# Данные

## Целые числа

**1) Какое значение примет переменная a после выполнения операции сложения?**

int a;

a = 2147483647 + 2;

-2147483647

**2) Разместите в порядке возрастания значений переменные a, b, c, d**

int a, b, c, d;

b = 2147483645 + 1;

a = b + 1;

d = a + 1;

c = d + 1;

**dcba**

**3) Какое целое число представлено фрагментом дампа оперативной памяти (byte order little-endian)?**

ABCDEF01

Положительное

**4) Какая строка представляет собой дамп памяти, содержащей переменную a (byte order little-endian) ?**

int a = 7777;

611E0000

**5) В каком коде представляются отрицательные целые числа?**

в дополнительном

**6) В 16-ричной системе счисления 32 – разрядное целое число выглядит так: FF FF 1A 40. Что будет записано в памяти в случае Intel-архитектуры?**

40 1A FF FF

**7) В памяти записано такое число: 00 00 00 00. Это…**

0

**8) Представьте в памяти число -16**

Недостаточно сведений, чтобы записать представление

**10) Обратный порядок записи бай в память соответствует**

little-endian

## Системы счисления

**1) Основание системы счисление – это…**

Количество символов, используемых для записи числа в данной системе счисления

**2) Выберите позиционные системы счисления**

- двоичная

- шестнадцатеричная

- двенадцатиричная

**3) У позиционной системы счисления могут быть следующие основания**

- 2

- 44

**4) Для представления чисел в памяти ЭВМ используется…**

… двоичная система счисления

**5) Максимальное число, которое может быть представлено в системе счисления с** **основанием 3 – это…**

Система счисления не определяет максимальное число, представимое в данной системе

**6) Шестнадцатеричная система счисления…**

Используется для упрощения записи двоичных чисел

**8) Переведите в систему счисления с основанием 3 число 33 (10 с.с.)**

1020

**9) Переведите в систему счисления с основанием 2 число 129 (10 с.с.)**

10000001

**11) Число в десятичной системе счисления A при переводе в двоичную систему счисления дало число C. В числе C…**

количество разрядов не меньше, чем в числе A

**12) Выберите особенности машинной арифметики**

- вычисления над целыми числами дают точный результат

- два вещественных числа могут быть равны друг другу лишь с некоторой степенью точности

**13) Порядок байтов в машинной записи числа может быть**

- от старшего к младшему (big-endian)

- от младшего к старшему (litle-endian)

**14) Что такое тетрада?**

- полубайт

- четыре разряда числа, представленного в двоичной системе счисления

**15) Какие системы счисления используются в современных ПК:**

- Позиционные

- Двоичные

**16) Какое основание системы счисления является наиболее экономичным с точки зрения реализации в ЭВМ?**

3

## Действительный числа

**1) Какое действительное число представлено фрагментом дампа оперативной памяти** **(byte order little-endian) 01004080?**

Отрицательное

**2) Выберите правильные утверждения**

- Неправильный выбор типа переменной может привести к ошибке вычисления значения

- Вычисления с действительными числами имеют приближённый характер

- Выбор типа переменных влияет на время выполнения алгоритма

**3) Чему равен первый бит у нормализованной двоичной мантиссы при подготовке к записи вещественного числа в оперативную память компьютера в форматах IEEE754?**

1

**4) Может ли число 0, при записи в вещественном формате, представляться как -0 и +0? Ответьте «Может» или «Не может».**

Может

## Вещественные типы

**1) Каким будет результат выполнения фрагмента программы:**

float f = 5.;

short s = 55;

cout << sizeof((f + 3) / (2.5f - s\*3.14) )<< endl;

8

**2) Какими будут результаты выполнения фрагмента программы:**

double b, a;

b = 100000001.f - 100000001.;

a = 10000001.f - 10000001.;

cout << \"b= \" << b << '\t'<<\"a= \" << a << '\n';

b=-1 a=0

**3) Выберите типы с плавающей точкой**

- float

- long double

- double

**4) Выберите типы/тип данных с плавающей точкой:**

- float

- double

**5) Выберите верные утверждения:**

- float занимает памяти вдвое меньше, чем double

- long float занимает столько же памяти, сколько double

- float занимает в памяти 4 байта

- double занимает в памяти 8 байт

**6) Выберите описание типа float?**

Тип данных с плавающей точкой

**7) Что будет выведено в результате выполнения программы?**

int x = 3;

cout >> x / 2;

1

**8) Что будет выведено в результате выполнения программы?**

int x = 3;

cout << (float)(x / 2);

1

**9) Что будет выведено в результате выполнения программы?**

float logs = 120 / 4 \* 5;

cout << logs;

150

**10) Что будет выведено в результате выполнения программы?**

if ( 9/5 == 9.0 / 5.0 ) {

сout << “1”;

}

else {

cout << “2”;

}

2

## Текстовые данные

**3) Сколько байтов требуется для кодирования строки символов qwrf\_бгдж в UTF-8?**

13

**4) Сколько байтов требуется для кодирования строки символов qwrf\_бгдж в UTF-16?**

18

**5) Русская строчная буква щ в кодовой странице CP1251 представлена значением** **F9.Каким значением представлена прописная буква Щ ?**

D9

**6) Выберите однобайтовые кодировки из перечисленных:**

- CP10007

- KOI8-R

- CP866

- CP-1251

- ISO 8859-5

## Целочисленные типы

**1) Выберите фрагменты, в которых правильно решена задача: значение целочисленной переменной c уменьшить на 1, целочисленной переменной a присвоить частное от деления переменной c на 2.**

int a , c = 6;

- c--; a = c / 2;

- a = --c / 2;

- a = (c = c - 1) / 2;

- a=(c-=1)/2;

- a = (c -= 1) / 2.;

**2) Что будет выведено на экран после выполнения фрагмента программы:**

short a = 0xFFFF;

unsigned short b = 0xFFFF;

a >>= 4; b >>= 4;

cout <<hex<<\"a=\"<<a << '\t' << \"b=\"<<b << endl;

a=ffff b=fff

**3) Что будет выведено на экране после выполнения фрагмента программы:**

short r = 0xffff;

unsigned v = r;

cout << hex << v << '\n';

ffffffff

**4) Что будет выведено на экране после выполнения фрагмента программы:**

int k = 0x21;

short d = 0xff00;

cout << hex << k + d << '\n';

ffffff21

**5) Какие базовые целочисленные типы существуют в С++?**

- int

- char

- long

**6) Расположите базовые целочисленные типы С++ в порядке не уменьшения ширины**

- char

- short

- Int

- long

- long long

**7) Команда sizeof(int)**

… возвращает размер типа int в байтах

**9) Целочисленный тип short имеет ширину не менее (выберите наиболее подходящий** **ответ)**

16 битов

**10) Верно ли утверждение: целочисленный тип int как минимум такой же, как short;?**

Да

**11) Верно ли утверждение: целочисленный тип long имеет ширину (память, необходимая для хранения значения) не менее 32 битов и как минимум такой же, как int?**

Да

**12) Верно ли утверждение: целочисленный тип long long имеет ширину (память, необходимая для хранения значения) не менее 64 битов и как минимум такой же, как long?**

Да

**13) Какие из указанных типов данных являются целочисленными?**

- char

- int

- long long

- long

- short

- unsigned char

- unsigned int

- unsigned long long

- unsigned long

- unsigned short

**14) В каких системах счисления в языке C++ записываются целочисленные константы?**

- 8

- 16

- 2

- 10

**15) Какое значение будет иметь переменная var\_unsigned\_short после выполнения операции декремент?**

short var\_short{};

unsigned var\_unsigned\_short{};

var\_short--;

var\_unsigned\_short--;

65535

**16) Сколько байт занимает в оперативной памяти константа: '\073' (ответ дайте числом)?**

1

**17) Выберите варианты ответов, где переменная приобретает значение false?**

- bool a = 0;

- bool a = false;

**18) Какие типы данных занимают в памяти за один символ один байт?**

- bool

- char

19) Что будет выведено в результате выполнения программы?

cout << sizeof(int) << sizeof(long) << sizeof(long long) <<sizeof(short);

4482

**20) Что будет выведено в результате выполнения программы?**

bool var = true;

if (var) {

Cout << “1”;

}

if (var == 1) {

Cout << “2”;

}

if (var == 2) {

Cout << “3”;

}

12

**21) Что будет выведено на экран при выполнении фрагмента программы?**

int a = 42;

int b = 0x42;

int c = 042;

cout << a << b << c;

426634

**22) Сколько бит занимает в памяти тип char?:**

8

**23) Что будет выведено в результате выполнения программы?**

int var = 5;

var \*= 2;

var = var – 1;

var = var / 3;

var = var \* 5;

var += 2;

var = var / 3;

cout << var;

5

**24) Что будет выведено в результате выполнения программы?**

int main()

{

bool p, q;

p = false;

q = true;

cout << (p && q);

cout << (p || q);

cout << (p && q || p);

return 0;

}

010

25) Что будет выведено в результате выполнения программы?

int main()

{

bool p, q;

p = true;

q = true;

cout << (p && q);

cout << (p || q);

cout <<(p || q ) && !(p && q);

return 0;

}

110

**26) Что будет выведено в результате выполнения программы?**

char a = 65;

char b = ‘A’;

int c = 65;

if (a == b) {

cout << “1”;

}

if (a == (char)c){

cout << “2”;

}

if ( (int)b == c) {

cout << “3”;

}

123

**27) Что будет выведено в результате выполнения программы?**

void func()

{

int x;

x = 5;

cout << x;

}

int main()

{

int x=10;

func();

cout << x;

return 0;

}

510

# Введение в С++

**1) Расширения файлов исходного кода допустимые в реализациях С++:**

- схх

- cpp

- cc, c, C

- c++

- vcpp

- js

- xml

**2) Расшифруйте сокращение «ООП»:**

объектно-ориентированное программирование

**3) Корректно ли написана программа?**

#include <iostream>

int Main()

{

using namespace std;

cout << “Come up and C++ yourself some time.” << endl;

return 0;

}

Нет

**5) В каких строках программы допущены ошибки форматирования исходного кода?**

#include <iostream>

int ma in()

{

using namespace std;

cout <<

“Come up and C++ yourself some time.”

<< endl

;

return 0;

}

- 2

- 9

- 10

**6) В каких строках программы представлен прототип функции Mark?**

#include <iostream>

using namespace std;

void Mark(int n);

int main()

{

Mark(4);

return 0;

}

void Mark (int n)

{

cout << 5 - n << “points were not enough to excellent!\n”;

}

- 3

**7) Сколько символов переноса строк в следующей строке:**

\"\ttable n\n book \n\r pencil \n\r \npen\\\"

4

**8) Что можно определить по заголовку функции в С++?**

- Типы аргументов

- Количество аргументов

- Имя функции

- Тип возвращаемого значения

# Массивы

**1) Строки кода, которые позволяют установить все элементы массива в 0:**

- unsigned int counts[10] = {};

- float balances[42] {};

**2) Размер в байтах указанного массива:**

// bool – 1 byte, char – 1 byte, int – 4 bytes, long – 4 bytes

// float – 4 byte, long long – 8 bytes, double – 8 bytes

unsigned long long withoutSize[] = { 52423, 12312123123, 7777, 98765 };

32

**3) Массив – это:**

- структура данных, которая содержит множество значений, относящихся к одному и тому же типу

**4) Массив — это структура данных, которая содержит множество значений, относящихся к ….**

… одному и тому же типу данных

**5) Выберите правильные объявления массива:**

- int myArray[10];

- float myArray[10];

**6) Что правильно характеризует массив?**

- Массив содержит элементы одного и того же типа

- К элементам массива можно обращаться индивидуально

- Размер массива может быть задан только целочисленным значением

**7) Что произойдет при такой инициализации массива?**

Все элементы массива будут проинициализированы нулем

**8) Выберите фрагменты, в которых правильно решена задача: определен статический массив из 10 элементов типа int. Элементу массива с индексом 1 присвоить значение 5.**

int mas[10];

- mas[1] = 5;

- \*(mas + 1) = 5;

- \*(1 + mas) = 5;

- 1[mas] = 5;

**9) Какие фрагменты программ создания массива с инициализацией не содержат ошибок?**

- int arr1[]={1,2,3,4};

- double arr2[8]={};

- int arr4[5]={3,4};

- char arr7[4] = { 'z', 48, 'x', '\0' };

**10) Где выделяется память под динамический массив?**

в куче

**11) Какие объявления массива допустимы?**

- int a[20];

- int a[15 + 5]

- int a[5 \* sizeof(int)];

**12) Объявлен массив int a[20]; К чему приведет следующая строка: a[20]=42;?**

Нельзя сказать точно

**13) Объявлен массива int a[N] = {1, 42}; N > 10. Чему равен элемент a[5]?**

0

**14) Завершите предложение, выбрав наиболее подходящий вариант. Массив - это** **структура данных, которая содержит...**

множество значений, относящихся к одному и тому же типу.

**15) Компьютер хранит все элементы массива (выберите верные утверждения):**

- в памяти последовательно.

- в памяти друг за другом.

**16) Что означает следующий фрагмент кода** «months[12]»**? (выберите все заведомо верные утверждения):**

- Объявлен массив months из 12 элементов.

- Производится обращение к 12-му по индексу элементу массива months.

- Производится обращение к 13-му по счёту элементу массива months.

**17) Завершите предложение: Нумерация индексов массивов в языке программирования С++ начинается...**

- ...с нуля.

- ...с 0.

- ...с нулевого элемента.

**18) Объявлен массив как ... m[10], какой элемент будет последним?**

m[9]" ResponseOptions[]="m[9]

**19) Объявлен массив как int m[10], будет ли ошибка компиляции, если да, то на какой строчке(ах)?**

- m[4]+m[5]=m[6];

**20) Будут ли ошибки компиляции? Если да, то на каких строчках?**

- hand[4] = {5, 6, 7, 10};

**21) Дан фрагмент программы:** int a[50] = {3}; cout << a[3] **; Что будет выведено в результате его исполнения?**

- 0

**22) Дан фрагмент программы:** int a[50] = {1,2,3}; cout << a[3]**; Что будет выведено в результате его исполнения?**

-0

**23) Дан фрагмент программы:** int a[50] = {1}; cout << a[3]**; Что будет выведено в результате его исполнения?**

- 0

**24) Дан фрагмент программы:** int a[50] = {1,2}; cout << a[3]**; Что будет выведено в результате его исполнения?**

- 0

**25) Как выглядит общая формула объявления массива?**

имя\_типа имя\_массива[размер\_массива];

**26) Что выведет на экран следующий фрагмент программы**

long long thing[] = {1, 5, 12444 , 42, 777, 445, 11 };

int num\_element = sizeof(thing);

cout<< num\_element;

56

**27) Массив – это**

Структура данных, содержащая значения, относящиеся к одному типу данных

**28) Укажите правильные варианты объявления массива**

- int hand[4];

- int cards[4] = {3, 6, 8, 10};

**29) Объявите следующий массив и инициализируйте его первыми пятью натуральными числами через пробел: artor – массив из 5 элементов**

int artor[5] = {1, 2, 3, 4, 5};

**30) Что будет выведено на экран после выполнения фрагмента программы?**

int var = 0;

int arr[5] = { 11, 32, 31, 12, 5 };

for (int i = 0; i < size(arr); i++)

{

if (arr[i] > var)

{

var = arr[i];

}

}

cout << var;

32

**31) Что будет выведено на экран после выполнения фрагмента программы?**

int arr[5] = { 11, 32, 31, 12, 5 };

int var = arr[0];

for (int i = 0; i < size(arr); ++i)

{

if (arr[i] < var)

{

var = arr[i];

}

}

cout << var;

5

# Строки

**1) Какие объявления строк являются корректными:**

char dog[8] = {‘b’, ‘e’, ‘a’, ‘u’, ‘x’, ‘ ’, ‘I’, ‘I’};

char cat[8] = {‘f’, ‘a’, ‘t’, ‘e’, ‘s’, ‘s’, ‘a’, ‘\0’};

char bird[11] = “Mr. Cheeps”;

char fish[] = “Bubbles”;

char tiger[6] = ‘taiga’;

2, 3, 4

**2) Какая строка кода не является корректной:**

char charr1[20];

char charr2[20] = {‘j’, ‘a’, ‘g’, ‘u’, ‘a’, ‘r’,‘\0’};

string str1;

string str2 = “panther”;

charr1 = charr2;

str1 = str2;

5

**3) Работа со строками в С++ возможна через…**

- … хранение строки в массиве char

- … использование библиотечного класса string

**4) Выберите варианты ответов, в которых происходит правильная инициализация строки**

- char myString[11] = "My string!";

- char myString[] = "My string!";

- char myString[11] = {'M, 'y', ' ', 's', 't', 'r', 'i', 'n', 'g', '!','\0'};

**5) Какое значение вернет функция** strlen**?**

char myString[11] = \"My string\";

cout << strlen(myString);

9

**6) Что будет находиться в массиве name после ввода Ivan Ivanov?**

char name[30];

cout << \"Enter your name:\n\";

cin >> name;

{'I', 'v', 'a', 'n', '\0' }

**7) Каким символом должна оканчиваться строка:**

\0

**8) Объявлена переменная** char str[] = “pikabu”**; Что вернет выражение** sizeof(str)**?**

7

**9) Объявлена переменная** char str[] = “pikabu”**; Что вернет выражение** strlen(str)**?**

6

**10) Объявлена переменная** char str[] = “pikabu”**; Что выведет на экран выражение** cout << str + 1**;?**

ikabu

**11) Объявлена переменная** char str[] = R\"(we're in \"shit\")\"; **Что выведет на экран выражение** cout << str;**?**

we're in \"shit\"

**12) Выберите наиболее подходящее продолжение предложения. Строка - это...**

серия символов, сохранённая в расположенных последовательно байтах памяти.

**13) Выберите правильное написание нулевого элемента (конца строки, символа с ASCII кодом 0)?**

\0

**14) Словосочетание** “Hello world!” **может быть сохранено в символьном массиве размером n элементов. Укажите чему равно n?**

13

**15) Выберите корректное объявление строк**

- char cat[6] = { ‘f’, ‘a’, ‘t’, ‘e’, ‘s’, ‘\0'};

- char fish[] = “Bubbles”;

- сhar bird[] = “Mr.Cheeps”;

**16) Что будет выведено на экран после выполнения фрагмента программы?**

int mass1[4] = { 3, 6, 8, 10 };

char mass2[3] = { 'e', 'd', '\0'};

char mass3[] = \"12\";

std::cout << size(mass1) + size(mass2) + strlen(mass3);

9

# Указатели

**1) Что будет выведено на консоль в результате работы следующего кода:**

int \*pt = new int[10];

\*pt = 5;

pt[0] = 6;

pt [9] = 42;

printf(“%d”, \*pt);

6

**2) Чему будут равны значения переменных?**

void f(int x)

{

x = 5;

}

...

int a = 2;

int &b = a;

b = 11;

f(b);

a = 11 b = 11

**3) Какие арифметические операции можно выполнять с указателями?**

- вычитать из одного указателя другой указатель такого же типа

- складывать указатель с целым числом

- вычитать из указателя целое число

**4) Выберите фрагменты программ, не содержащие ошибок:**

- int a = 10; double b = 4.5; void \*ptr = &a; ptr=&b;

- float c = 2.8f; float \*ptrf; ptrf = &c; cout << \*ptrf;

- int x = 10, \*ptri = &x; void \*ptr = ptri; cout << ptr << '\t' << \*ptr << '\n';

**5) Выберите фрагменты программ, не содержащие ошибок:**

- enum ex1{zero,one,two,three}; int arr1[three]; double arr2[ex1(2)] = {};

- enum ex2{two=2,three =two+1,five=two+three};ex2 y = five;cout << y << endl;

- enum ex3{two=2,three =3,five=5};ex3 a=three, b, c;c = b = a;cout << a <<'\t'< (дальше ещё что-то)

**6) Что произойдет, если не вызвать delete после создания переменной с помощью операции new в свободном хранилище?**

Переменная или конструкция, динамически выделенная в области свободного хранилища, останется там, даже при условии, что память, содержащая указатель, будет освобождена в соответствии с правилами видимости и временем жизни объекта.

**7) Что будет выведено на экран после выполнения фрагмента данной программы?**

int var = 1001;

int \*pt = new int;

\*pt = var;

cout << \*pt + var;

2002

**8) Корректен ли код?**

int \*ps = new int;

delete ps;

delete ps;

Нет

**9) Корректен ли код?**

int \*ps = new int;

delete ps;

Да

**10) Корректен ли код?**

int var = 5;

int \*pi = &var;

delete pi;

Нет

**11) Корректен ли код?**

int \*pt = new int;

short \*ps = new short[500];

delete []pt;

delete ps;

Нет

# Объединения

**2) Каким будет результат выполнения фрагмента программы:**

#pragma pack(2)

struct student

{

char name[9];

int code;

};

cout << sizeof(student) << endl;

14

**3) Что будет выведено на экране после выполнения фрагмента программы:**

union example

{

short x;

char arr[3];

};

example obj, \*ptr=&obj;

obj.arr[0] = obj.arr[1] = obj.arr[2] = 0xff;

cout << ptr->x << endl;

-1

**4) Что будет выведено на экран?**

union U

{

int value;

struct Bytes

{

char a0;

char a1;

char a2;

char a3;

};

};

int main()

{

cout << sizeof(U);;

return 0;

}

4

**5) Что делает следующий код: enum spectrum {red, orange, yellow, green, blue, violet, indigo, ultraviolet};**

- Объявляет имя нового типа — spectrum; при этом spectrum называется перечислением, почти так же, как переменная struct называется структурой.

- Устанавливает red, orange, yellow и т.д. в качестве символических констант для целочисленных значений 0-7. Эти константы называются перечислителями.

# Объявление и инициализация

**1) Какие из вещественных констант записаны без ошибок?**

3.14L

3.14F

3.14

1e-2f

05E-2

**2) Какие из целочисленных констант записаны без ошибок?**

0XFULL

0x1AD

'\x00030'

'\100'

L'\0'

**3) Какие фрагменты программ определения переменной с инициализацией не содержат ошибок?**- int x = 5;

- double b = {};

- int z1{5};

- int z3(2);

- int z5=(15);

- int a=2, b=2\*a;

**4) Выберите правила, по которым в С++ можно именовать переменные**

- В именах разрешено использовать только алфавитных символов, цифр и символа подчеркивания (\_)

- В качестве имени нельзя использовать ключевое слово C++

**5) Выберите допустимые имена переменных**

- int myParam;

- int \_myParam;

- int my\_param

- int intmyParam

**6) Выберите варианты ответов, в которых происходит правильная инициализация переменной**

- int param1 = 10;

- int param1 = {};

- int param1(10);

- int param1{10};

**7) В каких системах счисления С++ позволяет записывать целочисленные константы?**

- В восьмеричной

- В десятичной

- В шестнадцатеричной

**8) Выберите правильные записи шестнадцатеричных констант**

- 0x123

- 0X123

**11) При именовании переменных (выберите верные утверждения):**

- Символы в верхнем и нижнем регистре рассматриваются как разные.

- Символы в нижнем и верхнем регистре рассматриваются как разные.

**12) Выберите допустимые имена переменных:**

- Имя переменной abc

- Имя переменной abc\_23

**13) При объявлении переменных базовых типов, выберите строки, где компилятор НЕ выдаст ошибки:**

- int terrier;

- int terrier, integer;

- int terrier, Integer;

**14) При объявлении переменных базовых типов, выберите строки, где компилятор не выдаст ошибки:**

- float flood;

- float integer, unsined;

- float unsined, integer;

**15) В какой строке производится инициализация переменной во время объявления?**

int a=3;

**16) Идентификатор это –**

любая последовательность букв A, B, C, …, Z, a, b, c, …, z, цифр 0, 1, …, 9 и символов подчеркивания, не начинающаяся с цифры

**17) Константа — это**

неизменяемое арифметическое значение целого, вещественного, символьного или перечислимого типа, нулевой указатель либо строковый литерал

**18) Указанный ниже код содержит ошибку?**

#define n 5

const int dz = 12;

void fizzle ()

{

int n;

int dz;

}

Да

**19) Данный текст содержит ошибки?**

int emus{7};

int rheas = {12};

Нет

**20) Лексемы это –**

- идентификаторы

- разделители

- ключевые слова

- константы

- операции

**22) Оператор z = x+++y; содержит ошибку?**

- Нет

**23) Определить, чему равно sim (ответь дать в шестнадцатеричной системе счисления заглавными буквами, например, AB).**

unsigned char ch = '\061', sim;

sim = ~ch;

CE

**25) Какие существуют модификаторы типов?**

- signed

- short

- long

- unsigned

**26) Заголовочный файл “climits” определяет:**

Символические константы для представления ограничений типов

**27) Какое значение присвоится переменной rocs?**

int rocs = {};

**0**

**28) Какие есть арифметические операторы в C++?:**

+

/

%

--

++

-

\*

**29) Какие есть логические операторы и операторы отношений в C++?:**

==

!=

>

<

>=

<=

||

!

&&

**30) Что такое auto в С++?**

Компилятор назначает переменной тот же самый тип, что у инициализатора

# Вывод

**1) Строка ‘\t’ – это:**

Горизонтальная табуляция

**2) Как перенести каретку на новую строку в C++ при выводе на экран?**

- std::endl

- ‘\n’

- “\n”

# Циклы

**1) Что будет выведено на экран после выполнения следующих операторов:**

#include <iostream>

int a[2] = { 1, 2 };

int b = 3;

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

std::cout << ((i < 2) ? !i ? a[i] : b : a[1]);

132

**2) Что будет выведено на экран при вызове функции?**

#include <iostream>

void foo()

{

for(int 1\_i = 0; 1\_i < 9; ++1\_i)

std::cout << 1\_i + 1;

}

Программа не будет откомпилирована из-за ошибок

**3) Какой результат будет в переменной x после выполнения следующих операторов:**

int x = 0;

int i = 2, j = 3;

for (i = 0; i < 6; ++i)

{

x += 2;

for (j = 0; j < 5; ++j)

{

int x = 0;

x += j;

}

}

12

**4) Что будет выведено на экран при выполнении следующего фрагмента кода:**

int m = 4;

while (++m < 9)

std::cout << m++;

std::cout << std::endl;

57

**5) Что будет выведено на экран при вызове функции?**

#include <iostream>

int count = 0;

void foo()

{

while (1)

{

if (count > 5)

break;

++count;

}

{

int count = 0;

while (1)

{

if (count > 20)

break;

++count;

}

}

std::cout << \"Count = \" << count << std::endl;

}

Count = 6

**6) Укажите, какой будет результат выполнения циклов:**

int m = 2, n = 5;

while (m <= 3)

{

while (m <= n)

{

n = m;

break;

}

break;

++m;

}

m = 2; n = 2;

**7) Что будет выведено на экран при выполнении следующего кода:**

#include <iostream>

int i = 5;

do

{

std::cout << (--i)-- << \" \";

}

while ( i >= 2 && i < 5 );

4 2

**8) Какое значение вернёт следующая функция:**

int foo()

{

int arr[5] = {1, 2, 3, 4, 5};

for (auto &x : arr)

{

x \*= x;

}

int result = 0;

for (auto x : arr)

{

result += x;

}

return result;

}

55

**9) Какое значение вернёт следующая функция:**

int foo()

{

int arr[5] = {1, 2, 3, 4, 5};

for (auto x : arr)

{

x \*= x;

}

int result = 0;

for (auto x : arr)

{

result += x;

}

return result;

}

15

**10) Цикл с постусловием ….**

… обязательно выполняется хотя бы один раз

**12) fог(инициализационное\_выражение; проверочное\_выражение; обновляющее\_выражение) оператор; Каких элементов цикла может не быть?**

- инициализационное\_выражение

- проверочное\_выражение

- обновляющее\_выражение

- оператор

**13) Можно ли изменить шаг цикла fог**

да, но только на целое значение

**14) С помощью какого цикла можно переписать цикл for?**

С помощью любого цикла

**15) Можно ли сделать так, чтобы тело цикла включало больше одного оператора?**

Можно, для этого применяется составной оператор

**16) Как можно работать в С++ с двумерными массивами?**

С помощью массива, каждым элементом которого является массив

**17) Оператор цикла while выполняется при 0≤x≤5, x=40 и при нечетных x, больших 50. Выберите правильное выражение-условие:**

0<=x && x <= 5 || x == 40 || x > 50 && x%2!=0

**18) После выполнения какого оператора в теле цикла возможно прекращение работы оператора цикла**

- Оператора break

- Оператора return

- Оператора goto

**19) Какой оператор for не содержит ошибок?**

- for (;;);

- for (int k = 5; k; --k);

- for (int p = 1;; ++p);

**20) Что будет выведено на экран после выполнения следующего фрагмента программы?**

int p = 1;

for (int i = 1; i < 11; ++i)

{

if (i % 4 != 0) continue;

p \*= i;

}

cout << p << '\n';

32

**21) Что будет выведено на экран после выполнения следующего фрагмента программы?**

int i = 0, s = 0;

while (1)

{

++i;

if (i > 10) break;

if (!(i % 2)) continue;

s += i;

}

cout << s << '\n';

25

**22) Задан цикл for (i = 0; i < 5; i++), что есть инициализация?**

i = 0

**23) Задан цикл for (i = 0; i < 4; i++), что есть проверочное выражение цикла?**

i < 4

**24) Задан цикл for (i = 0; i < 6; i++), выполнение цикла производится когда (выберите наиболее правильный ответ):**

проверочное выражение истинно

**25) Задан цикл for (i = 0; i < 2; i++), к какому типу приводится проверочное выражение:**

Bool

**26) Задан цикл for (i = 0; ; i++). Что произойдёт (остальная часть кода задана без ошибок)?**

Компилятор скомпилирует программу без ошибки.

**27) Задан цикл for (i = 2; ; i++). Выберите наиболее верное утверждение.**

Тело цикла выполнится хотя бы один раз.

**28) Задан цикл for (i = 0; i < j; k++) Проверочное условие не удовлетворено при первой же проверке. При этом тело цикла...**

ни разу не выполнится

**29) Задан цикл for (j = 0; i ; k++) и переменная i=0. Это означает:**

Цикл for ни разу не выполнится.

**30) Задан цикл for (i = 0; i < j; k++) Это означает:**

- Цикл for является циклом с предусловием.

- Цикл for является циклом с входным условием.

**31) Задан цикл int limit=2; for (int i = limit; i; i--). Тело цикла:**

- выполнится 2 раза

- выполнится два раза

- выполнится дважды

**32) В C++ принят стиль помещения или отсутствия пробелов между for и последующими скобками? Выберите правильное написание в соответствии со стилем.**

for (і = 6; і < 10; і++)

**33) Задан цикл for (int i = 0; i < 5; i++). Чему будет равно значение переменной i после завершения цикла?**

не определено

**34) Заданы переменная и цикл int a=0; while (a++ < 2) cout << a; Что будет выведено?**

12

**35) Заданы переменная и цикл int a=0; while (a++ < =2) cout << a; Что будет выведено?**

123

**36) Заданы переменная и цикл int a=0; while (++a < 2) cout << a; Что будет выведено?**

1

**37) Заданы переменная и цикл int a=0; while (++a <= 2) cout << a; Что будет выведено?**

12

**38) Заданы переменная и цикл int a=0; while ((++a)++ < 2) cout << a; Что будет выведено?**

2

**39) Заданы переменная и цикл int a=0; for (;a<2;a++,a++) cout << a; Что будет выведено?**

0

**40) Заданы переменная и цикл int a=0; for (;a<=2;a++,a++) cout << a; Что будет выведено?**  
02

**41) Заданы переменная и цикл int a=0; for (;a<=2;++a,a++) cout << a; Что будет выведено?**

02

**42) Заданы переменная и цикл int a=0; for (;a<=2;a++,++a) cout << a; Что будет выведено?**

02

**43) Заданы переменная и цикл int a=0; for (a++;a<=2;a++,a++,a++) cout << a; Что будет выведено?**

1

**44) Заданы переменная и цикл int a=0; for (a++;a<2;a++,a++,a++) cout << a; Что будет выведено?**

1

**45) Заданы переменная и цикл int a=0; for (a++;a<2;a++) cout << a; Что будет выведено?**

1

**46) Заданы переменная и цикл int a=0; for (a++;a<=2;a++) cout << a; Что будет выведено?**

12

**47) Заданы переменная и цикл int a=0; for (a++;a<=2;++a) cout << a; Что будет выведено?**

12

**48) Заданы переменная и цикл int a=0; for (++a;a<=2;a++) cout << a; Что будет выведено?**

12

**49) Заданы переменная и цикл int a=0; for (++a;a<2;a++) cout << a; Что будет выведено?**

1

**50) Заданы переменная и цикл int a=0; for (a++;a<2;++a) cout << a; Что будет выведено?**

1

**51) Заданы переменная и цикл int a=0; for (++a;a<2;++a) cout << a; Что будет выведено?**

1

**52) Заданы переменная и цикл int a=-3; for (a++;a++;a++) cout << a; Что будет выведено?**

-1

**53) Заданы переменная и цикл int a=-4; for (++a;++a;++a) cout << a; Что будет выведено?**

-2

**54) Заданы переменная и цикл int a=4; for (--a;--a;--a) cout << a; Что будет выведено?**

2

**55) Заданы переменная и цикл int a=3; for (a--;a--;a--) cout << a; Что будет выведено?**

1

**56) Заданы цикл for ( ; ; ) cout << 1; Что будет выведено?**

единица бесконечное число раз

**57) Заданы цикл for ( ; 2 ; ) cout << 1; Что будет выведено?**

единица бесконечное число раз

**58) Заданы переменная и цикл int a=2; do { cout << a; } while (a=0); Что будет выведено?**

2

**59) Заданы переменная и цикл int a=1; do { cout << a; } while (a--); Что будет выведено?**

10

**60) Заданы переменная и цикл int a=1; do { cout << a; } while (--a); Что будет выведено?**

1

**61) Заданы цикл do { int a=1; cout << a; } while (--a); Что будет выведено?**

Код не скомпилируется

**62) Чему будет равна переменная a после выполнения этого кода?**

int a;

for(a = 0; a < 10; a++){}

10

**63) До каких пор будут выполняться операторы в теле цикла?**

while(x < 100)

Пока x строго меньше 100

**64) Тело любого цикла выполняется до тех пор, пока его условие …?**

Истинно

**65) Выберите цикл с постусловием**

do…while

**66) Выберите циклы с предусловием**

- for

- while

**67) Что будет выведено на экран?**

int i = 0;

string str = \"Thereisheavyrain.Theweatherizterrible!\";

while (i != 7)

{

cout << str[i];

i++;

}

Thereis

**68) Что будет выведено на экран?**

int i = 0;

while (i != sizeof(int))

{

cout << i;

i++;

}

0123

**69) На каком числе остановится цикл for?**

int i = 0;

for (;;)

{

cout << i << endl;

i++;

if (i == 30)

break;

}

29

**70) На каком числе остановится цикл for?**

int i = 0;

for (;;i++)

{

cout << i << endl;

if (i == 30)

break;

}

30

**71) Что будет выведено на экран?**

char name[] = \"Hello World !\";

int i = 0;

while(name[i] != ' ')

{

cout << name[i];

i++;

}

Hello

**72) Что будет выведено на экран?**

char name[] = \"Hello World !\";

int i = 0;

while(name[i] != '\0')

{

cout << name[i];

i++;

}

Hello World !

**73) Чему равен определитель напечатанной в консоли матрицы?**

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

int matr[2][2];

int i, j;

int k = 0;

for (i = 0; i<2; i++)

{

for (j = 0; j<2; j++)

{

matr[i][j] = arr[k];

k++;

cout << matr[i][j] << \" \";

}

cout << \"\n\";

}

}

-8

**74) Есть ли нулевой столбец в напечатанной в консоли матрице?**

int arr[18] = {0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1};

int matr[3][3];

int i, j;

int k = 0;

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

{

for (j = 0; j<3; j++)

{

matr[i][j] = arr[k];

k+=2; cout << matr[i][j] << \" \";

}

cout << \"\n\";

}

Да

**75) Есть ли нулевая строка в напечатанной в консоли матрице?**

int arr[18] = {0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1};

int matr[3][3];

int i, j;

int k = 0;

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

{

for (j = 0; j<3; j++)

{

matr[i][j] = arr[k];

k+=2;

cout << matr[i][j] << \" \";

}

cout << \"\n\";

}

Нет

**76) Является ли напечатанной в консоли матрица единичной?**

int arr[18] = {0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1};

int matr[3][3];

int i, j;

int k = 0;

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

{

for (j = 0; j<3; j++)

{

matr[i][j] = arr[k];

k+=2;

cout << matr[i][j] << \" \";

}

cout << \"\n\";}

Нет

**77) Цикл while выполняется до тех пор пока:**

Условие цикла остается истинным

**78) Укажите пример правильного синтаксиса цикла for:**

for (инициализация; условие; итератор) { операторы(ы) ; }

**79) Цикл for используется если…?**

Если требуется значение индекса

**80) Какое количество итераций выполнит данный цикл for( int i = 0; i < 10; i--)?**

Бесконечное число

**81) Какое количество точек будет выведено на экран?**

using namespace std;

int j = 0;

for(int i = 1; i % 7 != 0; j++)

{

cout << “.”;

i += j;

cout << “.”;

}

8

**82) Какое количество точек будет выведено на экран?**

using namespace std;

int j = 0;

for(int i = 1; i % 8 != 0; j++)

{

cout << “.”;

i += j;

cout << “.”;

}

12

**84) За время выполнения цикла for проверка условия была произведена 8 раз. Сколько раз выполнилось тело цикла?**

7

**85) Выражения возвращающие этот тип могут являться проверочными в циклах.**

- bool

- char

- double

- int

**86) Какие куски кода, при подстановке вместо многоточия, приведут к выводу семи точек?**

using namespace std;

…

cout << “.”;

- for(int i = 8; 1 < i; i--)

**-** for(int i = 0; i < 7; i++)

**87) Какое число выводит программа?**

using namespace std;

int i = 0;

do

i++;

while(i < 3);

cout << (i, i = 1);

1

**88) Какое число выводит программа?**

using namespace std;

int i = 0;

do

i++;

while(i < 3);

cout << i, i = 1;

3

**89) Как трактуется пропуск проверочного выражения?**

True

**90) Сколько элементов двумерного массива будет выведено на экран?**

using namespace std;

int arr[5][4]={

{1, 0, 0, 0},

{0, 1, 0, 0},

{0, 0, 1, 0},

{0, 0, 1, 0},

{0, 0, 0, 1}

};

int colNum = 4;

int rawNum = 5;

for(int cntr = 0; cntr < 4; cntr++)

{

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

{

cout << arr[4 - i][i];

}

}

6

**91) Сколько элементов двумерного массива будет выведено на экран?**

using namespace std;

int arr[5][4]={

{1, 0, 0, 0},

{0, 1, 0, 0},

{0, 0, 1, 0},

{0, 0, 1, 0},

{0, 0, 0, 1}

};

for(int cntr = 0; cntr < 4; cntr++)

{

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

{

if(!arr[4 - i][i])

cout << arr[4 - i][i];

}

}

5

**92) Сколько элементов двумерного массива будет выведено на экран?**

using namespace std;

int arr[5][4]={

{1, 0, 0, 0},

{0, 1, 0, 0},

{0, 0, 1, 0},

{0, 0, 1, 0},

{0, 0, 0, 1}

};

for(int cntr = 0; cntr < 4; cntr++)

{

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

{

if(!arr[4 - i][i])

cout << arr[4 - i][i];

if(cntr != 1)

continue;

cout << arr[4 - i][i];

}

}

6

**93) Какое число будет выведено на экран?**

using namespace std;

int cntr = 0;

while(true)

{

if(cntr > 5)

break;

cntr++;

}

cout << cntr;

6

**94) Какое число будет выведено на экран?**

using namespace std;

int cntr = 0;

while(cntr < 8)

{

for(;;cntr++)

if(cntr > 5)

break;

cntr++;

}

cout << cntr;

8

# Операторы и выражения

**2) Выберите правоассоциативные операторы**

- Унарнарный оператор разыменования \*

- Унарнарный оператор инкремента ++

- Тернарнарный условный оператор ? :

- Бинарнарный оператор присваивания =

- Бинарнарный оператор присваивания +=

**3) Какое значение будет у переменной i после выполнения следующих операторов:**

int i = 0x10;

i += 010;

24

**4) Что будет выведено в консоль?**

int a = 20;

int b = 20;

cout << \"а = \" « а ++ << \": b = \" << ++ b << \"\n\";}

а++ = 20: ++Ь = 21

**8) Оператор break прекращает работу**

- Оператора цикла

- Оператора-переключателя

**9) Сколько итераций совершит цикл?**

for (int x = 0; x =3; x++);

Это бесконечный цикл

**10) Выберите верные утверждения**

- Операторы break и continue позволяют программе пропускать часть кода

- Оператор continue применяется только в циклах и вынуждает программу пропустить остаток тела цикла и сразу начать следующую итерацию

- Оператор break можно использовать в операторе switch и в любых циклах

# Операторы ветвления

**1) Выберите все корректные логические выражения, которые можно разместить в круглых скобках для проверки условия x [a,b]:**

if (...)...

- x < a || x > b

- !(x >= a && x <= b)

- !(x >= a) || !(x <= b)

**2) Что будет выведено на экран после выполнения следующих операторов:**

#include <iostream>

char c = 0xff;

if (c == 0xff)

std::cout << \"YES!\" << std::endl;

else

std::cout << \"NO!\" << std::endl;

NO

**3) Что будет выведено на экран при выполнении следующего кода:**

#include <iostream>

int k = 3;

switch(k)

{

case 2: std::cout << \"aaa\";

case 3: cout << \"bbb\";

default: cout << \"ccc\";

}

bbbccc

**4) Какое значение будет иметь переменная x после выполнения следующего кода:**

int x = 1;

switch (x)

{

case 0: x = 1;

case 1: x = 3;

case 2: x += 4;

case 3: x = 2;

default: x += 2;

}

4

**5) Что будет выведено на экран?**

int a = -10;

if(a) cout << “choice 1”

else

cout << “choice 2”

choice 1

**6) Каков результат выполнения данного фрагмента программы?**

int b=4;

if(b=5)

{ b+=2; cout<<b<<'\n';}

else

{ b-=2; cout<<b<<'\n';}

7

**7) Что будет выведено на экран после выполнения следующего фрагмента программы?**

int x = 1;

switch (x < 2 ? 3 : 5){

case 1: cout << 1 << '\n'; break;

case 2: cout << 2 << '\n'; break;

case 3: cout << 3 << '\n'; break;

case 5: cout << 5 << '\n'; break;

default: cout << \"Error!\n\";

}

3

**8) Что будет выведено на экран после выполнения следующего фрагмента программы?**

int i = 7, j = 8;

(i & j) || (j ^ i) ? cout << \"true\n\" : cout << \"false\n\";

True

**9) Каков результат выполнения данного фрагмента программы?**

int b = 15,res;

if (b < 15 || b > 100)

res = 1;

else if (!(b ^ 3))

res = 2;

else if (b / 4 == 3)

res = 3;

else

res = 4;

cout << res << '\n';

3

**10) Какой синтаксис оператора if верный?**

- if (проверочное-условие) оператор

- if (проверочное-условие) оператор1 else оператор2

- if (проверочное-условие) оператор2 else оператор1

**11) Какой синтаксис оператора if верный?**

- if (проверочное-условие) оператор

- if (проверочное-условие) оператор1 else оператор2

- if (проверочное-условие) оператор2 else оператор1

**12) Дано a=1, b=1 и фрагмент кода if (a=b) cout << 1; Что будет выведено?**

единица один раз

**13) Дано a=1, b=2 и фрагмент кода if (a=b) cout << 1; Что будет выведено?**

единица один раз

**14) Дано a=1, b=2 и фрагмент кода if (a=b) cout << a; Что будет выведено?**

2

**15) Дан код if (5==3 && 4==4 ) cout << a; Общее условие оператора if:**

Ложно

**16) Дан код if (5>3 && 5>10) cout << a; Общее условие оператора if:**

Ложно

**17) Дан код if (5>8 && 5<10) cout << a; Общее условие оператора if:**

Ложно

**18) Дан код if (5>8 && 5<2) cout << a; Общее условие оператора if:**

Ложно

**19) Дан код if (5<8 && 5>2) cout << a; Общее условие оператора if:**

Истинно

**20) Дан код if (a > 10 && a < 20 || b > 30) ...**

Логическая операция \"И\" имеет более высокий приоритет, чем логическая операция \"ИЛИ\".

**21) Дан код if (a > 10 && a < 20 || b > 30) ... Как он выполнится (обратите внимание на скобки)?**

if ((a > 10 && a < 20) || b > 30) ...

**22) Дано выражение 5 > 3 ? 20 : 10 Чему оно эквивалентно?**

20

**23) Дано выражение 5 < 3 ? 20 : 10 Чему оно эквивалентно?**

10

**24) Дано выражение 5 <= 3 ? 20 : 10 Чему оно эквивалентно?**

10

**25) Дано выражение 5 >= 3 ? 20 : 10 Чему оно эквивалентно?**

20

**27) Что будет выведено на экран после выполнения фрагмента программы?**

int a, b;

for (a = 140, b = 1742; a != b;)

{

if ( a > b)

a -=b;

else

b -= a;

}

cout << a;

2

**28) Какой из операторов является оператором ветвления?**

If

**29) Можно ли вкладывать операторы if друг в друга?**

Да

**30) Какой тип переменных используется для хранения логических данных**

Bool

**31) Какие утверждения относительно оператора switch верны?**

- Если целочисленное значение переменной count не совпадает ни с одним константным выражением, то управление передается ветви программы, содержащей зарезервированное слово default

- Оператор switch может содержать, а может не содержать зарезервированное слово default

- Если значение переменной не совпадет ни с одним константным выражением и не будет default, то программное управление перешло бы к первому оператору после switch

**32) Что будет выведено на экран после выполнения фрагмента программы?**

int a = 1;

swith (a)

{

case 1:

a++;

case 2:

a++;

case 3:

a++;

}

cout << a;

4

**33) Что будет выведено на экран после выполнения фрагмента программы?**

for (int ix = 0; ix < 10; ix++)

{

continue;

cout << ix;

}

Ничего не будет выведено

**34) Сколько чисел будет выведено на экран после выполнения фрагмента программы? (ответ дайте в виде числа)**

for ( int i = 1; i <= 70; i++)

{

if (i % 7 != 0)

{

continue;

}

cout << i << endl;

}

return 0;

10

**35) Что будет выведено на экран после выполнения фрагмента программы?**

int count(0);

while (count < 10)

{

if (count == 5)

сontinue;

cout << count << “ “;

++count;

}

0 1 2 3 4

**36) Что будет выведено на экран после выполнения фрагмента программы?**

char ch = “I\_want\_ \_to\_ \_break\_ \_ \_ \_free.”;

int spaces = 0;

while (ch != ‘.’)

{

If (ch == ‘ ‘)

++spaces;

}

cout << spaces;

5

**37) Верно ли использован синтаксис оператора if?**

if (num < 10)

{cout << ‘a’ << endl;}

else

{cout << ‘b’ << endl;}

Да

**38) Какой вариант использования оператора if является корректным?**

If (3 == myNumber)

**39) (a>5)||(b<10). В каком случае условие будет истинно?**

- Если а > 5 и b < 10

- Если а > 5 или b < 10

**40) (a>5)&&(b<10). В каком случае условие будет истинно?**

Если а > 5 и b < 10

**41) Какой из вариантов эквивалентен выражению “if (17 <age <35)”?**

if ( (17 < age ) < 35)

**42) Выберите правильные утверждения для операции “!”?**

- Выполняет отрицание

- Обращает истинность выражения, следующего за ней

- Если А имеет значение false, то !А имеет значение true

**43) Что делает функция isalpha(ch)?**

Возвращает ненулевое значение если ch – буква, и ноль в противоположном случае

**44) Что делает функция ispunct(ch)?**

- Возвращает значение true, если ch знак препинания

- Возвращает ненулевое значение, если ch знак препинания

**45) Какая из функций относится к библиотеке cctype?**

isspace()

**46) Правильно ли используется тернарный оператор в данном примере?**

a > b ? cout << a : cout << b;

Да

**47) Какие принципы соответствуют работе оператора switch?**

- Если целочисленное значение переменной count не совпадает ни с одним константным выражением, то управление передается ветви программы, содержащей зарезервированное слово default

- Оператор switch может содержать, а может не содержать зарезервированное слово default

- Если значение переменной не совпадет ни с одним константным выражением и не будет default то программное управление в этом случае перешло бы к первому оператору после switch

**48) Какие утверждения характерны оператору switch?**

- Не позволяет обрабатывать диапазоны

- Каждая метка оператора switch должна быть представлена одиночным значением

- Метка case должна быть константой

**49) int А = 5 > 3 ? 10 : 12 Чему будет равно значение переменной А?**

- 10

**50) Какой служебный знак ставится после оператора множественного выбора switch?**

:

# Использование функций

**1) Отметьте, каким определениям функции может соответствовать вызов func(5.98):**

- void func(double x);

- double func(double x, int y = 12, int z = 5);

- double func(double x, int k = 5);

**2) Что будет выведено на экран при вызове функции f3?**

int f1(int &x) { return x += 5; }

int f2(int \*y) { return \*y = \*y + 1; }

void f3()

{

int a = 7, \*k = new int;

\*k = f1(a);

std::cout << a << \" \" << \*k << \" \" << f2(k);

delete k;

}

12 13 13

**3) Какое значение будет содержаться в переменной result после возврата из вызова** функции f2(10, 2, result)?

void f2(int x, int n, int y)

{

int i;

for(i = 0, y = 1; i < n; ++i)

y = y \* x;

}

результат не определен

**4) В фрагменте программы определены две функции и массив указателей на функции с инициализацией. Как можно вызвать функцию f2() с помощью элемента pa[1] массива указателей на функции**

void f1()

{

cout << \"f1()\n\";

}

void f2()

{

cout << \"f2()\n\";

}

int main()

{

void(\*pa[2])() = { f1, f2 };

}

- pa[1]();

- (\*pa[1]();

**5) Что будет выведено на экран после выполнения фрагмента программы**

double& Max(double &a, double &b)

{

return a>b?a:b;

}

int main()

{

double t=8.7,r=5.3;

Max(t,r)=45.5;

cout << t << \" \" << r << '\n';

return 0;

}

45.5 5.3

**6) Что будет выведено на экран после выполнения фрагмента программы**

int fun(int \*begin,int \*end)

{int p = 1;

for (int \*ptr = begin; ptr != end; ++ptr)

p \*= \*ptr;

return p;

}

int main()

{

int arr[10] = { 1, 2, 3, 4, 5 };

cout << fun2(arr + 2, arr + 5) << endl;

return 0;

}

60

**7) После какого ввода функция func закончит своё выполнение?**

using namespace std;

struct Coordinates

{

int x;

int y;

};

void func(Coordinates structure)

{

while(cin >> structure.x >> structure.y)

cout << 1;

}

- После ввода значений несоответствующих требуемым

- После ввода строки “false 1”

**8) Какую строку программа выведет в консоль?**

#include <iostream>

#include <string>

using namespace std;

void function(string str1, string str2)

{

str1 = str1 + str2;

}

int main()

{

string a(\"Hello\"), b(\"world!\");

function(a, b);

cout << a;

cout << a.empty();

}

Hello0

**9) Какую строку программа выведет в консоль?**

int main()

{

array<string, 3> strArr = {\"a\", \"b\", \"c\"};

for\_each(strArr.begin(), strArr.end(), [](string &a){a = a + \"a\";});

cout << strArr[2];

}

Ca

**10) Какую строку выведет программа в консоль?**

#include <iostream>

#include <string>

using namespace std;

void function(string str1, string str2)

{

str1 = str1 + str2;

}

int main()

{

string a(\"Hello\"), b(\"World!\");

function(a, b);

cout << a;

}

Hello

**11) Что выведет программа?**

#include <iostream>

#include <string>

#include <array>

using namespace std;

int main()

{

array<string, 3> strArr = {\"a\", \"b\", \"c\"};

strArr.fill(\"7\");

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

{

cout << strArr[i];

}

}

777

**12) Что выведет программа?**

#include <iostream>

#include <string>

#include <array>

using namespace std;

int main()

{

array<string, 0> strArr;

cout << strArr.empty();

}

1

**13) Какое значение будет присвоено переменной X?**

int foo(int a, int b)

{

if (a > b)

return a;

if (a < b)

return b;

else

return 0;

}

int X = foo(7,5);

7

**14) Какое значение будет присвоено переменной X?**

double cube(double x)

{

return x\*x\*x;

}

int val = 10;

double X = cube(val);

1000

**15) Корректен ли код?**

const float g\_earth = 9.80;

const float \* pe = &g\_earth;

Да

**16) Корректен ли код?**

int age = 39;

const int\* pt = &age;

\*pt = 20;

age = 20;

Нет

**17) Корректен ли код?**

const float g\_moon = 1.63;

float \* pm = &g\_moon;

Нет

**18) Какие из следующих операций корректны?**

const int \*\*pp2;

int \*p1;

const int n = 13;

- \*pp2 = &n;

- \*p1 = 10;

**19) Выберите верные утверждения:**

int sloth = 3;

const int \* ps = &sloth;

int \* const finger = &sloth;

- ps – указатель на const int

- finger – const-указатель на int

**20) Указан прототип функции pam(). Выберите верное объявление соответствующего типа указателя(на функцию):**

double pam(int);

double (\*pf) (int);

**21) Выберите верные утверждения:**

double ned(double);

int ted(int);

double(\*pf)(int);

pf = ned;

pf = ted;

- pf = ned; операция выполнится некорректно

- pf = ted; операция выполнится некорректно

**22) Выберите верные утверждения:**

1.double pam(int);

2.double(\*pf)(int);

3.pf = pam;

4.double x = pam(4);

5.double y = (\*pf)(5);

- 1 строка – объявление прототипа функции

- 2 строка – объявление указателя на функцию

- 4 строка – вызывается функция pam() по имени

- 4 строка – вызывается функция pam() по имени

- 5 строка – вызывается функция pam(), используя указатель

# Использование Функций. Общие понятия

**1) Сколько функций может быть в программе С++?**

минимум одна

**2) Может ли в С++ функция определяться внутри другой функции?**

в языке С++ это недопустимо

**3) Прототип функции задаёт**

тип функции, включая тип результата, а также количество, порядок и типы аргументов

**4) Отметьте истинное высказывание, если вызываются подряд несколько функций:**

после выполнения одной функции управление переходит к следующей

**5) Отметьте все истинные высказывания:**

- если функция не принимает никаких параметров, то скобки или оставляются пустыми, или в них записывается ключевое слово void.

**6) Если функция имеет тип void, то неверно, что**

- она не может содержать оператор return

- она не может содержать несколько операторов return

**7) Запись ::func(x) означает, что**

вызывается функция из глобального контекста

**8) Укажите правильные объявления функции:**

- double Average2(int arr[], int size);

- float Seg(double w, char k, int x = 0);

- bool IsSorted(double \*arr, int size);

**9) Какие определения функций не содержат ошибок?**

- double fun1(double a,double b,double){return a\*b;}

- int fun3(int a, int b=2){return a+b;}

- void fun8(){ cout<< \"fun8\n\";}

**10) По каким причинам функция со спецификатором inline может трактоваться как обычная функция?**

- функция является рекурсивной

- функция имеет слишком большой размер, чтобы сделать подстановку

- функция, которая вызывается в выражении более одного раза

- функция, у которой вызов размещается до ее определения

- функция с переменным числом аргументов

- функция, содержащая циклы, переключатели и операторы переходов

**12) Зачем нужны прототипы функции?**

- Прототип описывает интерфейс функции для компилятора.

- Прототип сообщает компилятору какие аргументы и какого типа должны передаваться в функцию.

**13) Дан код void n(int n);**

- Это прототипирование функции.

**14) Задан прототип void viod(int);**

С точки зрения обработки кода компилятором ошибки нет.

**15) Задана функция think(), получите её адрес**

&think

**16) Выберите задачи, выполняемые компилятором:**

- Проверяет, правильно ли указано количество аргументов

- Корректно обрабатывает возвращаемое значение

- Проверяет правильность типов аргументов

**17) Переменная, объявленная в заголовке функции и используемый в теле функции:**

- Формальный аргумент

- Формальный параметр

**18) Какой вариант объявления функции является верным?**

void function(float, char);

**19) Какие из перечисленных вариантов записи функции main не являются стандартом?**

void main() {}

**20) Что из нижеперечисленного не является прототипом функции?**

- double funct (char x){}

**21) Можно ли перегрузить функцию main?**

- Нет

**22) Какие возможны варианты представления строки в языке С?**

- Массив char

- Указатель char, установленный в адрес начала строки

**23) Плюсы прототипирования функций заключаются в:**

- Корректной обработке возвращаемого значения компилятором

- Проверке правильности типов аргументов. Если тип не подходит, компилятор преобразует его в правильный, когда это возможно.

**24) Если у функции НЕ задан прототип, то:**

- При ее вызове с количеством аргументов большим, чем при описании, компилятор «проглотит» ошибку

- При несоответствии числовых типов аргументов, типы могут быть неверно преобразованы

# Аргументы функций

**1) В чем разница между фактическими и формальными параметрами функции?**

формальные параметры задаются в определении функции, а фактические – значения параметров, используемые при вызове функции

**2) В чём заключаются недостатки при использовании передачи аргументов в функцию по значению?**

- затраты времени на копирование значений и затраты памяти для хранения копий при передаче больших объёмов данных

- невозможность изменения передаваемых данных

**3) Какими способами передаются аргументы в функцию?**

- передача по значению

- передача по ссылке

- передача с помощью указателя

**4) В функцию n\_chars() передаётся два параметра типа int, как правильно разделить аргументы (объявленные переменные a и b) при вызове функции?**

n\_chars(a , b) ;

**5) В функцию b() передаётся переменная a типа int;**

Функция работает с создаваемой копией переменной.

**6) В функцию b() передаётся массив a типа int;**

Функция работает с оригиналом.

**7) При необходимости передачи структуры в функцию может передаваться:**

Допустимы оба варианта

**8) При необходимости передачи структуры в функцию может передаваться:**

- Непосредственно структура

- Адрес структуры

**9) В прототипе функции имена переменных … пропускать.**

- Разрешается

**10) Переменная, которая используется для приема переданного значения, называется:**

Формальным аргументом (формальным параметром)

**11) Значение, переданное функции называется:**

Фактическим аргументом (фактическим параметром)

**12) Влияет ли изменение значений формального параметра внутри функции на данные вызывающей программы?**

Нет

**13) Когда функция вызывается, компьютер выделяет память, необходимую для ЭТИХ переменных. Когда функция завершается, компьютер освобождает память, которая была использована ЭТИМИ переменными. ЭТИ переменные называются:**

Локальными переменными

**14) Комбинированное объявление аргументов функций…**

Не допускается

**15) Верно ли следующее объявление аргументов функции?**

Нет

**16) Верно ли следующее объявление аргументов функции?**

Да

**17) Первый аргумент в функции sum\_arr является:**

int sum\_arr(int arr[], int n)

Указателем

**18) Идентичны ли следующие прототипы функций?**

int sum\_arr(int arr[], int n);//1

int sum\_arr(int\* arr, int n);//2

Да

**19) Идентичны ли параметры при следующих вызовах функций?**

int sum\_arr(cookies + 4, 4);//1

int sum\_arr(cookies[4], 4);//2

Нет

**20) Корректны ли следующие прототипы?**

void sum\_arr(int arr[], int size);//1

int sum\_arr(int arr[size]);//2

Нет

# Возврат из функций

**1) Какие из следующих утверждений об операторе return являются верными?**

- Оператор return завершает выполнение функции

- В теле функции может присутствовать несколько операторов return

**2) Как определяется функция, не возвращающая значений?**

void имяФункции(списокПараметров)...

**3) Объявлена функция void n(int n)..., выберите верное утверждение**

функция n() ожидает передачи значения типа int

**4) Функции с возвращаемыми значениями требуют использования**

оператора return

**5) Если объявленным возвращаемым типом функции указан double, а функция возвращает выражение int, то**

то int приводится к double

**6) Язык C++ накладывает ограничения на типы возвращаемых значений с помощью return:**

- возвращаемое значение не может быть массивом

**7) После выполнения оператора return:**

- функция завершается

**8) Функция:**

- может возвращать тип ранее объявленной структуры

**9) Выберите верное утверждение:**

Функция может возвращать адрес структуры.

**10) Функции, не возвращающие значений, имеют тип возвращаемого значения:**

Void

**11) Что произойдет, если функция будет содержать более одного оператора return?**

Функция прекратит свою работу при достижении первого оператора return

**12) Укажите тип возвращаемого значения функции int \* func(char &sym, float\* matrix);**

- адрес int переменной, возвращаемой в return

**13) Какую строку вернет функция function, если аргумент с будет равен символу «f», а n равен 7?**

char \*function(char c, int n)

{

char \*str = new char[n+1];

str[n] = ‘\0’;

while(n-- > 0)

str[n] = c;

return str;

}

fffffff

**14) Какое число вернёт функция calcTotalCost при получении в качестве аргумента результата выполнения функции niceFunc()?**

struct Product

{

int \*price;

int number;

};

Product niceFunc()

{

int \*arr = new int [10];

for(int i = 0; i < 10; ++i)

{

arr[i] = i + 1;

}

Product a = {arr, 10};

return a;

}

int calcTotalCost(Product prod)

{

int cost = 0;

for(int i = 0; i < prod.number; i++)

{

cost += prod.price[i];

}

return cost;

}

55

**15) Функции могут возвращать:**

- Целые числа

- Числа с плавающей точкой

- Указатели

- Структуры

- Объекты

**16) Функция в C++ завершается после выполнения оператора:**

Return

**17) Пусть think() – некая функция. Укажите верные утверждения:**

- process(think) – передача адреса think() функции process()

- thought(think()) – передача возвращаемого значения think() функции thought()

# Режим файла. Бинарные файлы

**1) С помощью чего определяется режим открытия файла?**

С помощью констант режима файла

**3) Расположите в правильном порядке константы при их использовании в методе open()**

ios\_base::in

ios\_base::out

ios\_base::app

Любой

**5) Вспомнив особенности класса fstream, ответьте, где именно перемещается указатель вывода методом seekp()?**

В соответствующем буфере

**6) Что означает произвольный доступ?**

Возможность перемещения в любую позицию файла

**7) Так случилось, что Вам необходимо узнать текущее положение указателя ввода и сдвинуть его на 2 позиции влево, если это возможно. Какие два метода Вам точно понадобятся?**

- tellg()

- seekg()

# Общие понятия io

**1) "Завершите фразу: \"Стандартный пакет ввода-вывода\" (\"Standard Input/Output package\")...**

...есть неотъемлемая составная часть стандартной библиотеки С.

**2) В языке С++:**

сіn и istream — это не ключевые слова.

**3) Завершите фразу: Программа на C++ воспринимает ввод и вывод...**

...как потоки байтов.

**4) Выберите верное определение.**

Буфер — это блок памяти, используемый в качестве промежуточного временного хранилища при передаче информации от устройства в программу или из программы устройству.

**5) Завершите предложение. Для обработки вывода применяется такой объект ostream, как...**

Cout

**6) Завершите предложение. Создание объекта ostream...**

- открывает поток;

- автоматически создаёт буфер;

- ассоциирует буфер с потоком;

**7) Выберите подходящие утверждения. Объект cout с операцией << также называется:**

- операцией вставки;

# Файловый ввод-вывод

**3) Если файл с таким именем существует, что произойдёт с ним после команды open()?**

Станет доступен для записи, но он будет усечён так, чтобы запись осуществлялась в пустой файл

**4) Что позволяет проверить состояние потока при работе с файлами?**

- Использование метода is\_open()

- Проверка состояния методами класса iostream

**5) Выберите строки строго необходимые для создания программы осуществляющей запись строки «Dull Data» в файл «jar.txt»**

- ofstream fout;

- fout.open(“jar.txt”);

- fout << “Dull Data”;

**6) Выберите строки, содержащие ошибки:**

- ofstream str(“jar.txt”) << “abc”;

- ofstream bool(“question.txt”);

- iostream temp(“file.txt”);

**7) Выберите строку, которая откроет для потока fout файл «JJ.txt» в двоичном виде для записи, перейдя к концу файла после его открытия**

fout.open(“JJ.txt”, ios\_base :: binary | ios\_base :: ate | ios\_base :: out)

**8) Какие методы из перечисленных возвращают текущую позицию файлового указателя?**

- tellg()

- tellp()

**9) Выберите утверждения относящиеся к функции tmpnam():**

- Создает уникальное имя временного файла

- Константа TMP\_MAX устанавливает максимальное число вызовов функции без повторения имен

- Генерируемые имена файлов зависят от компилятора

# Ostream. Форматированный вывод

**1) Что выведется на экран? (одна из характеристик данных)**

**Приведён фрагмент кода программы**

char \* amount = \"dozen\";

cout<<(void \*) amount;

Адрес строки

**2) Почему возможна конкатенация вывода? (например cout.put(‘ ‘).put(‘ ‘))**

Потому что put() возвращает ссылку на вызывающий объект cout

**4) Если выбрано шестнадцатеричное представление выводимых данных, то какую максимальную ширину поля можно установить?**

Ширина ограничена техническими характеристиками используемого устройства

**5) Приведён фрагмент кода**

#include <iostream>

int main()

{

using std::cout;

using std::ios\_base;

float price1=20.40;

cout.setf(ios\_base::showpoint);

cout.precision(2);

cout<<price1;…

**20.**

**7) Сколько способов установки флагов существует?**

2: с помощью функции setf() и operator<<() вместе с манипулятором

# Модели памяти, пространства имен

**1) Сколько схем хранения данных в C++11?**

Четыре

**2) Какая схема хранения данных впервые появилась в C++11?**

Потоковая продолжительность хранения

**3) Может ли в функции объявлено несколько переменных с одинаковым именем?**

Да

**4) Может ли в файле исходного кода объявлено несколько глобальных переменных с одинаковым именем?**

Да

**5) Можно ли изменять переменную-аргумент функции?**

Да

**6) Где хранятся автоматические переменные?**

В стеке

**7) Каким образом использовать одну и ту же глобальную переменную в разных файлах исходного кода?**

Объявить в одном файле исходного кода, в остальных применить ключевое слово extern

**8) Что будет выведено на экран?**

void Func()

{

static int n = 0;

cout << ++n;

}

int main()

{

Func();

Func();

return 0;

}

12

**9) От чего может уберечь использование ключевого слова volatile?**

От чтения «грязных» данных

**10) Применение ключевого слова static к функции:**

Ограничивает область видимости этой функции

**11) Что определяет применение спецификации extern “C”:**

Соглашение об именовании

**12) Можно ли разместить объект в стеке с применением оператора new:**

Можно

**13) Выберите верные утверждения:**

- Декларативная область – область в которой может производиться объявление функции

- Программа может содержать более одной декларативных областей

- Область видимости есть часть программы, в которой доступна переменная

**14) Что такое декларативная область?**

Это область, в которой могут делаться объявления

**15) С помощью какого ключевого слова создается новое пространство имен?**

Namespace

**16) Где могут находится пространства имен?**

- На глобальном уровне

- Внутри других пространств имен

**17) С помощью какого оператора можно получить доступ к именам заданного пространства имен?**

::

**18) Что происходит при объявлении using var1::var2 ?**

Добавляет в декларативную область отдельное имя var2 из пространства имен var1

**19) С помощью какой конструкции подключается пространство имен?**

using namespace

**20) Что произойдет при выполнение следующего фрагмента кода?**

namespace Jill {

double fetch;

}

char fetch;

int main ()

{

using namespace Jill;

double fetch;

cin >> fetch;

…

}

Чтение значения в локальную переменную fetch

**21) Что произойдет при выполнение следующего фрагмента кода?**

namespace Jill {

double fetch;

}

char fetch;

int main ()

{

using namespace Jill;

double fetch;

cin >> ::fetch;

…

}

Чтение значения в глобальную переменную fetch

**22) Что произойдет при выполнение следующего фрагмента кода?**

namespace Jill {

double fetch;

}

char fetch;

int main ()

{

using namespace Jill;

double fetch;

cin >> Jill::fetch;

…

}

Чтение значения в переменную Jill::fetch

**23) Что произойдет при выполнение следующего фрагмента кода?**

namespace Jill {

double fetch;

}

char fetch;

int main ()

{

using Jill;

double fetch;

cin >> fetch;

…

}

Ничего не произойдет – код с ошибкой

**24) Можно ли объявлять вложенные пространства имен?**

namespace name1

{

namespace name2

{

int a;

...

}

}

Можно

**25) Получится ли обратится к переменной var1 с помощью конструкции nameSpace1::var1?**

namespace nameSpace1

{

using nameSpace2::var1;

...

}

Получится

**26) Что произойдет после выполнения следующего фрагмента кода?**

namespace ILA = i\_love\_academy;

Для пространства i\_love\_academy имен создается псевдоним ILA

**27) Что выведется на экран после выполнения данного кода?**

#include <iostream>

using namespace std;

namespace nl

{

int x = 1;

}

namespace n2

{

int x = 2;

}

int main()

{

using namespace nl;

cout << x;

{

int x = 4;

cout << x << nl::x << n2::x;

}

using n2::x;

cout << x << endl;

return 0;

}

14122

**28) Что выведется на экран после выполнения данного кода?**

#include <iostream>

using namespace std;

namespace nl

{

int x = 2;

}

namespace n2

{

int x = 1;

}

int main()

{

using namespace nl;

cout << x;

{

int x = 2;

cout << x << nl::x << n2::x;

}

using n2::x;

cout << x << endl;

return 0;

22211

**29) Что выведется на экран после выполнения данного кода?**

#include <iostream>

using namespace std;

namespace nl

{

int x = 1;

}

namespace n2

{

int x = 1;

}

int main()

{

using namespace nl;

cout << x;

{

int x = 2;

cout << ++x << nl::x << n2::x++;

}

using n2::x;

cout << x << endl;

return 0;

13112

**30) Что выведется на экран после выполнения данного кода?**

#include <iostream>

using namespace std;

namespace nl

{

int x = 1;

}

namespace n2

{

int x = -1;

}

int main()

{

using namespace nl;

cout << ++x;

{

int x = 1;

cout << ++x << nl::x << n2::x++;

}

using n2::x;

cout << x << endl;

return 0;

}

222-10

**31) Что выведется на экран после выполнения данного кода?**

#include <iostream>

using namespace std;

void other();

namespace nl

{

int x = 1;

}

namespace n2

{

int x = 2;

}

int main()

{

using namespace nl;

cout << x;

{

int x = 1;

cout << x << nl::x << n2::x;

}

using n2::x;

cout << x;

other();

return 0;

}

void other()

{

using namespace n2;

cout << x;

{

int x = 7;

cout << ++x << nl::x << n2::x++;

}

using n2::x;

cout << x;

}

1112228123

**32) Что выведется на экран после выполнения данного кода?**

#include <iostream>

using namespace std;

void other();

namespace nl

{

int x = 5;

}

namespace n2

{

int x = 3;

}

int main()

{

using namespace nl;

cout << x;

{

int x = 1;

cout << x << nl::x << n2::x++;

}

using n2::x;

cout << ++x;

other();

return 0;

}

void other()

{

using namespace n2;

cout << x;

{

int x = 7;

cout << ++x << nl::x << n2::x++;

}

using n2::x;

cout << x;

}

5153558556

**33) Чем отличаются записи в угловых скобках(<...>) и двойных кавычках (“…”) при подключении заголовочного файла?**- Если имя файла помещено в угловые скобки, то компилятор ищет его в той части базовой файловой системы, где расположены стандартные заголовочные файлы.

- Если имя файла помещено в двойные кавычки, то компилятор ищет файл в текущем рабочем каталоге или в каталоге с исходным кодом, а затем в стандартном местоположении

**34) При включении собственного заголовочного файла, имя файла заключается в:**

“…”

**35) Какая директива используется для включения заголовочных файлов? #...**

Include

**36) В С++11 применяются следующие схемы хранения данных:**

- Автоматическая продолжительность хранения

- Статическая продолжительность хранения

- Потоковая продолжительность хранения

- Динамическая продолжительность хранения

**37) Область видимости определяет доступность имени в единице трансляции.**

Верно

**38) Область видимости определяет доступность имени в целом проекте.**

Неверно

**39) Выберите верные утверждения.**

- Имя с внешним связыванием может совместно использоваться разными файлами

- Имя с внутренним связыванием может совместно использоваться функциями внутри файла

- Статические переменные могут иметь разные области видимости в зависимости от того, как они определены

**40) По умолчанию параметры функции и переменные, объявленные внутри функции, имеют … продолжительность хранения.**

Автоматическую

**42) Слово auto в С++11 используется для…**

Автоматического определения типа компилятором

**43) Стек построен по принципу:**

Last In First Out (LIFO)

**44) Если статическая переменная не инициализирована явно, то компилятор присваивает ей значение …**

0

**45) Чтобы создать статическую переменную с внешним связыванием, нужно:**

Объявить ее вне всех блоков

**46) Чтобы создать статическую переменную с внутренним связыванием, нужно:**

Объявить ее вне всех блоков и указать модификатор класса хранения «static»

**47) Чтобы создать статическую переменную без связывания, нужно:**

Объявить ее внутри какого-либо блока, используя модификатор «static»

**48) В каких строках (какой строке) переменная инициализируется константным выражением?**

#include <cmath>

int x;

int y = 5;

long z = 13\*13;

const double pi = 4.0 \* atan(1.0);

- 3

- 4

**49) В каких строках(какой строке) переменная инициализируется динамически?**

#include <cmath>

int x;

int y = 5;

long z = 13\*13;

const double pi = 4.0 \* atan(1.0);

- 5

**50) В C++11 спецификаторами класса хранения являются:**

- register

- static

- extern

- mutable

**51) В C++11 cv-квалификаторами являются:**

- const

- volatile

**52) Динамическая память:**

- Может быть выделена с помощью оператора new

- Может быть выделена с помощью функции malloc()

**53) В каких строках код является верным?**

int\* pi = new int(6);

struct where {double x; double y; double z;};

where\* one = new where{2.5, 5.3, 7.2};

int\* ar = new int[4]{2,4,6,7};

double\* pdo = new double{99.99};

- 1

- 2

- 3

- 4

- 5

**54) Пользовательское пространство имен создается с помощью ключевого слова…**

Namespace

**55) Директива using:**

Делает доступным пространство имен в целом

**56) Объявление using:**

Добавляет в декларативную область отдельное имя

**57) Перед Вами часть кода. Вторая строка удаляется. Что нужно написать перед «cout» и «endl», чтобы программа вывела на экран «Nirvana»?**

#include <iostream>

using namespace std;

int main()

{

cout << “Nirvana” << endl;

return 0;

}

std::

# Семантика переноса и ссылки rvalue

**1) Что будет выведено на консоль в результате работы следующего кода:**

double tc = 21.5;

double && rd = 1.8 \* tc + 32.0;

printf(“rd value : %f”,rd);

70.7

**2) Корректен ли код представленный ниже:**

#include <iostream>

inline double f(double tf) {return 5.0\*(tf-32.0)/9.0;}

int main()

{

using namespace std;

double tc = 21.5;

double && rd1 = 1.8 \* tc + 32.0;

double && rd2 = f(rd2);

cout << “rd1 value and address: ” << rd1 << “, ” << &rd1 << endl;

cout << “rd2 value and address: ” << rd2 << “, ” << &rd2 << endl;

return 0;

};

Да

**Функции для работы с регулярными выражениями**

**1) Выберите основные функции для работы с регулярными выражениями:**

- regex\_match

- regex\_search

- regex\_replace

**2) Для чего используется функция regex\_match?**

- Для применения регулярного выражения ко всей последовательности символов

**3) Для чего используется функция regex\_search?**

- Для применения регулярного выражения к любой части последовательности символов

**4) Для чего используется функция regex\_replace?**

- Для замены вхождения регулярного выражения текстом замены

**5) Что вернет функция regex\_match в следующем примере:**

std::string text = “Hello world”;

std::regex reg(“Hello(\\w)\*”);

regex\_match(text.begin(), text.end(), reg);

false

**6) Что вернет функция regex\_match в следующем примере:**

std::string text = “Hello world”;

std::regex reg(“Hello (\\w)\*”);

regex\_match(text.begin(), text.end(), reg);

true

**7) Укажите в каких строках функция regex\_search() использует корректное пространство** имен:

- std::tr1::regex\_search(str.c\_str(), res, rx);

**8) Какое значение будет в переменной a1?**

0 bool a1 = false, a2 = false;

1 std::string str = “Hello world”;

2 std::tr1::regex rx(“ello”);

3 a1 = regex\_match(str.begin(), str.end(), rx);

4 a2 = regex\_ search(str.begin(), str.end(), rx);

False

**9) Какое значение будет в переменной a2?**

0 bool a1 = false, a2 = false;

1 std::string str = “Hello world”;

2 std::tr1::regex rx(“ello”);

3 a1 = regex\_match(str.begin(), str.end(), rx);

4 a2 = regex\_search(str.begin(), str.end(), rx);

True

**10) Как в С++ TR1 называется шаблон класса, который позволяет хранить совпадения,** найденные после поиска:

- match\_result

**13) Какая фраза будет храниться в переменной str2 после выполнения следующего кода:**

std::string str = “Hello world”;

std::tr1::regex rx(“ orld”);

std::string replacement = “planet”;

std::string str2 = std::tr1::regex\_replace(str, rx, replacement);

Hello world

**14) В каких строках кода допущены ошибки:**

0 std::string str = “Hello world”;

1 std::regex rx(“ orld”);

2 std::string replacement = “planet”;

3 std::string str2 = std::regex\_replace(str, rx, replacement);

- 1

- 3

**15) Выберите регулярное выражение, которое можно вставить на место пропуска (''\_\_\_\_\_'’), чтобы на экран вывелось true.**

string str = \"...\";

regex regular(\"\_\_\_\_\");

if (regex\_match(str.c\_str(), regular))

{

cout << \"true\";

}

else cout << \"false\";

- …

- .\*

- .{3}

- .{0,}

- .{1,}

- .{0,3}

- .+

- \.\.\.

**16) Выберите регулярное выражение, которое можно вставить на место пропуска (''\_\_\_\_\_'’), чтобы на экран вывелось true.**

string str = \" .\.\\\\"\";

regex regular(\"\_\_\_\_\");

if (regex\_match(str.c\_str(), regular))

{

cout << \"true\";

}

else cout << \"false\";

- .\*

- ..\\\\\\ "

- .\*\\\\"

**17) Выберите регулярное выражение, которое можно вставить на место пропуска (''\_\_\_\_\_'’), чтобы на экран вывелось true.**

string str = \"ColorColour\";

regex regular(\"\_\_\_\_\");

if (regex\_match(str.c\_str(), regular))

{

cout << \"true\";

}

else cout << \"false\";

- .\*

- (Colou?r)+

- (Colou\*r)+

- (Colou?r)\*

**18) Выберите регулярное выражение, которое можно вставить на место пропуска (''\_\_\_\_\_'’), чтобы на экран вывелось true.**

string str = \"ColorColour\";

regex regular(\"\_\_\_\_\");

if (regex\_match(str.c\_str(), regular))

{

cout << \"true\";

}

else cout << \"false\";

- .\*

- (Colou{0,}r)+

- (Colou{0,1}r)+

- (Colou?r){0,1}

- (Colou{0,5}r)+

**19) Выберите регулярное выражение, которое можно вставить на место пропуска (''\_\_\_\_\_'’), чтобы на экран вывелось true.**

string str = \"ColorColour\";

regex regular(\"\_\_\_\_\");

if (regex\_match(str.c\_str(), regular))

{

cout << \"true\";

}

else cout << \"false\";

- .\*

- (Colou{0,}r)+

- (Colou{0,1}r)+

- (Colou?r){0,1}

- (Colou{0,5}r)+

# Графические данные

**1) Черный цвет кодируется числом 000000, Белый цвет кодируется числом FFFFFF. Какой объем памяти занимает информация о цвете одного пикселя?**

- 3 байта

**2) Каким количеством байт кодируется каждый цвет, составляющий RGB-представление (красный, зеленый, синий)**

1

# Грамматика регулярных выражений

**1) Выберите несколько вариантов регулярного выражения для описания выражения** **«С++»:**

- “C\+\+”

- “C\+{2}”

**2) Выберите символ соответствующий данному числу повторений: {0, 1}:**

?

**3) Выберите число, соответствующее данному числу повторений: {7, 9}:**

от 7 до 9 раз

**4) Выберите скобки, в которых указывается класс символов:**

[]

**5) Выберите скобки, в которых указывается группа символов:**

()

**6) Какая из двух последовательностей соответствует “ab+”?**

Abb

**7) Какая из двух последовательностей соответствует “(ab)+”?**

Abab

**8) Какое регулярное выражение корректно выделит email из следующего предложения:**

- \\\w+@\\w+\\.\\w+\

- \"[a-zA-Z0-9\_]\*@[a-zA-Z]\*[:punct:]?[a-zA-Z]{2,}\"

- \"[a-zA-Z0-9\_]+@[a-zA-Z]+\\.[a-zA-Z]+\"

**9) Каким способом можно задать последовательность символов, состоящих из букв, цифр и знака подчеркивания?**

- [a-zA-Z0-9\_]\*

- \w

**10) Какая библиотека предоставляет класс регулярных выражений:**

(сломано)

**11) Каков результат выполнения данного фрагмента программы?**

std::string a = “D0gsAndC4ts”;

std::regex b(“Dogs(\\w)\*”);

if (regex\_match(a.begin(), a.end(), b))

std::cout << “Strings matched!”;

else

std::cout << “Strings didn’t match!”;

Strings didn’t match!

**12) Выберите запись, соответствующую данному регулярному выражению:**

a\\\\b

a\\b

**13) Выберите запись, соответствующую данному регулярному выражению:**

a\[F\]

a[F]

**14) Что означает символьный класс \w?**

Цифры, символы верхнего, нижнего регистра и \"\_\"

**15) Что означает символьный класс \d?**

Только цифры

**16) Что означает символьный класс \D?**

Не цифры

**17) Что означает символьный класс \s?**

Пробельные символы

**18) Что означает символьный класс \S?**

Непробельные символы

**19) Выберите регулярное выражение, эквивалентное классу символов \d?**

[0-9]

**20) Выберите регулярное выражение, эквивалентное классу символов \D?**

[^0-9]

**21) Выберите регулярное выражение, эквивалентное классу символов \w?**

[a-zA-Z0-9\_]

**22) Выберите регулярное выражение, эквивалентное классу символов \W?**

[^a-zA-Z0-9\_]

**23) Выберите регулярное выражение, эквивалентное классу символов \s?**

[ \t\n\r\f\v]

**24) Выберите регулярное выражение, эквивалентное классу символов \S?**

[^ \t\n\r\f\v]

# Основные определения

**1) Регулярные выражения – это:**

Формальный язык поиска и осуществления манипуляций с подстроками в тексте, основанный на использовании метасимволов

**2) Что означают термины BRE, ERE, ECMA, grep, egrep, awk?**

грамматики регулярных выражений

**3) Регулярное выражение — это (Выберите наиболее полное определение)**

Способ описания шаблонов для поиска текста и проверки соответствия шаблону с помощью метасимволов

**4) Метасимволы — это**

Все ответы верны

**5) Якорь – это (Выберите наиболее полное определение)**

Указатель на начало или конец чего-либо

**6) Какие виды квантификаторов существуют?**

- Ленивые

- Жадные

**7) Какие квантификаторов являются «жадными»?**

- +

- \*

**8) Какой квантификатор является «ленивым»?**

?

**9) Какие из следующих выражений являются якорями?**

- ^

- \$

**10) Что нужно вписать на место пропуска, чтобы можно было пользоваться регулярными выражениями?**

#include <\_\_\_\_\_>

Regex

**11) Какой тип переменной может хранить результат работы регулярного выражения?**

Cmatch

# Алгоритмы STL

**1) Какое значение окажется в массиве a после выполнения следующей программы?**

int a[8] = {28, 15, 18, 20, 3, 1, 2, 5};

std::partial\_sort(a, a + 3, a + 8, greater<int> ());

{28, 20, 18, 15, 3, 1, 2, 5}

**2) Что будет выведено на экран после выполнения следующего фрагмента программы?**

int b[8] = {13, 14, 25, 16, 7, 9, 10, 12};

std::cout << std::binary\_search(b, b + 8, 7);

false

**3) Что будет выведено на экран после выполнения следующего фрагмента программы?**

int b[8] = {7, 9, 10, 12, 13, 14, 16, 25};

std::cout << std::binary\_search(b, b + 8, 12);

true

**4) Выберите алгоритмы, модифицирующие последовательность:**

- generate

- unique

- remove

**5) Что будет выведено на экран после выполнения следующего фрагмента программы? (Ответ указать последовательностью чисел без пробелов)**

int b[5] = {1, 2, 3, 4, 5};

std::reverse(b, b + 5);

std::rotate(b, b + 2, b + 5);

std::reverse(b, b + 5);

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

std::cout << b[i];

45123

# Общее

**1) Контейнер — это**

Тип данных, позволяющий инкапсулировать в себе объекты других типов.

**2) Итератор – это**

Объект, который выполняет итерацию элементов в контейнере, а также предоставляет доступ к отдельным элементам

**3) Выберите виды последовательных контейнеров?**

- list

- deque

**4) Какой из типов контейнеров не относится к библиотеке STL?**

Произвольные

**5) Какой заголовочный файл можно подключить для использования типа данных string?**

(сломано, но string.h и «все ответы не верны» - неверные ответы)

**6) Какие методы содержит шаблонный класс string?**

- size

- capacity

- begin

- end

**7) Какое описание подходит классу стандартной библиотеки шаблонов vector?**

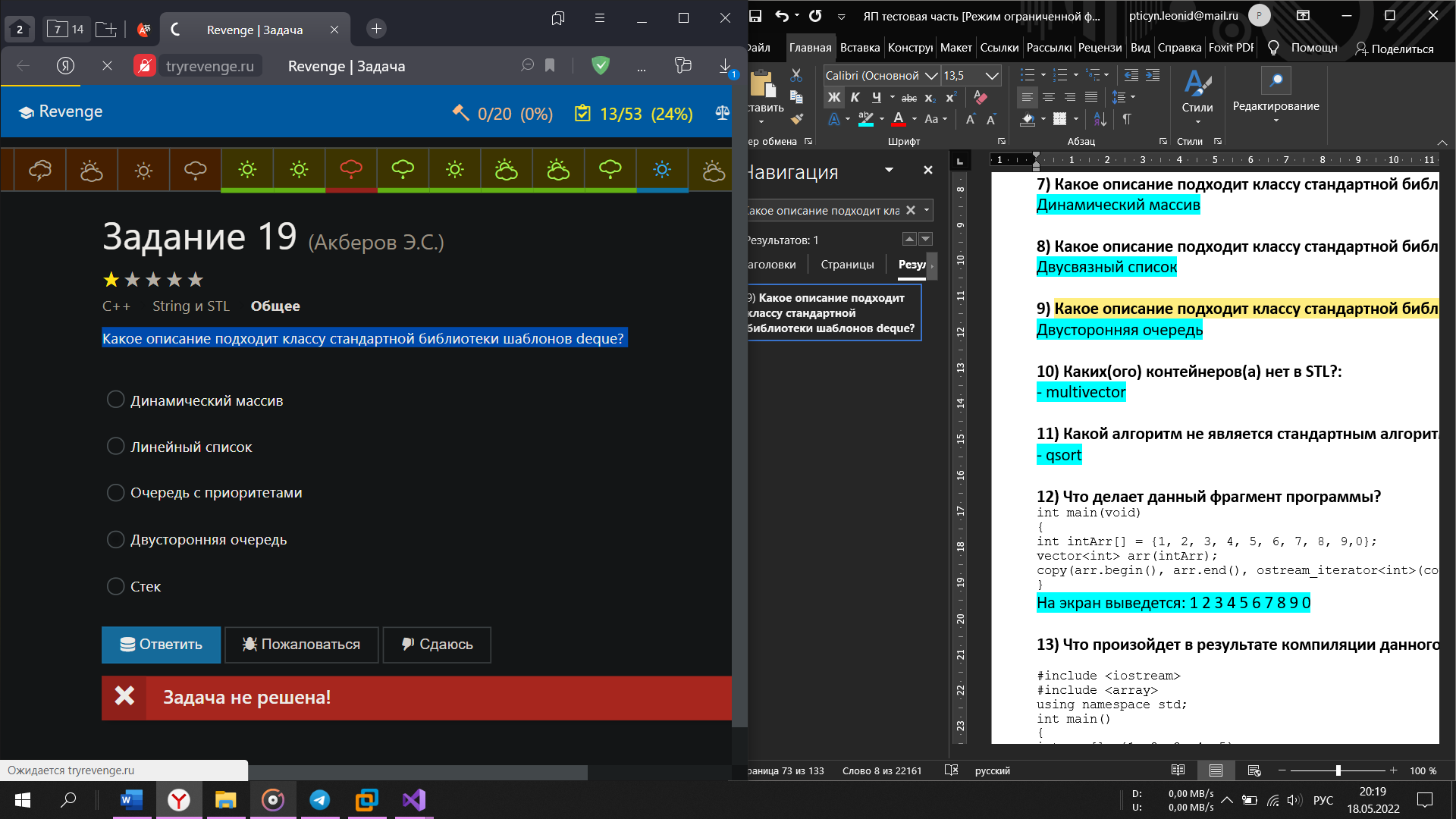
Динамический массив

**8) Какое описание подходит классу стандартной библиотеки шаблонов list?**

Двусвязный список

**9) Какое описание подходит классу стандартной библиотеки шаблонов deque?**

Двусторонняя очередь



**10) Каких(ого) контейнеров(а) нет в STL?:**

- multivector

**11) Какой алгоритм не является стандартным алгоритмом сортировки STL?**

- qsort

**12) Что делает данный фрагмент программы?**

int main(void)

{

int intArr[] = {1, 2, 3, 4, 5, 6, 7, 8, 9,0};

vector<int> arr(intArr);

copy(arr.begin(), arr.end(), ostream\_iterator<int>(cout, “ “));

}

На экран выведется: 1 2 3 4 5 6 7 8 9 0

**13) Что произойдет в результате компиляции данного кода?**

#include <iostream>

#include <array>

using namespace std;

int main()

{

int arr[] ={1, 2, 3, 4, 5};

vector<int> a(arr);

for(auto &rit: a)

++rit;

for(auto rit: a)

cout << rit;

}

Ошибка компиляции

**14) Какие интеллектуальные указатели существуют?**

- unique\_ptr

- auto\_ptr

- weak\_ptr

- shared\_ptr

**15) Какая главная особенность, использования умных указателей?**

- Не нужно беспокоиться об освобождении памяти

**16) Что будет выведено в результате выполнения программы?**

#include <iostream>

#include <vector>

#include <algorithm>

int main()

{

std::vector<int> v = { 1, 2, 3, 2, 1};

std::remove(v.begin(), v.end(), 2);

std::cout << v.size();

}

5

**17) Метод end в vector, указывает на:**

Область памяти после последнего элемента вектора

**18) STL представляет собой**

Коллекция шаблонов

**19) Является ли std::string контейнером STL**

Нет

# Контейнеры

**1) Что обычно возвращает метод begin() контейнеров STL?**

Итератор на первый элемент

**2) Что обычно возвращает метод end() контейнеров STL?**

Итератор на элемент контейнера, следующий за последним

**3) Что возвращает оператор \* итератора STL?**

Ссылку на элемент, на который указывает итератор

**4) Если c – контейнер STL, то что вернет строка \*c.end();?**

Возникнет ошибка в выполнении программы

**5) Если c – контейнер STL, то что вернет строка ++c.end()?**

Возникнет ошибка в выполнении программы

**6) Какая функция предназначена для выполнения действия над каждым элементом контейнера?**

for\_each

**7) Как отсортировать контейнер функцией sort с нестандартным условием порядка?**

Передать в функцию sort параметр – указатель на функцию сравнения (или предикат)

**9) Что появится на экране после выполнения данного кода?**

int main()

{

vector<int> vec;

for (int i = 0; i < 6; ++i)

{

vec.push\_back(i);

}

vec.pop\_back();

int result = vec.size();

cout << result;

return 0;

}

5

**10) Что выйдет на экран после выполнения данного кода?**

int main

{

vector<int> vec;

for (int i = 0; i < 8; ++i)

{

vec.push\_back(6 - i);

}

vec.pop\_back();

int result = vec.size();

cout << result;

}

7

**11) Как правильно создать шаблонный класс array?**

Array <имяТипа, количество\_Элементов> arr;

**12) Шаблонный класс vector**

Использует операции new и delete автоматически

**13) Что будет выведено на экран после выполнения фрагмента данной программы?**

vector<int> vec;

for (int i = 0; i < sizeof(int); ++i) {

vec.push\_back(1);

}

vec.pop\_back();

for (int i = 0; i < vec.size(); ++i)

{

cout << vec[i];

}

111

**14) Что будет выведено на экран после выполнения фрагмента данной программы?**

vector<int> vec;

for (int i = 0; i < sizeof(double); ++i)

{

vec.push\_back(1);

}

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

{

vec.pop\_back();

}

for (int i = 0; i < vec.size(); ++i)

{

cout << vec[i];

}

1111

**15) Сколько элементов будет в векторе vec?**

vector<int> vec;

for (int i = 0; i < 6; ++i)

{

vec.push\_back(1);

}

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

{

vec.pop\_back();

}

for (int i = 0; i < vec.size(); ++i)

{

cout << vec[i];

}

2

**16) Шаблонный класс array**

- Имеет фиксированный размер

- Использует стек

**17) Выберите верные утверждение:**

- vector – последовательный контейнер

- В stack допускается только две операции изменяющие его

**18) Выберите методы контейнера stack:**

- pop()

- push()

**20) Что выведет на экран следующая программа?**

int main ()

{

std::vector<int> v = {1, 2, 3, 2, 1};

std::remove(v.begin(), v.end(), 2);

std::cout << v.size();

}

5

**21) Контейнер list это:**

- Коллекция, хранящая элементы в виде двунаправленного связанного списка

**22) Контейнер vector это:**

Коллекция элементов, сохраненных в массиве, изменяющегося по мере необходимости размера

**23) Контейнер map это:**

Коллекция, хранящая пары вида , пара вида <ключ, значение>, при этом однозначная. Ключ – некоторая характеризующая значение величина, для которой применима операция сравнения.

**24) Контейнер set это:**

Это отсортированная коллекция одних только ключей, т.е. значений, для которых применима операция сравнения, при этом уникальных – каждый ключ может встретиться во множестве только один раз.

**25) Итератор для обхода коллекции от меньшего индекса к большему:**

(forward) iterator

**26) Итератор для обхода коллекции от большего к меньшему:**

Reverse\_iterator

**27) Итератор для коллекции в любом направлении:**

Random access iterator

# Строки

**1) Как позволяет модифицировать строку оператор +=?**

- Добавить строку в конец  
- Добавить символ в конец

**2) Чем опасен конструктор string (const char\* s, size\_type n)?**

Обращением к невыделенной области памяти

**3) Каким образом сравниваются объекты std::string?**

Сравниваются в лексикографическом порядке

**4) Объект string объявляется как**

обычная переменная

**5) Что будет выведено на экран после выполнения фрагмента программы?**

string str1 = \"H\";

string str2 = \"W\";

str2 += str1;

string str3 = str2 + str1;

cout << str3;

WHH

**6) Что будет выведено на экран после выполнения фрагмента данной программы?**

string str1 = \"abracadabra\";

cout << str1[(str1.size())-5];

d

**7) Что будет выведено на экран после выполнения фрагмента данной программы?**

string str0;

string str1 = \"good\0\";

str0 = str1;

int count = str0.size();

cout << count;

4

**8) Что будет выведено на экран после выполнения фрагмента данной программы?**

string str1 = \"ghost\";

char str2[] = \"fantom\";

int var = str1.size() \* strlen(str2);

cout << var;

30

**9) Что делает следующий код?**

cin.getline(charr, 20);

Читает строку в массив

10) Выберите верные утверждения об объектах класса srtring?

- Чтобы сохранить клавиатурный ввод объекте string, можно использовать cin

- Объект string можно инициализировать строкой в стиле C

11) Какая функция позволяет полностью очистить строку:

Оба варианта верны

12) Чем функция data() отличается от функции c\_str():

data()не добавляет нуль-символ в конец строки

13) Что будет выведено на экран после выполнения следующего фрагмента программы?

std::string str = “Running in the night”;

std::cout << str.find\_first\_of(“n”, 10);

15

14) Что будет выведено на экран после выполнения следующего фрагмента программы?

std::string s1 = “Hello world”;

std::string s2 = “friend”;

std::cout << s1.replace(6, 5, s2);

Hello friend

15) Что будет выведено на экран после выполнения следующего фрагмента программы?

std::string s1 = “Hello, world”;

std::string s2 = “I love apples”;

std::string s3 = “Is it you”

std::cout << s1.replace(6, 6, s2.replace(7, 6, s3, 6, 3));

Hello,I love you

16) Что будет выведено на экран после выполнения следующего фрагмента программы?

std::string s1 = “Octopus”;

std::string s2 = “puss”;

std::cout << s1.compare(4, 3, s2, 0, 3);

0

17) В каком случае функция compare вернет -1:

если вызывающая строка меньше

18) Выберите метод(ы) класса string для определения размера строки:

- length()

- size()

19) Чтобы присвоить одну строку другой в классе string, помимо оператора присваивание, надо использовать метод:

- assign()

20) Выберите метод(ы) вставки символа в строке:

- insert()

21) С объектами класса string можно использовать операторы:

- =

- +

- +=

- ==

- !=

- !=

- <

- <=

- >

- >=

- []

**22) Метод класса string для нахождения последнего вхождения подстроки или символа:**

rfind()

**23) Что будет выведено в результате выполнения программы?**

#include <iostream>

#include <string>

int main()

{

using namespace std;

string one(“Revenge”);

cout << one;

string two(3, ‘$’);

cout << two;

string three(two);

cout << three;

one += “2018”;

cout << one;

}

Revenge\$\$\$\$\$\$Revenge2018

**24) Что будет выведено в результате выполнения программы?**

#include <iostream>

#include <string>

int main()

{

using namespace std;

string one(“cobra”);

int where = one.find\_first\_of(“hark”);

cout << where;

where = one.find\_last\_of(“hark”);

cout << where;

}

34

**25) Что будет выведено в результате выполнения программы?**

#include <iostream>

#include <string>

int main()

{

using namespace std;

string s1 = “0123456789”;

string s2 = “345”;

int pos = s1.rfind(s2);

std::cout << pos;

}

3

**26) Что будет выведено в результате выполнения программы?**

#include <iostream>

#include <string>

int main()

{

using namespace std;

string s1 = “0123456789”;

string s2 = “345”;

int pos = s1.rfind(s2, 12);

cout << pos;

}

3

**27) Что будет выведено в результате выполнения программы?**

#include <iostream>

#include <string>

int main()

{

using namespace std;

string s1 = “0123456789”;

string s2 = “345”;

int pos = s1.rfind(s2, 2);

cout << pos;

}

-1

**28) Выберите правильное описание к перегурженному методу find**

size\_type find (const string & str, size\_type pos = 0) const;

Ищет в вызывающей строке первое вхождение подстроки str, начиная с позиции pos. Возвращает индекс первого символа найденной подстроки или string::npos, если подстрока не найдена

**29) Переменная npos – это**

Статический член класса string

# Классы

**1) Для чего нужны классы?**

- Для комбинации представления данных

- Для комбинации методов манипулирования представленных данных в пределах одного аккуратного пакета

- Для трансляции абстракций в пользовательские типы

**2) Что такое объект?**

Экземпляр класса

**3) Как принято называть совокупность открытых методов класса?**

Интерфейс

**4) Скомпилируется ли программа?**

class Foo

{

public:

int GetX()

{

return \_x;

}

void SetX(int x)

{

\_x = x;

}

private:

int \_x;

};

int main()

{

Foo foo;

return 0;

}

Да

**5) Скомпилируется ли программа?**

class Foo

{

public:

Foo(int x) : \_x(x) {}

int GetX()

{

return \_x;

}

void SetX(int x)

{

\_x = x;

}

private:

int \_x;

};

int main()

{

Foo foo;

...

return 0;

}

Нет, так как не реализован конструктор по умолчанию.

**6) Скомпилируется ли программа?**

class Foo

{

public:

Foo(int x) : \_x(x) {}

int GetX()

{

return \_x;

}

void SetX(int x)

{

\_x = x;

}

private:

int \_x;

};

int main()

{

...

}

Неизвестно (зависит от кода на месте «…»)

**7) Сколько существует разновидностей друзей у класса (дайте ответ в виде числа)?**

3

**8) Каким ключевым словом обозначается класс?**

Class

**9) Правильно ли определен класс?**

class World

{

float mass;

char name[20];

public:

void tellall(void);

…

}

Да

**10) Для чего предназначен класс? (много вариантов, 1 верный)**

- Для трансляции абстракции в пользовательский тип

**11) Напишите название модификатора доступа обозначающего, что существует возможность использовать элементы класса либо внутри непосредственно своего основного класса, либо непосредственно в своем потомке-классе.**

Protected

**12) Каким ключевым словом обозначают указатель на объект, для которого вызван метод?**

This

**13) Каким свойством обладает статический член класса? (много вариантов)**

- Создается одна копия статической переменной класса независимо от количества создаваемых объектов

**14) Сколько копий статической переменной класса создаст программа, если созданы 3 объекта класса, причем один из них унаследован от второго?**

1

**15) Спецификация класса (типа) определяет:**

- Объем памяти необходимый для размещения объекта в памяти

- Как интерпретируется выделенная под объект память

- Какие методы и операции могут использоваться для этого объекта

**16) Под классом можно понимать:**

- Данные и методы класса для манипулирования этими данными

- Данные и код функций-членов, позволяющих манипулировать этими данными

- Данные и операторы класса, позволяющих манипулировать этими данными

**17) В случае если в начале объявления тела класса с помощью ключевого слова class явно не указан спецификатор доступа, то будет использоваться спецификатор:**

Private

**18) В случае если в начале объявления тела класса с помощью ключевого слова struct явно не указан спецификатор доступа, то будет использоваться спецификатор:**

Public

**19) Список ключевых слов, позволяющих управлять доступом к членам класса**

private, protected, public

**20) Отличие объявления типа с помощью ключевого слова class и struct отличается**

- спецификатором доступа по умолчанию (private для class, public для struct)

- спецификатором доступа при наследовании

**21) Указатель this используется для: (много вариантов)**

- обращений к объекту, который использован для вызова функции-члена

**22) К указателю this можно обратиться из:**

- функции-члена (не конструктор и деструктор)

- конструктора

- деструктора

- метода

**23) При определении функции-члена для идентификации класса, которому принадлежит функция, используется оператор разрешения контекста. Запишите этот оператор:**

:: (это ответ, если что)

## Конструкторы и деструкторы

**1) Выберите правильные утверждения:**

- У конструктора могут быть параметры

- Конструктор вызывается автоматически при объявлении объекта

- Если не объявлен ни один конструктор, компилятор создаст автоматически конструктор по умолчанию

**2) Освобождает ли деструктор память, занимаемую объектом класса?**

Да

**3) Может ли деструктор иметь аргументы?**

Нет

**4) Выберите допустимые варианты объявления конструктора копирования:**

- Foo(Foo& other);

- Foo(const Foo& other);

**5) Выберите наиболее предпочтительный вариант объявления конструктора копирования:**

Foo(const Foo& other);

**6) Как можно запретить создание объекта через конструктор копирования:**

Реализовать приватный (private) конструктор копирования

**7) Чему будет равна переменная-член** \_x **объекта** foo2**:**

class Foo

{

public:

Foo () : \_x(0) {}

Foo(int x) : \_x(x) {}

Foo(const Foo& other) : \_x (other.\_x) {}

private:

int \_x;

};

int main()

{

Foo foo1(42);

Foo foo2 = foo1;

return 0;

}

42

**8) Чему будет равна переменная-член** \_x **объекта** foo2**:**

class Foo

{

public:

Foo () : \_x(0) {}

Foo(int x) : \_x(x) {}

Foo(const Foo& other) : \_x (other.\_x) {}

private:

int \_x;

};

int main()

{

Foo foo1(42);

Foo foo2(11);

foo2 = foo1;

return 0;

}

42

**9) Для чего служит ключевое слово explicit?**

Для запрета неявного преобразования некоторого типа в объект данного класса

**10) С какого символа начинается объявление деструктора?**

~

**11) Сколько конструкторов по умолчанию можно создать (ответ дайте в виде числа)?**

1

**12) Пример какого конструктора класса** World **приведен в листинге?**

class World

{

public:

World();

…

}

Конструктор по умолчанию

**13) Пример какого конструктора класса** World **приведен в листинге?**

class World

{

public:

World(int & arg1) : \_arg1(arg1) {};

…

}

Конструктор с параметрами

**14) Пример какого конструктора класса** World **приведен в листинге?**

class World

{

public:

World(World & obj){…};

…

}

Конструктор копирования

**15) Какие инициализации объекта корректны исходя из прототипа конструктора, приведенного в листинге?**

Bozo(const char \* fname, const char \* lname);

…

- Bozo bozetta = Bozo(“Bozetta”, “Biggins”);

- Bozo \*fufu = new Bozo{“Fufu”, “Dweep”};

- Bozo fufu(“Fufu”, “Dweep”);

- Bozo \*fufu = new Bozo(“Fufu”, “Dweep”);

- Bozo bozetta = Bozo{“Bozetta”, “Biggins”};

- Bozo fufu{“Fufu”, “Dweep”};

**16) Выберите наиболее подходящий прототип конструктора копирования**

- ClassName (const ClassName &)

**17) Что выведет на экран следующий фрагмент кода:**

#include <iostream>

class Klunk

{

public:

Klunk() { number = 0; }

Klunk(int num = 100) { number = num; }

void print() { std::cout << this->number; }

private:

int number;

};

void main()

{

Klunk klunk;

klunk.print();

}

- Ничего, код не скомпилируется

**18) Что выведет на экран следующий фрагмент кода:**

#include <iostream>

class Klunk

{

public:

Klunk(int num = 100) { number = num; }

void print() { std::cout << this->number; }

private:

int number;

};

void main()

{

Klunk klunk;

klunk.print();

}

- 100

**19) Что выведет на экран следующий фрагмент кода:**

#include <iostream>

class Klunk

{

public:

Klunk() { number = 0; }

Klunk(int num) { number = num; }

void print() { std::cout << this->number; }

private:

int number;

};

void main()

{

Klunk klunk;

klunk.print();

}

- 0

**20) Что выведет на экран следующий фрагмент кода:**

#include <iostream>

class Klunk

{

public:

Klunk() { number = 0; }

Klunk(int num = 100) { number = num; }

void print() { std::cout << this->number; }

private:

int number;

};

void main()

{

Klunk klunk(20);

klunk.print();

}

- 20

**21) Когда используется конструктор копирования?**

- При инициализации нового объекта уже существующим

- При передаче объекта по значению

- Когда функция возвращает объект

**22) Что выведет на экран следующий фрагмент кода?**

#include <iostream>

class Klunk

{

public:

Klunk(int num = 0) { number = num; }

Klunk(const Klunk&) { number = 10; }

void print() { std::cout << this->number; }

private:

int number;

};

void main()

{

Klunk klunk1(20);

Klunk klunk2(30);

klunk2 = klunk1;

klunk1.print();

}

- 20

**23) Что выведет на экран следующий фрагмент кода?**

#include <iostream>

class Klunk

{

public:

Klunk(int num = 0) { number = num; }

Klunk(const Klunk&) { number = 10; }

void print() { std::cout << this->number; }

private:

int number;

};

void main()

{

Klunk klunk1(20);

Klunk klunk2(klunk1);

klunk2.print();

}

-10

**24) Что выведет на экран следующий фрагмент кода?**

#include <iostream>

class Klunk

{

public:

Klunk(int num = 0) { number = num; }

void print() { std::cout << this->number; }

private:

int number;

};

void main()

{

Klunk klunk1(20);

Klunk klunk2(30);

klunk2 = klunk1;

klunk2.print();

}

- 20

**25) Что выведет на экран следующий фрагмент кода?**

#include <iostream>

class Klunk

{

public:

Klunk(int num = 0) { number = num; }

void print() { std::cout << this->number; }

private:

int number;

};

void main()

{

Klunk klunk1;

Klunk klunk2(30);

klunk2 = klunk1;

klunk2.print();

}

- 0

**26) Что выведет на экран следующий фрагмент кода?**

#include <iostream>

class Klunk

{

public:

Klunk(int num = 0) { number = num; }

Klunk(const Klunk&) { number = 10; }

void print() { std::cout << this->number; }

private:

int number;

};

void main()

{

Klunk klunk1;

Klunk klunk2;

klunk2 = klunk1;

klunk2.print();

}

- 0

**27) Чему соответствует данный прототип?**

Klunk(const Klunk&);

Конструктору копирования

**28) Что такое конструктор по умолчанию?**

Конструктор, который используется для создания объекта, когда не предоставлены явные инициализирующие значения

## Члены класса

**1) Для доступа к элементам объекта используются:**

При обращении через имя объекта – точка, при обращении через указатель – операция «->»

**4) Дано объявление класса**

class monstr {

int health, armo;

monstr(int he, int arm, int amm);

public:

monstr(int he=50, int arm=10);

int color;

}

**Укажите поля и методы, доступные внешним функциям:**

int color;monstr(int he=50, int arm=10);

**5) Выберите допустимые варианты объявления объекта класса, если в нем реализован конструктор по умолчанию и конструктор с параметром типа int:**

- Foo foo;

- Foo foo(42);

- Foo foo = {42};

- Foo foo {42};

- Foo foo = Foo(42);

**6) Класс может содержать 2 метода с одинаковым именем, если они отличаются хотя бы (выберите такие варианты, каждый из которых является правильным в независимости от других):**

- Набором переменных

- Модификатором const

**7) Что будет напечатано на экране?**

class Foo

{

public:

void Do()

{

std::cout << \"I'm Do()!\";

}

void Do() const

{

std::cout << \"I'm Do() const!\";

}

private:

int \_x;

};

int main()

{

Foo foo1;

const Foo foo2;

foo1.Do();

foo2.Do();

return 0;

}

I'm Do()!I'm Do() const!

**8) Что будет напечатано на экране?**

class Foo

{

public:

void Do() const

{

std::cout << \"I'm Do() const!\";

}

private:

int \_x;

};

int main()

{

Foo foo1;

const Foo foo2;

foo1.Do();

foo2.Do();

return 0;

}

I'm Do() const! I'm Do() const!

**9) Что будет напечатано на экране?**

class Foo

{

public:

void Do()

{

std::cout << \"I'm Do()!\";

}

private:

int \_x;

};

int main()

{

Foo foo1;

const Foo foo2;

foo1.Do();

foo2.Do();

return 0;

}

Программа не скомпилируется

**10) Необходимо реализовать оператор сложения (+) в классе Foo, какой тип возвращаемого значения должен быть?**

Foo

**11) Необходимо реализовать оператор вычитания (-) в классе Foo, какой тип возвращаемого значения должен быть?**

Foo

**12) Необходимо реализовать оператор присваивания (=) в классе Foo, какой тип возвращаемого значения должен быть?**

Foo&

**13) В классе Foo реализован оператор сложения. Будут ли одинаковыми результаты сложения двух объектов типа Foo: foo1 + foo2 и foo2 + foo1?**

Зависит от реализации оператора

**14) В классе Foo реализован оператор вычитания. Будут ли одинаковыми результаты сложения двух объектов типа Foo: foo1 - foo2 и foo2 - foo1?**

Зависит от реализации оператора

**15) Какие из следующих операций можно перегрузить в классе:**

- \*

- ->

**16) Какие из следующих операций можно перегрузить в классе:**

- %

- int()

- !=

**17) Какие из следующих операций можно перегрузить в классе:**

- ==

- char\*()

- ->

**18) Какой тип следует возвращать унарному оператору инкремента (++) класса Foo?**

Зависит от формы оператора

**19) Какой тип следует возвращать унарному префиксному оператору инкремента (++) класса Foo?**

Foo&

**20) Какой тип следует возвращать унарному постфиксному оператору инкремента (++) класса Foo?**

Foo

**21) Выберите правильный прототип оператора вывода в поток?**

std::ostream& operator<<(std::ostream& stream, const Foo& foo);

**22) Выберите операторы, которые нельзя перегружать в C++**

- .\*

- ::

**23) Выберите операторы, которые нельзя перегружать в C++**

- typeid

- .\*

- :?

**24) Выберите операторы, которые нельзя перегружать в C++**

- dynamic\_cast

- static\_cast

- ::

**25) Возможно ли в C++ определить в классе данный оператор?**

…

operator \*\*(int num)

{

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

this.\_num \*= this.\_num

}

…

Нет

**26) Чем отличается функция-член от метода?**

- Ничем

**27) Сколько видов доступа существует к методам класса (дайте ответ в виде числа)?**

3

**28) Какая операция используется при определении функции-члена для идентификации класса, которому принадлежит функция?**

::

**30) Сколько операндов имеет оператор индексирования? (дайте ответ в виде числа)**

1

**31) Какая последовательность вызова методов класса присуща данному участку кода:**

{

Car myCar = Car(“Vantage”);

Car \* sisterCar = new Car(“Matiz”);

cout << sisterCar->GetCarMode();

}

Car::Car(),Car::Car(),Car::GetCarMode(),Car::~Car()

**32) Какой результат выведет следующий код:**

class Employee

{

public:

Employee(int id, const std::string & name)

: Id(id), Name(name) {}

int Id;

std::string Name;

};

int main()

{

const int arraySize = 3;

Employee employees[arraySize] =

{

Employee(0x111, \"Petya\"),

Employee(0x222, \"Vasya\"),

Employee(0x333, \"Kolyla\")

};

Employee emp = employees[1];

emp.Id = 0x444;

emp.Name = \"Valera\";

std::cout <<

employees[1].Id << \" \" << employees[1].Name;

return 0;

}

546 Vasya

**33) Какой результат выведет следующий код:**

class Employee

{

public:

Employee(int id, const std::string & name)

: Id(id), Name(name) {}

int Id;

std::string Name;

};

int main()

{

const int arraySize = 3;

Employee employees[arraySize] =

{

Employee(0x111, \"Petya\"),

Employee(0x222, \"Vasya\"),

Employee(0x333, \"Kolyla\")

};

Employee & emp = employees[1];

emp.Id = 0x444;

emp.Name = \"Valera\";

std::cout <<

employees[1].Id << \" \" << employees[1].Name;

return 0;

}

1092 Valera

**34) Какие утверждения верны относительно перегрузки операторов?**

- должен быть минимум один операнд типа, определяемого пользователем

- нельзя нарушать правила синтаксиса исходной операции

- нельзя определять новые символы операций

**35) Какие утверждения не верны относительно перегрузки операторов?**

- нельзя добавлять const в конце прототипа перегруженного оператора

- все ссылки на операнды должны быть константными

**36) Какие операторы можно перегрузить только в виде функций-членов класса?**

- =

- ()

- []

- ->

**37) Какие операторы можно перегрузить не только в виде функций-членов класса?**

- <<

- >>

- +

**38) Какого ключевого слова не хватает для корректной компиляции и работы программы?**

class Employee

{

public:

Employee(int id, const std::string & name)

: Id(id), Name(name)

{

employeeCounter++;

}

int Id;

std::string Name;

int employeeCounter;

};

int Employee::employeeCounter = 0;

int main()

{

Employee \* first = new Employee(1, \"Vasay\");

Employee \* second = new Employee(2, \"Lesha\");

std::cout << Employee::employeeCounter;

return 0;

}

Static

**39) Какие операторы потребуются для освобождения памяти в следующем участке кода?**

class Employee

{

public:

Employee(int id, const std::string & name)

: Id(id), Name(name)

{

employeeCounter++;

}

int Id;

std::string Name;

static int employeeCounter;

};

int Employee::employeeCounter = 0;

int main()

{

const int arraySize = 5;

Employee \*\*employees = new Employee\*[arraySize];

for(int index = 0; index < arraySize; index++)

{

employees[index] = new Employee(index, \"Vasay\");

}

return 0;

}

- delete []

- delete

**40) Сколько операторов освобождения памяти потребуется в следующем участке кода? (дайте ответ в виде числа)**

class Employee

{

public:

Employee(int id, const std::string & name)

: Id(id), Name(name)

{

employeeCounter++;

}

int Id;

std::string Name;

static int employeeCounter;

};

int Employee::employeeCounter = 0;

int main()

{

const int arraySize = 5;

Employee \*\*employees = new Employee\*[arraySize];

for(int index = 0; index < arraySize; index++)

{

employees[index] = new Employee(index, \"Vasay\");

}

return 0;

}

6

**41) Какие функции-члены генерируются компилятором автоматически?**

- Оператор взятия адреса

- Оператор присваивания

- Конструктор по умолчанию

- Деструктор

**42) Какое ключевое слово необходимо для объявления статической функции?**

Static

**43) Укажите свойства статических функций-членов:**

- Может вызываться через имя класса

- Может использовать только статические члены данных

**44) Укажите свойства статических функций-членов:**

- Не обязательно вызывать через объект

- Не получает указатель this

**45) Укажите правильные способы вызова метода Method:**

class Klunk

{

public int Method();

};

void main()

{

Klunk klunk;

Klunk\* klunkptr = &klunk;

}

- klunk.Method()

- klunkptr->Method()

**46) Что означает эта строка в программе?**

Stock Stock::add(double operand){…}

Определение функции-члена класса Stock

**48) Как называется любая функция с определением внутри объявления класса:**

Встроенная

**49) С помощью какого квалификатора можно сделать встроенной функцию-член, определенную вне объявления класса?**

Inline

**50) Какие из вариантов правильно отображает использование метода show() объектом video?**

- video.show();

**54) Какой из предложенных вариантов ответов правильно отображает перегрузку** **оператора сложения:**

operator+( Myclass& object)

**56) Разрешено ли перегружать operator::()?**

Нет

## Структуры

**1) Корректен ли код представленный ниже:**

struct torgle\_register

{

unsigned int SN : 4;

unsigned int : 4;

bool goodIn : 1;

bool goodTorgle : 1;

};

Да

**2) Структура может содержать внутри себя данные …**

… разных типов

**3) Что НЕ может быть членом структуры?**

Сама эта структура

**4) Какой модификатор доступа по умолчанию в структуре?**

Public

**5) Можно ли реализовать нестандартный конструктор копирования для структуры?**

Можно

**6) Можно ли использовать структуру как элемент, возвращаемый функцией?**

Да, можно.

**7) Можно ли передавать функции в качестве параметра переменную типа структуры?**

Да, можно.

**8) Можно ли создавать массивы структур?**

Да, можно.

**9) Можно ли создавать структуры из массивов?**

Да, можно.

**11) Структура – это**

Универсальная форма данных, способная хранить элементы более чем одного типа

**12) Правильно ли использована процедура инициализации структуры и её экземпляра?**

struct duck { char name[20]; double volume; string price;};

duck first = {\"Daphe\", 12, 5};

Нет

**13) Может ли структура содержать член типа** string**?**

Да

**14) Может ли структура содержать член типа** std::runtime\_error**?**

Да

## Шаблонные классы

**1) Возможно ли указание значений по умолчанию для параметров шаблонного класса?**

Да

**2) В качестве чего может выступать шаблонный класс?**

- В качестве базового класса

- В качестве компонента других классов

## Семейство классов iostream

1) Являются ли операции ввода-вывода встроенными в язык C++

Нет

2) Сколько основных стадий включает в себя управление вводом и какие?

- Ассоциирование потока с вводом данных

- Подключение потока к файлу

3) Что такое буфер?

Блок памяти, используемый в качестве промежуточного временного хранилища при передаче информации от устройства в программу или из программы устройству

4) Что такое сабвуфер?

Акустическая система, воспроизводящая звуки низких частот

6) Дайте объяснение необходимости введения расширенных типов char

Чтобы сделать язык С++ интернациональным, позволяющим обрабатывать большее количество символов

7) Сколько потоковых объектов (cin, wcin, и т.д.) создаёт включение библиотеки iostream? (укажите только число)

8

8) Оба следующих оператора читают и отбрасывают символы, вплоть до конца строки, включая его. Будет ли форма ignore() порождать сбойные ситуации, если строка ввода превысит 80 символов?

Да

9) Выберете верные варианты конструкторов класса istrstream (файл <strstream>), при помощи которых создаются входные строковые потоки?

- "istrstream(const char \*\_Ptr, streamsize \_Count);

- istrstream(char \*\_Ptr, int \_Count);

10) Какой тип данных должна возвращать функция перегрузки вывода в поток?

ostream&

11) Какими манипуляторами можно установить систему счисления при выводе целочисленных значений?

- oct

- dec

- hex

12) Какой функцией можно изменить символ-заполнитель?

Fill

13) Какие функции предназначены для управления форматированием вывода?

- setf

- fmtflags

14) Как можно считать строку с пробелами из входного потока?

- В цикле с использованием функции get

- С использованием функции getline

15) Какая функция позволяет считать указанное количество байтов в бинарном виде?

Read

16) Какой режим открытия файла предусмотрен на добавление данных в конец файла?

App

17) Какой подход позволяет выделить память равную размеру файла?

Использовать функции seekg-tellg-seekg

18) Что будет, если при перегрузке оператора вывода в поток возвращать не ostream&, а ostream?

Программа не скомпилируется

19) Управление вводом включает в себя следующие стадии:

- Подключение потока к файлу

- Ассоциирование потока с вводом программы

20) Выберете все заголовочные файлы, связанные с потоками ввода – вывода:

- iostream

- cstdio

- fstream

21) (битое) Включение в программу файла iostream автоматически создает потоковые объекты. Потоку ошибок соответствуют объекты:

Выберите утверждения относящиеся к функции tmpnam():

Некоторая программа содержит следующий фрагмент. При использовании перенаправления вывода в файл на экране могут оказаться только строки:

cout << “Here come the goodies!\n”;

cerr << “Something horrible has happened.\n”;

exit(1);

Выберите верные утверждения:

- Объект cin представляет стандартный ввод в виде потока байтов

- Извлечение символов из потока подразумевает преобразование типа

-Пробельные символы при извлечении из потока с помощью cin пропускаются

22) Что будет выведено на экран следующей программой при вводе с клавиатуры последовательности «hi there»?

char first;

char second [10]{};

cin.get(first);

cin >> second;

cout << first <<\"+\"<< second;

h+i

23) Что будет выведено на экран после ввода с клавиатуры последовательности «I knew him, Goracio»?

char str[20]{};

cin.getline(str, 10, 'n');

cout << str;

I k

24) Выберите утверждения, описывающие метод read:

- Считывает заданное количество байтов

- Не добавляет нулевой символ в конце ввода

- Вызывается с двумя аргументами

25) Что будет выведено на экран следующей программой при вводе с клавиатуры последовательности «Hi there»?

[code]

cout << (char)cin.peek();

[\code]

**H**

26) Что будет выведено на экран следующей программой при вводе с клавиатуры последовательности «Beautiful»?

[code]

char temp[10]{};

cin >> temp;

char ch = cin.peek();

cin.putback(ch);

cout << cin.gcount();

[\code]

**0**

27) Буфер – это

Блок памяти, используемый в качестве промежуточного временного хранилища при передаче информации от устройства в программу или из программы устройству.

28) Помещать в буфер можно

Любой тип информации

29) Тип bitmask предназначен…

Для хранения значений индивидуальных битов

## Состояния потока

1) Какие функции позволяют отследить состояние потока?

- rdstate()

- exception()

2) Как установка битов состояния потока влияет на дальнейший ввод?

Поток закрывается для ввода, пока биты не будут очищены

3) Какие функция способна считывает символы пробела и новой строки, не прерывая ввод на них?

- get(char &)

- get(void)

- read()

- operator>>()

- getline()

4) Что возвратит функция get(void) при достижении конца файла?

Значение символьной константы EOF

6) В языке С++ операция “ << ” означает:

Побитовый сдвиг двоичного представления переменной

7) Какие из следующих выводов корректны?

- cout << “one” << “two” << “three” ;

- cout << “one\”<<<\”two”;

8) Какие из следующих выводов корректны?

- cout.put(42)

- cout.put(4).put(2)

- cout.put(‘W’)

- cout.write(\"42\", 2);

9) Аргументами метода write являются:

- Адрес строки, которую нужно отобразить

- Количество отображаемых символов

10) Выберете строки в которых произойдет произойдёт очистка буфера.

cout << “Enter a number: “;

int num;

cin >> num;

cout << “It is your number: “ << num << endl;

cout << flush;

- cout << “Enter a number: “;

- cout << “It is your number: “ << num << endl;

- cout << flush;

11) Укажите ключевое слово – манипулятор, изменяющее основание системы счисления для cout на шестнадцатиричную

Hex

12) Символы используемые для дополнения поля вывода, обеспечивая заданную ширину поля называются:

Символы – заполнители

13) Что будет выведено в результате работы следующей программы?

cout.width(5);

cout.fill('#');

cout<<12345;

cout<<12;

1234512

14) Выберете методы не применимые для cout:\*---

- cout.is\_number()

- cout.size()

## Полиморфизм

**1) Предположим, что определена функция, которая принимает в качестве аргумента ссылку на объект базового класса. Может ли эта функция также использовать в качестве аргумента объект производного класса?**

Да

2) Что можно сделать для корректной компиляции класса SuperCar:

class Car

{

public:

Car()

{

…

}

Car(const string & model)

{

…

}

virtual string GetModel() const

{

…

}

};

class SuperCar : public Car

{

};

можно ничего не делать

3) Что нужно сделать для корректной компиляции класса SuperCar:

class Car

{

public:

Car(const string & model)

{

…

}

virtual string GetModel() const

{

…

}

};

class SuperCar : public Car

{

public:

SuperCar(const string & model)

:Car(model)

{

…

}

};

можно ничего не делать

4) Что нужно сделать для корректной компиляции класса SuperCar:

class Car

{

public:

Car(const string & model)

{

…

}

virtual string GetModel() const = 0;

};

class SuperCar : public Car

{

public:

SuperCar(const string & model)

{

…

}

};

Car \* subaru = new SuperCar(\"Subaru Legacy\");

реализовать метод GetModel

5) Что нужно сделать для корректной компиляции кода:

class Car

{

public:

Car(const string & model)

{

…

}

virtual string GetModel() const

{

…

}

};

class SuperCar : public Car

{

SuperCar(const string & model)

{

…

}

};

Car \* subaru = new SuperCar(“Subaru Legacy”);

добавить перед конструктором класса SuperCar ключевое слово public

6) Можно ли присвоить объект производного класса объекту базового класса?

Да

7) Как объявить чисто виртуальную функцию call() в абстрактном классе?

virtual void call() const = 0;

8) Почему обычно лучше передавать объекты по ссылке, а не по значению?

- Передача объекта по ссылке позволяет использовать виртуальные функции

- Передача объекта по ссылке может расходовать меньше памяти

9) Какие функции-члены могут быть виртуальными?

- =

- &

- Оператор преобразования типа

- Деструктор

10) Пусть AAA – базовый класс, BBB – производный класс. В каждом из них определен виртуальный метод head(), pp – указатель на тип AAA, а переменной pp присвоен адрес объекта BBB. Метод какого класса будет вызван в строке pp->head()?

BBB

11) Пусть AAA – базовый класс, BBB – производный класс. В Каждом из них определен невиртуальный метод head(), pp – указатель на тип AAA, а переменной pp присвоен адрес объекта BBB. Метод какого класса будет вызван в строке pp->head()?

AAA

12) Какое ключевое слово используется при определении виртуального метода?

Virtual

13) Какие методы не могут быть виртуальными?

- Конструктор по умолчанию

- Конструктор с параметрами

- Конструктор копирования

14) К чему приводит использование виртуальных методов?

- Размер каждого объекта увеличивается

- Для каждого класса компилятор создает таблицу адресов виртуальных функций

- При каждом вызове функции выполняется дополнительный шаг поиска адреса в таблице

- К увеличению затрат памяти

- К снижению скорости выполнения

15) Как называется интерпретация вызова функции в исходном коде в виде выполнения определенной части кода?

Связывание

16) Если необходимо переопределить метод в производном классе, в базовом он указывается как виртуальный, на что это влияет?

Программа выбирает версию метода, основываясь на типе объекта

**17) Верно ли утверждение: «функции, определенные со ссылкой или указателем на базовый класс в качестве аргументов, могут использоваться с объектами как базового, так и производного класса»?**

Нет

18) Статическое связывание – это

Связывание, выполняемое во время компиляции, является ранним

**19) Динамическое связывание – это**

Связывание, выполняемое во время работы программы, является поздним

## Перегрузка

**1) В программе перегружены функции. Выберите строки кода, в которых возникает неоднозначность**

float fun1(float x)

{

return x;

}

double fun1(double x)

{

return x\*x;

}

double fun1(double &x)

{

return 2 \* x;

}

- double x = 2.4; cout << fun1(x) << endl;

- cout << fun1(5) << endl;

## Дружественные функции

**1) Имеет ли дружественна функцией класса доступ к его защищенным (protected) и собственным (private) компонентам?**

Да

**2) Как объявить дружественной функцию класса, если его объявление расположено ниже?**

Использовать объявление forward

**3) Выберете верные утверждения**

- Одна функция может быть дружественной нескольким классам

- Дружественная функция объявляется внутри класса, к элементам которого ей нужен доступ

4) Выберете правильное объявление дружественной функции?

friend Time operator\* (double m, const Time & t);

**5) В каком поле класса может располагаться дружественная функция?**

В любом поле класса – private, public или protected

6) Может ли дружественная функция быть глобальной функцией?

Да

7) Дружественная функция – это

Функция, которая не является членом класса и имеет доступ к членам класса объявленным в полях private и protected

8) В каком разделе объявляется дружественная функция?

В любом из них

9) Может ли одна функция быть дружественной сразу к нескольким классам?

Да

10) Передается ли указатель this дружественной функции?

Нет

11) Предположим, что прототип функции с ключевым словом friend объявлен сначала в классе А, а потом в классе B. Для какого класса эта функция будет являться дружественной?

Для обоих классов

**12) Каким образом происходит вызов дружественной функции?**

Явно

13) Может ли дружественный метод быть константным?

Да

14) Что обязательно должно передаваться в параметры дружественной функции?

Нет особых требований

15) Допустим требуется вызвать дружественную функцию GetName() для объекта Girl1. Какой вызов будет корректным?

GetName(Girl1)

16) К полям какой видимости имеет доступ дружественная функция?

- private

- protected

- public

17) Дружественная функция…

- …может модифицировать все элементы класса, включая скрытые поля

- …может являться компонентной функцией другого класса

18) Какие особенности характерны дружественным функциям?

- Производные классы не наследуют дружественные функции

- Дружественные функции не могут объявляться с ключевыми словами extern или static

- Дружественные функции могут быть дружественными для любого количества классов

19) Какие утверждения о вызове дружественной функции верны?

- Дружественные функции нельзя вызывать через объекты классов, друзьями которых они являются

- Дружественные функции нельзя вызывать через указатели на объекты классов

20) Выберите верные утверждения:

- Функция может использоваться, как дружественная к нескольким классам

- В виде параметра в дружественную функцию необходимо передавать указатель или ссылку на объект класса

- При объявлении дружественной функции необходимо явно передавать в нее элементы класса в виде параметров функции

**21) Что будет результатом выполнения данного кода?**

class Woman

{

private:

char\* \_name;

int \_weight;

friend void setData(char \*n, int w, Woman &obj)

{

strcpy(obj.\_name, n);

obj.\_weight = w;

}

friend void getData(Woman &obj)

{

cout << obj.\_name << \" : \" << obj.\_weight << \" kg. \" << endl;

}

public:

Woman()

{

\_name = new char[15];

strcpy(\_name, \"Name\");

\_weight = 0;

}

};

int main()

{

Woman Ann;

Ann.setData(\"Ann\", 55, Ann);

Ann.getData(Ann);

}

- Ошибка компиляции

## Генерация исключений

1) Каким образом можно сгенерировать исключение?

Используя ключевое слово throw

2) Как передать информацию при генерации исключения?

Объявить свой класс исключения

3) Какого типа может генерироваться исключение (объект какого типа может следовать за ключевым словом throw)?

Любого типа

4) Правильно ли дано определение try-блоку:

Try-блок это так называемый блок повторных попыток. В нем надо располагать код, который может привести к ошибке и аварийному закрытию программы

Да

5) Какие формы имеет оператор генерации исключения throw?

- throw;

- throw выражение\_генерации\_исключения;

6) Выберите правильные утверждения

- Если в блоке try формируется исключение, то делается попытка найти соответствующий исключению обработчик и передать ему управление

- Выбор правильного оператора catch определяется типом исключения

- Если при выполнении операторов блока try не возникло исключений, то управление передается в точку непосредственно после обработчиков исключения

- При выходе из функции по исключению гарантируется вызов деструкторов для уничтожения локальных объектов

7) Какие фрагменты программ не содержат ошибок? (неполное)

- double a = 2.5;try {throw a;cout << a << endl;}catch (double){ goto M1;}M1: cout << a++ << endl;

- try{ throw \"Exception\";}catch (char \* str){cout << str << endl; exit(1);}

**8) Что будет выведено на экран после выполнения следующего фрагмента программы?**

int a = 5, n = 10;

for (int i = 1; i <= n; ++i)

{

try

{

if (i % 4 != 0)

throw 1;

}

catch (int)

{

continue;

}

a += i;

}

cout << a << endl;

17

9) Что будет выведено на экран после выполнения следующего фрагмента программы?

int x = 10, s = 1;

while (x)

{

try{ if (x % 2) throw 5.f; }

catch (float) { break; }

x -= 1;

s \*= x;

}

cout << s << endl;

9

10) К какому типу относится оператор throw:

Перехода

11) Какой тип данных может иметь сгенерированное исключение:

- Любой

12) Напишите ключевое слово для оператора генерации исключений:

Throw

13) Какой (какими) командой (командами) генерируются исключения?

- throw

14) Что делает функция abort()?

- Прекращает выполнение программы

- Отправляет в стандартный поток вывода сообщение, которое зависит от реализации функции

- Возвращает вызывающему процессу код аварийного завершения программы

15) Преимущества подхода, при котором функции, которые генерируют исключения, создают объекты:

- Возможность применения разных типов исключений

- Объект содержит информацию, которая определяет причины, вызвавшие исключение

16) Какой класс в C++ является базовым для других классов исключений?

Exception

17) Какие виды ошибок относятся к семейству ошибок logic\_error:

- domain\_error

- invalid\_argument

- length\_error

- out\_of\_bounds

18) Какие виды ошибок относятся к семейству ошибок runtime\_error:

range\_error

19) Для чего предназначено семейство overflow\_error?

Для обозначения ошибок переполнения

20) Для чего предназначено семейство out\_of\_bounds?

Для обозначения ошибок индексации

21) Исключение какого типа генерируется при ошибках выделения памяти?

std::bad\_alloc

## Дружественные операторы

**1) Почему операторы ввода и вывода в поток не могут быть методами класса?**

Потому что объект класса должен быть вторым аргументом

**3) Как правильно объявить перегруженный оператор «!» дружественным классу Flag?**

friend Flag& operator ! (Flag& f)

**4) Перегрузке операторов потока свойственны следующие утверждения:**

- Возвращаемым значением для операций ввода и вывода должна быть ссылка на поток

- В операцию вывода необходимо передавать константную ссылку на объект класса

## Наследование классов

1) От какого количества классов можно наследовать новый класс в C++?

От неограниченного количества

2) Для наследования класса достаточно иметь?

Определение (declaration) базового класса

**3) Открытое (public) наследование отражает отношение:**

Является

**4) Как присвоить значение закрытой переменной базового класса из производного (с открытыми наследованием), если в базовом классе отсутствует соответствующий открытый метод установки значения?**

Никак

**5) Как объявить абстрактный класс?**

Добавить хотя бы один чисто виртуальный метод

**6) Чисто абстрактный класс:**

Содержит только чисто виртуальные методы

**7) Какой размер в байтах будет занимать объект класса в x86?**

class Foo

{

public:

int GetX()

{

return \_x;

}

void SetX(int x)

{

\_x = x;

}

virtual void Redouble()

{

\_x \*= 2;

}

private:

int \_x;

};

8

**8) Наследует ли производный класс методы конструктора, деструктор, операцию присваивания, дружественные функции? (да\нет)**

Нет

**11) Какие термины являются корректными для описания использования членов класса, которые сами представляют собой объекты другого класса?**

- Включение

- Композиция

- Иерархическое представление

**12) Наследование в приведенном участке кода называется:**

class SuperCar : public Car

{

…

};

- открытым порождением

**13) Что производный класс не наследует от базового?**

- Конструкторы

- Дружественные функции

- Деструктор

- Операцию присваивания

**14) И в базовом, и в производном классе определен метод с одним и тем же именем, и производный класс вызывает этот метод. Выберите верные утверждения:**

- Вызывается только метод производного класса

- Метод базового класса вызывается тогда, когда производный класс не переопределяет метод базового класса

- Метод базового класса вызывается тогда, когда используется операция разрешения контекста

**15) Что позволяет делать наследование?**

- Изменять поведение методов класса

- Добавлять новые возможности в существующий класс

**16) Какие существуют типы отношений между базовым и производным классом?**

- is-a

- has-a

- is-like-a

- uses-a

- is-implemented-as-a

17) Обращение к каким полям класса FruitsCost вызовет ошибку?

class FruitsCost {

friend void Set(char& fruit);

public:

int bananaCost;

protected:

int pineappleCost;

private:

int papayaCost;

}

void Set(char& fruit) {

fruit.bananaCost = 12;

fruit.pineappleCost = 23;

fruit.papayaCost = 42;

}

- pineappleCost

- papayaCost

18) Какой класс при данном наследовании является базовым?

class Parents : public Children {};

Children

19) Какой класс при данном наследовании является производным?

class Fruit : private Banana {};

Fruit

20) Как при открытом порождении производный класс может получить доступ к частным полям базового класса?

Через публичные и защищенные методы базового класса

21) Верно ли утверждение: «ссылкам и указателям производного класса разрешено присваивать объекты и адреса базового класса»?

Нет

22) Нисходящее приведение – это

Преобразование ссылки или указателя на производный класс в ссылку или указатель на базовый класс, которое не требует явного приведения типа

23) Восходящее приведение – это

Преобразование ссылки или указателя на базовый класс в ссылку или указатель на производный класс, требует явного приведения типа

**24) Сколько вариантов наследования существует в с++??**

3

25) Выберете верные свойства открытого наследования?

- Открытые члены становятся открытыми членами производного класса

- Защищенные члены становятся защищенными членами производного класса

- Закрытые члены становятся доступными только через интерфейс базового класса

- Неявное восходящее преобразование присутствует

26) Выберете верные свойства защищенного наследования?

- Открытые члены становятся защищенными членами производного класса

- Защищенные члены становятся защищенными членами производного класса

- Закрытые члены становятся доступными только через интерфейс базового класса

- Неявное восходящее преобразование присутствует, но только внутри производного класса

27) Выберете верные свойства закрытого наследования?

- Открытые члены становятся закрытыми членами защищенного класса

- Защищенные члены становятся закрытыми членами производного класса

- Закрытые члены становятся доступными только через интерфейс базового класса

- Неявное восходящее преобразование отсутствует

## Рекурсия

**1) Как называется функция, которая вызывает саму себя?**

Рекурсивной

**2) Выберите определение функции, использующее рекурсию, если функция вычисления суммы целых чисел от 1 до n имеет прототип int sum(int n):**

{if (n == 1) return 1; return n + sum(n - 1);}

**3) Функция вычисления суммы целых чисел от start до end имеет прототип int sum(int start, int end); Как записать определение функции, используя рекурсию?**

{if (end == start) return start; return end + sum(start, end -1);}

**4) Выберите запись строки (???) для правильной реализации рекурсивной функции вычисления факториала числа n:**

int fact(int n)

{

(???)

return n \* fact(n - 1);

}

if (n == 0) return 1;

**5) Вызов из некоторой функции этой же самой функции - это:**

Прямая рекурсия

**6) Вызов из первой функции второй функции, когда вторая функция в своём вычислении использует первую функцию – это:**

Косвенная рекурсия

**7) Выберите верные утверждения о рекурсивной функции:**

- Любую рекурсию теоретически можно заменить обычными циклами.

- Основное достоинство рекурсии в том, что с помощью конечной рекурсивной программы можно описать бесконечное вычисление, причём программа не будет содержать явных повторений.

- Рекурсии требуют больше времени и памяти, чем требуют обычные циклы.

**8) Дана рекурсивная функция:**

int function(int n)

{

if (n == 0)

return 1;

return n\* function (n - 1);

}

**9) Верно ли написана эта же функция, построенная уже без использования рекурсий:**

int function (int n)

{

if(n == 0)

return 1;

int temp = 1;

for(int i = 1; i <= n; i++)

{

temp = temp\*i;

}

return temp;

}

Да

**11) Глубина рекурсии ограничена:**

Размером стека вызовов

**13) Выберите правильный ответ:**

Функция называется косвенной рекурсией, если она вызывает другую функцию.

**14) На каких строках происходит вызов рекурсивной функции?**

#include <iostream>

void countdown(int n);

int main()

{

countdown(4);

return 0;

}

void countdown(int n)

{

using namespace std;

cout << “Couting down …” << n << endl;

if(n>0)

сountdown(n-1);

cout << n << “: Kaboom!\n”;

}

- 5

- 13

## Общие понятия

**1) Что означает буква “S” в аббревиатуре SOLID?**

Принцип единственной ответственности

**2) Что означает буква “O” в аббревиатуре SOLID?**

Принцип открытости/закрытости

**3) Что означает буква “L” в аббревиатуре SOLID?**

Принцип подстановки Барбары Лисков

**4) Что означает буква “I” в аббревиатуре SOLID?**

Принцип разделения интерфейса

**5) Что означает буква “D” в аббревиатуре SOLID?**

Принцип инверсии зависимостей

**6) Что такое наследование?**

Это механизм, посредством которого производный класс получает элементы родительского и может дополнять либо изменять их свойства и методы

**7) Что позволяет делать наследование?**

- Добавлять новые возможности в следующий класс. Например, в существующий базовый класс массива можно добавить арифметические операции

- Добавлять данные, которые представляет класс. Например, взяв за основу базовый класс строки, можно породить класс, в котором добавлено поле, представляющее цвет, и которое будет использоваться при выводе строки на экран

- Изменять поведение методов класса. Например, от класса Passenger, который представляет услуги, предоставляемые пассажиру авиалинии, можно породить класс FirstClassPassengerс более высоким уровнем обслуживания

**8) Как называется способ, позволяющий порождать новые классы от старых?**

- Наследование

**9) Какими словами можно назвать прием использования объекта класса в качестве поля другого класса?**

- Включение

- Композиция

- Иерархическое представление

10) Какие типы данных предоставляют шаблоны?

Параметризованные

**11) Как называется актуализация шаблона?**

Специализация

## Порядок создания и уничтожения объектов

**1) Когда вызывается конструктор базового класса?**

До вызова конструктора производного

2) Когда вызывается деструктор базового класса?

После выполнения деструктора производного

**3) Какая ошибка произойдет в данной программе?**

class Base

{

public:

Base(int x) : \_x(x) {}

virtual int GetX()

{

return \_x;

}

private:

int \_x;

};

class Derived : public Base

{

public:

Derived(int x) : Base(x)

{

\_arr = new int[GetX()];

}

virtual int GetX()

{

if (GetX() > 10)

return GetX();

return \_arr[GetX() / 2];

}

private:

int\* \_arr;

};

int main()

{

Base\* obj1 = new Base(42);

Base\* obj2 = new Derived(42);

...

delete obj1;

delete obj2;

return 0;

}

Утечка памяти

4) При создании и удалении объекта производного класса конструкторы вызываются в порядке порождения, и первым вызывается наиболее вложенный в плане наследования конструктор, а деструкторы вызываются в обратном порядке (да\нет)

Да

**5) Если производный класс не добавляет члены данных в базовый класс, то нужны ли конструкторы для производного класса?**

Да

6) В каком порядке вызываются конструкторы и деструкторы класса при создании и удалении объекта производного класса?

Конструкторы - в порядке порождения, деструкторы в обратном порядке

**7) Наследует ли производный класс конструкторы базового класса?**

Нет

**8) Выберите правильные утверждения:**

- Конструкторы не могут быть виртуальными

- Деструкторы следует реализовывать виртуальными

**9) Какие основные моменты характерны для конструкторов производных классов**

- Сначала создается объект базового класса

- Конструктор производного класса должен передавать информацию базового класса конструктору базового класса через список инициализаторов членов

- Конструктор производного класса должен инициализировать данные-члены, добавленные в производном классе

**11) Верно ли утверждение: «если у класса есть непрямой виртуальный базовый класс, конструктор этого класса всегда должен явно вызывать конструктор виртуального базового класса»?**

Нет

## Дружественные классы

**1) Можно ли в C++ объявлять класс в другом классе?**

Да, но только с другим именем

**2) Как получить доступ ко вложенному классу, если он объявлен в public-области?**

С помощью операции разрешения контекста

3) Как получить доступ ко вложенному классу, если он объявлен в private-области?

Только через public-методы внешнего класса

4) Как нужно объявлять вложенный шаблонный класс в другом шаблонном классе?

Просто использовать шаблонный тип внешнего класса

6) Предоставляет ли особые права доступа вложение класса в другой класс к функциям-членам вложенного класса?

Нет

7) Соответствует ли данный фрагмент кода верному объявлению дружественного класса?

#include <iostream>

using namespace std;

class square;

class figure

{

public:

figure();

void getvalue();

friend square;

private:

int a;

}

void figure::getvalue()

{

cout << a;

}

class squared

{

public:

void getAreal();

}

void squared::getAreal(figure &obj)

{

cout << “Объем = ” << obj.a;

}

Да

8) Возможно ли создать более ограниченную форму отношения дружественности, чем взаимно дружественные классы, при которой только некоторые члены класса B являются друзьями для класса A и некоторые члены класса A – друзьями для B?

Нет

9) Допустим есть класс А, дружественный ему класс B, а у класса A есть наследники C, D, E. Будут ли наследники класса А иметь доступ к закрытым членам класса B?

Нет

10) Что неверно в следующей попытке создать дружественную конструкцию?

class Student

{

friend Teacher;

...

};

class Teacher {...};

- Неверно используется объявление дружественного класса, необходимо использовать friend class Teacher

11) Что может быть дружественным?

- Класс

- Функция

- Оператор

- Метод

12) Выберите ложные суждения. Если класс А – друг класса B, а класс B – друг класса C, то…

- …из этого следует, что класс A – друг класса С, а С – друг класса А

- …из этого следует, что класс А – друг класса С, класс В – друг класса А, класс С – друг А

- …из этого не следует, что В – друг А, С – друг А, но следует, что А – друг С

13) К чему может привести чрезмерное злоупотребление отношения дружественности?

- К нарушению инкапсуляции классов

- К снижению степени защиты разрабатываемого приложения

14) Выберите неверные утверждения:

class Y

{

//..

friend class X;

}

- Класс Х наследуется от класса Y

- Класс Y наследуется от класса X

- Такая запись вызовет ошибку

**15) Выберите верные утверждения:**

- Дружественная перегрузка позволяет перегружать операторы

## Друж. классы. Общие понятия

**1) Дружественным для класса может быть?**

- Другой класс

- Метод другого класса

- Функция

**2) Какие существуют разновидности друзей?**

- Дружественные функции

- Дружественные классы

- Дружественные функции-члены

**3) Может ли локальный класс использовать из объемлющей его области только имена типов, статические переменные, внешние (extern) переменные, внешние функции и элементы перечислений?**

Да

**4) Обладает ли дружественность свойствами симметричности и транзитивности?**

Нет

## Общие понятия об исключениях

1) Каким ключевым словом объявляется блок, в котором отслеживаются исключения?

Try

**2) Функция А в блоке try вызывает функцию B, функция B создает некоторый объект foo и вызывает С, а С генерирует исключение. Будет ли вызван деструктор объекта foo?**

Да

**3) Для чего применяются исключения в с++?**

- Для того, чтобы выполнить действие, при наступлении нестандартной ситуации

- Для того, чтобы обеспечить продолжение работы программы или, по крайней мере, ограничить последствия возникновения исключительных ситуаций

**4) На каких ключевых словах основан механизм обработки исключительных ситуаций в С++?**

- catch

- try

- throw

5) К каким событиям неприменим механизм обработки исключений?

- Ошибка оборудования

- Аварийное отключение питания

- Прерывание программы (нажатие клавиш Ctrl+C)

6) Исключения и ошибки это одно и тоже?

Нет

7) Когда возникают синхронные исключения?

В определенных, заранее известных точках программы

**8) Когда возникают асинхронные исключения?**

В любой момент времени, при этом они не зависят от того, какую инструкцию выполняет программа

**9) Когда возникают отложенные исключения?**

Таких исключений нет

**10) Существуют ли виды обработки исключений к программе?**

Да, структурная и неструктурная

**11) Для управления исключениями существуют следующие компоненты:**

- Генерация исключений

- Перехват исключений обработчиком

## Проталкивание исключений

1) Что означает спецификация noexcept

Функция не генерирует исключений

**2) Функция А в блоке try вызывает функцию B, функция B вызывает С, а С генерирует исключение. Что произойдет?**

Управление будет передано в соответствующий блок catch функции А

**3) К какой функции обращается программа в случае возникновения неперехваченного исключения:**

terminate()

## Перехват исключений

1) Сколько блоков catch может присутствовать в одной конструкции try...catch?

Неограниченное количество

2) Важен ли порядок объявления блоков catch?

Не всегда, зависит от типов перехватываемых исключений

3) Как перехватить исключения любого типа?

- Объявлением блока catch(…)

5) Дан код программы:

#include <iostream>

#include <cmath>

struct DataException : std::exception

{

DataException(float d1, float d2, char \*str)

: a(d1), b(d2), msg(str){ }

float a, b;

char \*msg;

};

float MRoot(float x, float y)

{

double z = x \* y;

if (z > 1e+70)

throw DataException(x, y, \"Переполнение!\");

if (x < 0 && y > 0)

throw DataException(x, y, \"Отрицателен 1-й параметр\");

if (x > 0 && y < 0)

throw DataException(x, y, \"Отрицателен 2-й параметр\");

return (float)sqrt(z);

}

void DataExceptionTesting()

{

setlocale(LC\_CTYPE, \"Russian\");

try

{

std::cout <<\"MRoot(x,t) = \"<<MRoot(16., 9.) << std::endl; //1

std::cout <<\"MRoot(x,t) = \"<<MRoot(1.6e38,9e36)<<std::endl;//2

std::cout <<\"MRoot(x,t) = \"<<MRoot(16., -9.) << std::endl; //3

}

catch (DataException e)

{

std::cout << e.msg<<\" x = \"<< e.a <<\" y = \"<< e.b <<std::endl;

}

}

**Какие строки будут присутствовать в результате выполнения функции DataExceptionTesting()**

- MRoot(x,t) = 12

- Переполнение! x = 1.6e+038 y = 9e+036

**6) Дан код программы:**

#include <iostream>

#include <cmath>

class OverFlow

{ };

float MRoot(float x, float y)

{

double z = x\*y;

if (z > 1e+70)

throw OverFlow();

if (x < 0 && y > 0)

throw \"Отрицателен 1-й параметр\";

if (x > 0 && y < 0)

throw \"Отрицателен 2-й параметр\";

return (float)sqrt(z);

}

void MRootTesting()

{

setlocale(LC\_CTYPE, \"Russian\");

try

{

std::cout <<\"MRoot(x,t) = \"<<MRoot(16., 9.); //1

std::cout <<\"MRoot(x,t) = \"<<MRoot(1.6e38,9e36);//2

std::cout <<\"MRoot(x,t) = \"<<MRoot(16., -9.); //3

std::cout <<\"MRoot(x,t) = \"<<MRoot(-16., 9.); //4

}

catch(OverFlow)

{

std::cout << \"Переполнение\" << std::endl;

}

catch(const char \* report)

{

std::cout << report << std::endl;

}

}

**Каков будет результат выполнения функции MRootTesting():**

- MRoot(x,t) = 12

Переполнение

7) При обработке исключений можно ли разделить блоки try и catch каким-нибудь промежуточным кодом?

Нет

**8) Должен ли соответствовать тип исключения throw блока типу принимаемого аргумента catch-блока?**

Да

**9) Определена функция деления двух целых чисел, что произойдет при вызове division(5, 0):**

int division(int n1, int n2)

{

if (n2 == 0)

{

throw 99;

}

return n1 / n2;

try

{

cout << division(5, 0);

cout << endl;

}

catch(int i)

{

cout << \"Ошибка №\" << i << \" Деление на ноль.\" << endl;

}

}

Деление на ноль не состоится и catch поймает число 99

**10) Предположим, что имеется иерархия классов исключений, порожденная от базового класса исключений. В каком порядке следует расположить блоки catch?**

Блоки catch необходимо упорядочить от наиболее глубокого в цепочке наследования до наименее глубоко

**11) Что будет выведено на экран после выполнения фрагмента программы**

int k = 1;

try

{

if (k == 0) throw new int(5);

if (k == 1) throw 5.;

if (k == 2) throw '5';

if (k == 3) throw 5;

}

catch (int\*)

{ cout << \"catch int\*\" << endl;}

catch (int)

{ cout << \"catch int\" << endl; }

catch (float)

{ cout << \"catch float\" << endl;}

catch (...)

{ cout << \"catch \" << endl; }

- catch

- catch int\*

- catch float

12) Каким будет результат выполнения программы?

#include <iostream>

using namespace std;

class Base { };

class Derived : public Base {};

int main() {

Derived test;

try{ throw test;}

catch (Base){ cout << \" Base \n\";}

catch (Derived) {cout << \" Derived \n\";}

return 0;

}

- Base

13) Каким будет результат выполнения программы?

#include <iostream>

using namespace std;

class Base { };

class Derived : public Base {};

int main() {

Derived test;

try{ throw test;}

catch (Derived){ cout << \"Derived\n\";}

catch (Base) {cout << \"Base\n\";}

return 0;

}

- Derived

14) Какие формы может иметь спецификация исключения, размещенная после служебного слова catch в круглых скобках?

- catch(тип имя) {//тело обработчика}

- catch(тип) {//тело обработчика}

- catch(…){//тело обработчика}

17) Оператор catch (…) выполняет:

Перехват исключения любого типа

18) Как следует располагать блоки catch при наследовании классов исключений:

В порядке, обратном порождению

**19) Если имеется иерархия исключений, то блоки catch необходимо располагать в следующем порядке:**

Сначала производные классы, затем базовые

**20) Что будет выведено на экран в результате выполнения кода?**

#include <iostream>

int hmean (int a, int b)

{

if (a == -b)

throw “bad hmean”;

return 2 \* a \* b / (a + b);

}

int main()

{

try {

std::cout << hmean(4, 3);

}

catch (const char \* s) {

std::cout << s << std::endl;

}

catch (…) {

std::cout << “Bad arguments!”;

}

return 0;

}

3

21) Что будет выведено на экран в результате выполнения кода?

#include <iostream>

#include <exception>

int hmean (int a, int b)

{

if (a == -b)

throw new std::exception();

return 2 \* a \* b / (a + b);

}

int main()

{

try {

std::cout << hmean(3, 3);

}

catch (const char \* s) {

std::cout << s << std::endl;

}

catch (…) {

std::cout << “Bad arguments!”;

}

return 0;

}

3

22) Что будет выведено на экран в результате выполнения кода?

#include <iostream>

#include <exception>

int hmean (int a, int b)

{

if (a == -b)

throw new std::exception();

return 2 \* a \* b / (a + b);

}

int main()

{

try {

std::cout << hmean(3, -3);

}

catch (const char \* s) {

std::cout << s << std::endl;

}

catch (…) {

std::cout << “Bad arguments!”;

}

return 0;

}

Bad arguments!

23) Что будет выведено на экран в результате выполнения кода?

#include <iostream>

int hmean (int a, int b)

{

if (a == -b)

throw “bad hmean!”;

return 2 \* a \* b / (a + b);

}

int main()

{

try {

std::cout << hmean(3, -3);

}

catch (const char \* s) {

std::cout << s << std::endl;

}

catch (…) {

std::cout << “Bad arguments!”;

}

return 0;

}

bad hmean!

## Шаблонные функции

**1) В каких вариантах переопределения доступа для применения методов из класса valarray с производным классом с помощью using допущена ошибка?**

- using std::valarray::max();

- using std::valarray::operator[]();