X = " << X << endl;

return 0;

}

[**R0037**](#R0037)

### #8

Что будет выведено на экране в результате выполнения данной программы?

#include <iostream>

using namespace std;

int main()

{

char ch='5';

int x=5;

while(ch<8)

ch++;

cout << "CH = " << ch << endl;

return 0;

}

[**R0039**](#R0039)

### #9

Чему будет равен Х и Y в результате выполнения данной программы?

#include <iostream>

using namespace std;

int main()

{

int x=10, y=8;

y/=2;

if(x=5)

y\*=x;

cout << "X = " << x << ", Y = " << y << endl;

return 0;

}

[**R0040**](#R0040)

### #10

Какой будет результат выполнения программы и есть ли разница?

#include <iostream>

using namespace std;

int main()

{

int x=4, y=3;

cout << (x | y) << endl;

cout << (x || y) << endl;

return 0;

}

[**R0044**](#R0044)

### #11

Чему равна переменная С?

#include <iostream>

using namespace std;

int main()

{

int a=0;

int b=1;

int c(a + b);

cout << c << endl;

return 0;

}

[**R0050**](#R0050)

## Переключение ввода-вывода

### #1

Что будет выведено на экране в результате выполнения данной программы?

#include <iostream>

#include <stdio.h>

using namespace std;

int main()

{

int x=1;

char buf[]=" ";

cout << x << buf << x << 1 << buf << x << 2 << endl;

printf("%d %d %d\n", x, x << 1, x << 2);

return 0;

}

[**R0043**](#R0043)

## Выбор вариантов

### #1

Что будет выведено на экран в результате выполнения данной программы?

#include <iostream>

using namespace std;

int ischar1\_3(char ch)

{

switch(ch)

{

case '1' : return(1);

case '2' : return(1);

case '3' : return(1);

default : return(0);

}

}

int main()

{

char ch3='3', ch8='8';

cout << "ch3 = " << ischar1\_3(ch3) << ", ch5 = " << ischar1\_3(ch8) << endl;

return 0;

}

[**R0041**](#R0041)

## Цикл

### Угадай число

Угадай задуманное мной число в диапазоне 1-100

### Награда изобретателю шахмат

На какой клетке шахматной доски не хватит урожая США для того, чтобы рассчитаться с изобретателем шахмат (1 клетка 1 зернышко пшеницы, 2 клетка 2 зернышка, 3 клетка 4 зернышка, …)

### #1

Дан код вывода 100 чисел. Есть ли ошибки?

unsigned int i;

for (i = 100; i >= 0; --i)

printf("%d\n", i);

[**R0007**](#R0007)

### #2

Чему будет равно x и n?

#include "iostream"

using namespace std;

int main()

{

int x = 1,n=0;

while(++x < 10)

{

n++;

if(!(x % 2)) continue;

if(!(x % 2)) break;

}

cout << "x=" << x << "; n=" << n << endl;

return 0;

}

[**R0008**](#R0008)

### #3

Сколько будет выведено звездочек на экране?

#include <iostream>

using namespace std;

int main()

{

int i, N=20;

for(i = 0 ; i < N ; i--)

printf("\*");

return 0;

}

[**R0051**](#R0051)

## Массив

### Среднее

Вычислить среднее арифметическое элементов массива без учета минимального и максимального элементов массива.

### Число в массиве

Проверить, находится ли введенное с клавиатуры число в массиве.

### Возрастающая последовательность

Проверить является ли введенная последовательность (массив) возрастающей последовательностью

### Одинаковые значения

Проверить, есть ли во введенном массиве элементы с одинаковым значением

### Число в массиве

Вычислить, сколько раз введенное с клавиатуры число встречается в массиве

### Частота цифр

Вычислить, частоту появления цифр в массиве

### Сортировка

Выполнить сортировку массива в прямом и обратном направлении

### Строки и массив

Ввести строки из цифр и вычислить сумму его элементов по столбцам

### #1

Что получится в результате выполнения программы? Как она работает?

#include <iostream>

using namespace std;

int i;

int main()

{

int a[] = {2, 3, 4};

for(;i!=sizeof(a)/sizeof(int);)

cout << i++[a];

cout << endl;

return 0;

}

[**R0009**](#R0009)

### #2

Чему будут равны значения x и y?

#include "iostream"

using namespace std;

int main()

{

int\* x = new int ();

int\* y = new int (rand() % 3);

cout << "x=" << \*x << endl;

cout << "y=" << \*y << endl;

return 0;

}

[**R0010**](#R0010)

### #3

Какой будет результат?

#include "iostream"

using namespace std;

int main ()

{

int \*x = (int\*)calloc(1,sizeof(int));

int \*y = (int\*)malloc(sizeof(int));

cout << "\*x=" << \*x << endl;

cout << "\*y=" << \*y << endl;

}

[**R0028**](#R0028)

### #4

Правильно ли выделена память для двумерного массива типа integer?

#include <iostream>

using namespace std;

int main()

{

int \*\*X;

X=new int [5][5];

return 0;

}

[**R0033**](#R0033)

### #5

Что будет выведено на экран в результате выполнения программы?(

#include <iostream>

using namespace std;

int main()

{

int a[5];

cout << a << endl;

cout << &a << endl;

return 0;

}

[**R0034**](#R0034)

### #6

Дана программа:

#include <iostream>

#include <ctime>

#include <cstdlib>

using namespace std;

int main()

{

int mass[3];

for(int i=0;i<3;i++)

mass[i]=rand() %5;

for(int i=0;i<3;i++)

cout << mass[i] << " ";

cout << endl;

return 0;

}

Программа выводит

5 rand

Какие числа будут сгенерированы при повторном запуске программы?

[**R0036**](#R0036)

## Указатель

### Сортировка строк

Создать программу сортировки массива строк, представленных в динамической памяти и адресуемых указателями. Признаком конца ввода - ввод пустой строки, количество строк не более k.

[R0011](#R0011)

### Матрицы и сортировка

Дано n матриц, размеры матриц не более 5 х 10. Используя указатели, рассортировать столбцы каждой матрицы следующим образом: первый столбец - по возрастанию, второй - по убыванию и т. д.

### Строки

Ввести массив строк. Удалить из каждой строки все слова (числа), состоящие только из цифр, а сумму этих чисел записать последним словом строки. Дополнительных строк или массивов создавать нельзя, стандартные функции работы со строками не использовать.

### Матрица указателей

Создать матрицу указателей на строки. Строки разметены в динамической памяти; размеры матрицы не более 10 х 10 Конкретный размер матрицы ввести с клавиатуры. Длина строки не более 40 знаков. В функции рассортировать строки, принадлежащие главной диагонали матрицы, в порядке возрастания. Преобразованную матрицу вывести в головном модуле.

### Массив указателей

Ввести пять символьных строк, используя массив указателей на них. Найти строку, содержащую слово с наибольшим (наименьшим) числом символов. Поиск реализовать в функции в которую в качестве параметра передается указатель на строку. Найденное слово вывести на экран, начиная с последнего символа.

### #1

Каков будет результат выполнения программы? Что будет если puts(++s1)?

int main()

{

char s1[]="1234";

char \*s2="4321";

puts(++s2);

return 0;

}

[**R0012**](#R0012)

### #2

Каков результат выполнения программы?

#include "iostream"

using namespace std;

int main()

{

char buf1[100] = "Hello";

char buf2[100] = "World";

char \*ptr1 = buf1+2;

char \*ptr2 = buf2+3;

strcpy(ptr1, buf2);

strcpy(ptr2, buf1);

cout << ptr1 << endl << ptr2 << endl;

return 0;

}

[**R0013**](#R0013)

### #3

Какой будет результат при выполнении каждой строки кода, если первоначальное значение адреса указателя P равно **0028FB04**

#include <iostream>

using namespace std;

int main()

{

int mass[10]={1,2,3,4,5,6,7,8,9,10}, \*p, \*point;

p=&mass[0];

point=p;

cout << "p = " << p++ << ", \*p = " << \*p++ << endl;

cout << "\*p++ = " << \*p << ", p++ = " << p << endl;

cout << "p+3 = " << (p+3) << ", \*(p+3) = " << \*p+3 << endl;

cout << "p-mass = " << p-point << endl;

return 0;

}

[**R0052**](#R0052)

## Функция

### Доход

Написать функцию, которая вычисляет помесячный доход по вкладу. Исходными данными для функции являются: величина вклада, процентная ставка (годовых) и срок вклада (количество дней)

### Очко

Реализуйте игру "21". Действия по выдаче очередной карты игроку и компьютеру реализуйте в виде функции

### #1

Выполнится ли программа? Почему?

#include <iomanip>

using namespace std;

void function (void\* v)

{

cout << \*((int\*)v) << endl;

}

int main()

{

int n=10,\*pn;

pn = &n;

function(pn);

return 0;

}

[**R0014**](#R0014)

### #2

Каков результат выполнения программы?

#include "iostream"

using namespace std;

class money

{

int sum;

public:

money(int x) {sum=x;}

void show(void)

{

cout << "sum=" << sum << endl;

}

};

void func (money t)

{

t.show();

}

int main()

{

func(1/2+3/2+5/2);

return 0;

}

[**R0015**](#R0015)

### #3

Чему будет равны a и b? Сколько раз будет выведено на экран "func!"?

#include "iostream"

using namespace std;

int func(int x)

{

cout << "func!" << endl;

return x;

}

int main ()

{

int a,b;

cout << "a" << endl;

a=func(1),func(2);

cout << "b" << endl;

b=(func(1),func(2));

cout << "a=" << a << "; b=" << b << endl;

}

[**R0027**](#R0027)

### #4

Какое число появится на экране в результате выполнения данной программы?

#include <iostream>

using namespace std;

int main()

{

int x=5;

cout << "x = " << func\_cube(x) << endl;

return 0;

}

int func\_cube(int x)

{

return(x\*x\*x);

}

[**R0042**](#R0042)

## Класс памяти

### #1

Чему равно r?

#include "iostream"

using namespace std;

int n;

int main()

{

static int n;

int r;

r=(++::n)++/++n + n/::n;

cout << "r=" << r<< endl;

return 0;

}

[**R0016**](#R0016)

## Дополнительные приемы программирования

### #1

Правильно ли использована условная компиляция в данном примере?

#include <iostream>

using namespace std;

#define X

#define Y

#ifdef X

#define X 5

#else

#define Y 8

int main()

{

cout << "X = " << X << endl;

return 0;

}

[**R0038**](#R0038)

### #2

Какой результат будет выведен на экране?

#include <iostream>

#line 100

using namespace std;

void main()

{

cout << \_\_LINE\_\_ << endl;

}

[**R0045**](#R0045)

## Структура

### #1

Чему будет равен последний элемент массива?

using namespace std;

struct st1

{

int n;

int mas[2];

};

int main()

{

st1 a = {1,2};

cout << "a.mas[1]=" << a.mas[1] << endl;

return 0;

}

[**R0017**](#R0017)

## Class

### #1

Выполнится данная программа или нет? Почему?

#include "iostream"

using namespace std;

class Class1

{

public:

int x;

Class1() :x(1) {}

};

class Class2 :private Class1

{

public:

using Class1 :: x;

};

int main()

{

Class2 d;

cout << "x=" << d.x << endl;

return 0;

}

[**R0018**](#R0018)

### #2

Полностью ли освободится память? Если нет, то как исправить это?

#include "iostream"

using namespace std;

class base

{

public:

~base() { cout << "base is deleted!" << endl; }

};

class derv :public base

{

public:

~derv() { cout << "derv is deleted!" << endl; }

};

int main()

{

base\* a = new derv;

delete a;

return 0;

}

[**R0019**](#R0019)

### #3

Выполнится ли данная программа? Если да, то почему? Если нет, исправьте её.

#include "iostream"

using namespace std;

class A

{

public:

void f(void) { cout << "A" << endl; }

};

class B : public A

{};

class C : public A

{};

class D : public B, public C

{};

int main()

{

D x;

x.f();

}

[**R0020**](#R0020)

### #4

В каком порядке вызовутся конструкторы и деструкторы? Почему?

#include "iostream"

using namespace std;

class A

{

public:

A() { cout << "Konstructor A" << endl; }

~A() { cout << "Destructor A" << endl; }

};

class B

{

public:

B() { cout << "Konstructor B" << endl; }

~B() { cout << "Destructor B" << endl; }

};

int main()

{

A x; B y;

return 0;

}

[**R0021**](#R0021)

### #5

Какая из функций show() вызывается?

#include "iostream"

using namespace std;

class base

{

public:

void show(void) { cout << "base" << endl; }

};

class derv : public base

{

public:

void show(void) { cout << "derv" << endl; }

};

int main()

{

derv a;

base\* p;

p=&a;

p->show();

return 0;

}

[**R0022**](#R0022)

### #6

Будет ли работать данная программа?

#include "iostream"

using namespace std;

class num

{

int a,b,c;

float avg;

public:

num& intput(int x, int y, int z) { a=x; b=y; c=z; return \*this; }

num& average(void) { avg=(a+b+c)/3.0; return \*this; }

num& output(void) { cout << "Average=" << avg << endl; return \*this; }

};

int main()

{

num t;

t.intput(1,2,3).average().output();

return 0;

}

[**R0023**](#R0023)

### #7

Будет ли правильной строчка cout << "n=" << num::n << endl; в этой программе? Почему?

#include "iostream"

using namespace std;

class num

{

public:

static int n;

void count(void) { cout << "n=" << ++n << endl; }

~num(void) { ++n; }

};

int num::n=0;

int main()

{

num \*t=new num;

t->count();

delete t;

cout << "n=" << num::n << endl;

return 0;

}

[**R0024**](#R0024)

### #8

Выполнится ли данная программа? Обязательна ли в программе строка: num f1 (num);?

#include "iostream"

using namespace std;

class num

{

int x;

public:

num(int t) : x(t) {}

void result(void) { cout << "x=" << x << "; x^2=" << x\*x << endl; }

};

num f1 (num);

num f1 (num t = f1(8))

{

return t;

}

int main()

{

f1().result();

f1(9).result();

return 0;

}

[**R0025**](#R0025)

### #9

Что будет выведено на экране?

#include "iostream"

using namespace std;

class A

{

A (int i) { cout << "A (int i)" << endl; }

};

class B

{

B (A a) { cout << "B (A a)" << endl; }

};

int main ()

{

int i = 1;

B b(A(i));

return 0;

}

[**R0026**](#R0026)

### #10

Какой будет результат?

#include "iostream"

using namespace std;

class A

{

public:

A() { cout << "Constructor A" << endl; }

};

int main ()

{

cout << "New:" << endl;

A \*x = new A;

cout << "Malloc:" << endl;

A \*y = (A\*)malloc(sizeof(A));

}

[**R031**](#R0031)

## Решение

### R0001

Рис01_04

### R0002

Рис01_00_02

### R0003

Рис01_00_03

### R0004



Оператор return в функции digit содержит выражение, которое истинно если его значение равно true и ложно если его значение равно false. Константы true и false эквивалентны 1 и 0. Поэтому оператор return возвращает 0 если переменная ch является цифрой иначе возвращает 1.

### R0005



(x == 2) соответствует true т.е. 1, а (y+=++x) равно 2+3=5 (т.к. ++ приоритет больше чем у +=), поэтому результатом ((x == 2)&&(y+=++x)) является true (так как выражение\_1&&выражение\_2==1, только в том случае если выражение\_1≠0 и выражение\_2≠0). Таким образом условие выполняется, т.е. теперь необходимо выполнить --х,у++, здесь происходит группировка выражений слева направо (с помощью операции «запятая»), поэтому обе эти операции выполняются.

### R0006

Она выполнится без ошибок, т.к. http: это обычная метка, а всё остальное в этой строке комментарий.

### R0007

В коде есть ошибки.

**Первая** заключается в том, что используется тип unsigned int, который работает только со значениями, большими или равными нулю. Поэтому условие цикла for всегда будет истинно, и цикл будет выполняться бесконечно.

**Вторая** ошибка. Корректный код, выводящий значения всех чисел от 100 до 1, должен использовать условие i > 0. Если нам на самом деле нужно вывести нулевое значение, то следует добавить дополнительный оператор printf после цикла for.

unsigned int i;

for (i = 100; i > 0; --i)

printf("%d\n", i);

printf("%d\n", i);

**Третья** ошибка — вместо %d следует использовать %u, поскольку мы выводим целые значения без знака.

unsigned int i;

for (i = 100; i > 0; --i)

printf("%u\n", i);

### R0008



В условии цикла while переменная x в первый раз будет равна 2, т.к. вначале выполняется ++x, а потом проверяется условие. В теле цикла есть два условия, второе не выполнится никогда. Когда будет выполнятся первое условие будет осуществляться переход на следующую итерацию цикла, т.е. минуя второе условие. Переменная n подсчитывает количество итераций цикла и с учётом вышесказанного она будет равна 8. Переменная x равна 10, т.к. на последней итерации цикла она равняется 9, затем идёт проверка условия цикла, к котором переменная x в начале инкрементируется и после этого цикл закончится.

### R0009



Эта программа выводит на экран элементы массива. Глобальная переменная i автоматически инициализируется нулём. Результатом sizeof(a)/sizeof(int) является 3, т.к. sizeof(a)=12 байт (по 4байта на каждый элемент), а sizeof(int)=4 байта, таким образом 3 – это количество элементов массива. По определению операция индексирования [] интерпретируется таким образом, что a[i] идентично \*((a)+(i)), т.е. i[a] тоже верно ведь оно равно \*((a)+(i)), поэтому в нашем случае записи i++[a]=a[i++]=\*(a+i++) эквивалентные.

### R0010



Данный оператор в обоих случаях выделяет sizeof(int) байт свободной памяти и записывает адрес этой памяти в переменные-указатели x и y соответственно. При этом динамической переменной x присваивается значение 0, а y присваивает случайное значение из диапазона [0,2].

### R0011

#include<stdio.h>

#include<string.h>

#include<alloc.h>

void main(void)

{

int cmp(char\*,char\*);

char \*str[25],\*p;

inti,j=0, k, g;

do{

puts("\n Введите количество строк не более 25");

scanf("%d",&k);

}while(k>25);

fflush(stdin);

puts (''Введите строки");

for( i=0;i<k;i++)

{

str[i]=(char\*)malloc(20); // запрашиваем динамическую память

gets(str[i]); // ввод строк длиной не более двадцати знаков

if(!strcmp(str[i],"\0")) break;

j=i+1; // фактически введено строк

}

for( i=0;i<j-1;i++)

for(g=i+1;g<j;g++)

if(strcmp(str[i],slr[g])>0)

{ p=str[i]; str[i]=str[g]; str[g]=p;}

puts("Oтсopтиpoвaнные строки:");

for(i=0;i<j;i++) printf("%s\n",str[i|);

}

### R0012



Во многих случаях эти два способа определения строк эквивалентны. Но есть небольшое различие: s1 – это адрес, то есть указатель-константа, а s2 – указатель-переменная. Поэтому s2 может изменять своё значение, а s1 нет. Функция puts получает инкрементированный указатель на строку, т.е. он уже указывает на второй элемент строки. Поэтому печать строки будет уже со второго элемента. А puts(++s1) будет неверным т.к. ++s1 не возможно (т.к. s1 – константный указатель).

### R0013

Безымянный

char \*ptr1 = buf1+2;

Обьявляем указатель ptr1, который будет указывать на адрес buf1 со смещением в две единицы, т.е. указывает на первую букву "l" в слове "Hello".

char \*ptr2 = buf2+3;

Обьявляем указатель ptr2, который будет указывать на адрес buf2 со смещением на три единицы, т.е. указывает на букву "l" в слове "World".

strcpy(ptr1, buf2);

Копировать в ptr1 (который указывает на первую букву "l" в слове "Hello") из buf2. Следовательно, в ptr1 у нас будет просто слово "World". Вот здесь не нужно забывать, что ptr1 всего лишь указывает на адрес buf1 со смещением в две единицы. Именно поэтому buf1 теперь содержит результат: "HeWorld", а указатель ptr1, в свою очередь будет указывать на "World" (которое только что скопировано из buf2).

strcpy(ptr2, buf1);

Аналогичное действие. Указатель ptr2 указывает на букву "l" в слове "World". Из предыдущего действия мы выяснили, что в buf1 хранится "HeWorld". После выполнения данной операции buf2 будет содержать "WorHeWorld", а указетель ptr2 так и будет указывать на начало текста "HeWorld".

cout << ptr1 << endl << ptr2 << endl;

Указатель ptr1 указывает на "World", а указатель ptr2 указывает на "HeWorld".

### R0014



Данная программа выполнится. Функция function принимает указатель типа int и это правильно благодаря тому что в объявлении функции указан параметр типа void\*. Однако для извлечения содержимого указателей, заданных как void\*, их сначала нужно привести к конкретному типу данных (в нашем случае это int).

### R0015



Функция void func (money t) получает объект по значению. Поэтому если в качестве параметра передаётся константное выражение (или просто константа), то компилятор вычисляет его на стадии компиляции и один раз вызывает конструктор инициализации, который конструирует из полученного числового значения передаваемый в функцию объект типа money.

### R0016



Для начала необходимо сказать, что глобальная переменная n=0 и локальная статическая переменная n равна 0. Определим приоритеты всех операций выражения:

1. **::**
2. **()**
3. **Постфиксный и префиксный ++**
4. **/**
5. **+**

Исходя из этого можно записать данное выражение в виде: 1/1+1/1, поэтому ответ равен 2.

### R0017



В строке st1 a={1,2}; происходит инициализация полей структуры, т.е. n=1 и mas[0]=2, а последний элемент не инициализирован явно. При частичной инициализации структуры не инициализированные переменные принимают значение 0, поэтому a.mas[1]=0.

### R0018



Строка Class2 d означает что d будет создан как объект класса Class2. В классе Сlass2 нет конструктора и поэтому используется подходящий конструктор из базового класса Class1 т.е. x инициализируется единицей. В нашем примере происходит закрытое наследование, т.е. это значит, что public-члены класса Class1 становятся private-членами Class2, поэтому из main нельзя получить доступ к переменной x. Но благодаря тому что в Class2 есть using Class1 :: x, это открывает переменную x, вот теперь к переменной х можно обратиться из main. Исходя из вышесказанного – программа выполнится.

### R0019



В этом примере происходит удаление указателя базового класса указывающего на производный класс, а так как деструктор базового класса не виртуальный то вызовется деструктор базового класса вместо того, чтобы запустить деструктор производного класса. Таким образом это приведёт к тому что будет удалена только часть объекта, которая относится к базовому классу. Для исправления такой ситуации необходимо деструктор базового класса сделать виртуальным, т.е. вместо строки ~derv() { cout << "derv is deleted!" << endl; } необходимо написать virtual ~derv() { cout << "derv is deleted!" << endl; } и тогда получится:



### R0020



Классы В и С являются производными класса А, а класс D является производным от классов В и С. Трудности начинаются, когда объект класса D пытается воспользоваться методом класса А. Так как классы В и С содержат в себе копии метода f(), унаследованные от класса А, то компилятор не может решить, какой из методов использовать, и сообщает об ошибке. Для исправления этой неоднозначности необходимо: сделать базовый класс А виртуальным, т.е. вместо

class B : public A

{};

class C : public A

{};

Нужно:

class B : virtual public A

{};

class C : virtual public A

{};

Тогда ответ получится такой:



### R0021



Конструкторы вызвались в порядке создания объектов, а деструкторы в порядке, обратном созданию. Можно предположить, что локальные объекты хранятся в стеке, и используется LIFO-подход.

### R0022

В main() создаётся объект класса derv и указатель на класс base. Затем адрес производного класса заносится в указатель базового класса. (т.к. указатели производных классов совместимы по типу с указателями базовых классов). По результату программы видно, что компилятор не смотрит на содержимое указателя p, а выбирает тот метод который удовлетворяет типу указателя.

### R0023



Каждый из методов содержит return \*this, а так как this является указателем на объект, чей метод выполняется, то \*this - это и есть сам объект, т.е. он возвращается по ссылке t.intput(1,2,3).average().output() – это сцепленный вызов функций. Когда закончит работу метод input(1,2,3) он вернёт объект по ссылке, т.е. это будет выглядеть уже так t.average().output(), таким образом выполнятся все методы.

### R0024



Класс num содержит статическую переменную n, которая инициализирована нулём. В main() выделяем память для переменной-указателя на num, затем обращаемся к методу count(), затем удаляем объект, т.е. вызывается деструктор который увеличивает статическую переменную. Строка cout << "n=" << num::n << endl; будет верна, т.к. статическое поле существует, это объясняется тем что оно может существовать даже тогда, когда не существует ни одного объекта класса.

### R0025



В функции f1() в качестве значения по умолчанию задана константа, в виде значения возвращаемого функцией f1(), с аргументом равным 8, в этом случае она вызывает конструктор инициализации для своего формального параметра t и инициализирует х значением 8, а затем она возвращает объект и теперь вызывается конструктор копирования который копирует возвращаемый объект в формальный параметр. f1().result() – эта строка вызывает фунция f1() использует значение по умолчанию, а так как она возвращает объект класса num, то здесь возможно осуществить сцепленный вызов. f1(9).result() – вызывает функцию f1() со значением равным 9, тут опять таки срабатывает конструктор инициализации, а затем происходит сцепленный вызов метода result().

Строка num f1 (num)прототип функции и она необходима в этой программе т.к. в описании параметра функции используется сама же функция, а иначе будет ошибка такого рода:



### R0026



Из результата видно, что ни один конструктор не вызвался, т.е. это значит, что не создалось никакого объекта. A(i) эквивалентно A i, т.е. теперь видно что B b(A i) – это прототип функции которая принимает объект типа А и возвращает объект типа В и на эту строку компилятор выдал такое предупреждение:



### R0027



Операция запятая определяет в программе точку следования, т.е. это означает, что сначала имеет самый низкий приоритет,поэтому строка a=func(1),func(2) эквивалентна (a= func(1)), func(2) и поэтому a=1. Однако в этой строке func(2) также вызывается. В строке b=(func(1),func(2)) вначале выполняются действия в скобках, т.е. это значит что все функции func() будут вызваны, но поскольку операция «запятая» возвращает правый аргумент, то b=2. Исходя из вышесказанного func() была вызвана 4 раза и соответственно сообщение "func!" будет выведено 4 раза.

### R0028



Результат выделения памяти оказался разным из-за того что malloc возвращает указатель на адрес памяти, по которому может быть записано всё что угодно, т.к. мы не знаем, что там записано - для нас это "мусор" и перед использованием этой памяти нам обязательно надо ее инициализировать, т.е записать какое-то значение, а calloc сама инициализирует выделенную память нулями и возвращает на нее указатель.

### R0029

1 x++

Ответ: 5, т.к. в данном случае используется операция декремента, которая увеличивает Х на единицу после выполнения умножения.

### R0030

2 ++y

Ответ: 9 т.к. в начале произойдет увеличение на единицу, а затем умножение.

### R0031



В этом примере разница между new и malloc состоит в следующем: new сначала выделяет память, а потом вызывает конструктор, а malloc только выделяет память. Именно поэтому конструктор был вызван один раз.

### R0032

3 8на3

2, т.к. по умолчанию данная операция имеет тип integer, и вся нецелая часть числа «обрезается» и остается только двойка.

### R0033

Нет, т.к. при выделении памяти для двумерного массива, сначала выделяется память для массива указателей на одномерные массивы, а затем в цикле выделяется память под одномерные массивы. Правильный листинг программы:

#include <iostream>

using namespace std;

int main()

{

int \*\*X;

X=new int \*[5];

for(int i=0;i<5;i++)

X[i]=new int [5];

return 0;

}

### R0034

5 адреса

На экран будут выведен один и тот же адрес, но только двумя разными способами.

### R0035

6 неХ

Переменная Х будет равна 2, т.к. из-за логического НЕ цикл while не будет выполнен ни разу.

### R0036

5 rand

Те же числа, потому что нет функции srand(), из-за которой и генерируются разные числа.

### R0037



Да, допустимо. В данном примере используется так называемый вложенный макрос.

### R0038

Нет, неправильно, т.к. при использовании условной компиляции обязательно после ее окончания должна быть директива #endif

#include <iostream>

using namespace std;

#define X

#define Y

#ifdef X

#define X 5

#else

#define Y 8

#endif

int main()

{

cout << "X = " << X << endl;

return 0;

}

9 endif

### R0039



5, потому что символ ch сравниваеися не с “восьмеркой” – числом, а с восьмым символом в таблице ASCII, '5' – 53 символ, цикл while не будет выполнен ни разу.

### R0040

11 if равно

y/=2; эквивалентно y=y/2; поэтому тут Y равен 4, далее в if ч приравнивается 5 и y\*=x; эквивалентно y=y\*x; или y=4\*5; Поэтому в результате получается x=5, y=20.

### R0041

12 switch

При вызовах функции она вернет значения 1 и 0 соответственно. В операторе switch нет оператора break, хотя он нужен после каждого case. Однако в данной программе т.к. оператор return возвращает значение и выходит из функции.

### R0042

икакое, т.к. для работы данной программы нужен прототип, которого здесь нет;

#include <iostream>

using namespace std;

int func\_cube(int);

int main()

{

int x=5;

cout << "x = " << func\_cube(x) << endl;

return 0;

}

int func\_cube(int x)

{

return(x\*x\*x);

}

13 прототип

### R0043

14 сдвиг

В данном примере использован побитовый сдвиг, однако в cout, это операция вставки и записаны просто без пробелов, но в С++ пробелы используются только для удобства чтения и не обрабатываются компилятором. В printf() происходит побитовый сдвиг двоичного числа 001, получается 010 и 100 (что равно в десятичной системе: 1, 2, 4)

### R0044



В первом случае используется побитовое или. В двоичной форме числа 4 и 3 – 100 и 11, результатом побитового или является 111 (7 в десятичной). Во втором случае используется булева операция ИЛИ, если оба аргумента не равны нулю, то возвращается 1.

### R0045



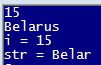
Результат будет 103, т.к. с помощью #line был определен счетчик строк, который начинается со 100.

### R0046

18 printf

Данная строка означает, что будет выведено поле шириной 10 символов, точность – 2 знака после запятой, минус – выравнивание по леврму краю, звездочки – подстановка значений 10 и 2.

### R0047



В строку str функция scanf() записывае только 5 первых символов.

### R0048

20 scanf a-z

Ответ: Она останется пустой, т.к. функция scanf() считывает только строчные символы [a-z], до ввода других символов.

### R0049

21 sscanf sprintf

Функции sscanf() и sprintf() делают преобразование типа из одного в другой.

### R0050

22 ()=

Данная запись эквивалентна c=a+b;

### R0051

23 stars

Это бесконечный цикл, т.к. переменная i идет на уменьшение и все время будет ментше 20.

### R0052

16 pointer begin

16 pointer

p=&mass[0];

//взятие адреса нулевого элемента

point=p;

//приранивание адреса нулевого элемента указателю poште

cout << "p = " << p++ << ", \*p = " << \*p++ << endl;

//вывод адреса, присвоенного переменной Р и операция разыменования

cout << "\*p++ = " << \*p << ", p++ = " << p << endl;

//P указывает на 2 элемент массива и +1 = 3, вывод адреса второго элемента

cout << "p+3 = " << (p+3) << ", \*(p+3) = " << \*p+3 << endl;

//выод адреса 5 элемента и разыменования 6 элемента массива

cout << "p-mass = " << p-point << endl;

//разность указателей

### R0053

### R0054

### R0055

### R0056

### R0057

### R0058

### R0059

### R0060

### R0061

### R0062

### R0063

### R0064

### R0065

### R0066

### R0067

### R0068

### R0069

### R0070

1. Эволюция и революция

## Второй уровень

### Третий уровень

Четвертый уровень

#### Пятый уровень

Основной текст.

* Список

1. Первый элемент списка.
2. Второй элемент списка

Рис. 1

p p p p p p p p p p p p p p p p p p