**Тест по C++**

Каждый вопрос имеет только один вариант ответа. Тесты не затрагивают стандарт С++11 и более поздние.

**1**

p = new car;

Какое значение будет иметь переменная p после выполнения этого кода?

1. Значение членов-данных объекта класса car
2. Указатель на динамически расположенный в памяти объект p
3. Указатель на динамически выделенную память под объект car
4. Указатель на динамически выделенную память под p
5. Неинициализированное значение NULL-указателя на new car

**2**

cout « "The Size of \"ulysses\".\n";

Каким будет вывод в этом примере?

1. The Size of ulysses.
2. The Size of "ulysses".\n
3. The Size of \"ulysses".
4. The Size of "ulysses".
5. The Size of \"ulysses"\.

**3**

CLASS\* pointer = new CLASS

Какую команду следует использовать для освобождения памяти, выделенной приведенной выше командой?

1. delete pointer
2. delete [] pointer ;
3. delete [1] pointer ;
4. delete pointer [] ;
5. delete pointer [1] ;

**4**

int vector[5];

int\* pointer = vector;

Почему инициализация указателя в приведенном коде является неверной?

1. Потому что она приводит к тому, что код становится зависимым от положения символов
2. Потому что статические переменные-указатели можно не инициализировать
3. Потому что vector — это массив, а не указатель на int
4. Потому что vector — это указатель на указатель на int
5. Она не является неверной

**5**

char\* pointer = new char[20];

Какую команду следует использовать для освобождения памяти, выделенной приведенной выше командой?

1. delete pointer ;
2. delete [] pointer ;
3. delete [20] pointer ;
4. delete pointer [] ;
5. delete pointer [20] ;

**6**

int acnt = 0;

switch(ch) {case 1: ++acnt; case 4: ++acnt; case 11: ++acnt;};

Пусть значение ch равно 1, чему будет равно значение acnt после выполнения приведенного выше блока кода?

1. 0
2. 1
3. 2
4. 3
5. 4

**7**

class Word;

bool operator!(Word st1, Word st2);

В чем заключается ошибка определения оператора ! в приведенном коде?

1. ! нельзя перегружать
2. параметры должны быть ссылками
3. слишком много параметров
4. неправильный тип возвращаемого значения
5. параметры типа класса в перегружаемом операторе должны быть указателями

**8**

Какое из приведенных утверждений о проектировании класса ЛОЖНО?

1. Если в классе нет оператора присваивания, компилятор создаст неявный оператор присваивания
2. Если в классе нет конструктора, компилятор создаст неявный конструктор по умолчанию
3. Если в классе нет конструктора копирования, компилятор создаст неявный конструктор копирования по умолчанию
4. Конструктор по умолчанию не может иметь параметров
5. С помощью оператора new нельзя создавать массивы объектов класса без конструктора по умолчанию

**9**

int SomeFunction (int input){

if(input > 0)

if(input > 100)

return 100;

else

return input;

return 0;}

Каким будет значение выражения SomeFunction(-1) после выполнения приведенного кода?

1. 0
2. 1
3. 100
4. -1
5. -100

**10**

int matrix[5, 4];

Что из нижеприведенного точнее описывает приведенное объявление?

1. Это одномерный массив целых чисел с индексами от 0 до 3
2. Это одномерный массив целых чисел с индексами от 1 до 4
3. Это двумерный массив целых чисел с индексами от 1 до 5 и от 1 до 4
4. Это двумерный массив целых чисел с индексами от 0 до 4 и от 0 до 3
5. Оно некорректно

**11**

class Foo{

public: Foo(); Foo(const Foo& f); Foo(double f);

private: Foo& operator=(const Foo& f);}

С учетом приведенного кода, какой из приведенных ниже фрагментов кода, при условии того, что он написан в теле функции, не являющейся функцией-членом, НЕ создает объект класса Foo?

1. Foo\* object = new Foo;
2. static volatile Foo f;
3. Foo f = 3;
4. extern Foo f;
5. const Foo& f = Foo();

**12**

int main()

{

samp\* p;

p = new samp[3] = {samp(1), samp(2), samp(3)};

return 0;

}

В чем заключается ошибка в приведенном коде?

1. Ошибки нет
2. Нельзя присваивать переменно р динамический массив
3. Динамические массивы можно инициализировать только с помощью конструктора по умолчанию
4. Нельзя приравнивать динамический массив указателю
5. Нельзя использовать new для создания массива объектов

**13**

Пусть pointer — указатель на функцию, являющуюся членом класса SomeClass, возвращающую void и имеющую один параметр типа int, someobject — указатель на объект класса SomeClass. Какой фрагмент кода позволяет корректно вызвать функцию-член, на которую указывает pointer?

1. pointer (1)
2. SomeClass::pointer (29)
3. someobject->pointer (10)
4. someobject->\*pointer (4)
5. (someobject->\*pointer) (2)

**14**

Что произойдет, если функция имеет спецификацию исключений и при ее вызове явно или неявно будет сгенерировано исключение, которого нет в спецификации?

1. Ничего не произойдет
2. Исключение приведет к вызову std::unexpected()
3. Будет сгенерировано std::cancel
4. Компилятор выдаст ошибку
5. Библиотека вызовет exit(1)

**15**

struct X { virtual ~X() {} };

class Y : public X {};

int main {

X\* p = new Y[2];

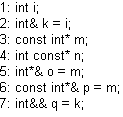
delete[] p;

return 0;}

В чем заключается ошибка в приведенном коде, если она есть?

1. Класс не может наследоваться от структуры
2. Код вызывает неопределенное поведение
3. Для main нужны аргументы
4. Для Y нужен деструктор
5. Все правильно

**16**



Какие строки вызовут ошибку компиляции?

1. 3 и 4
2. 4 и 5
3. 7
4. 5 и 6
5. 5 и 7

**17**

Какое утверждение верно по отношению к объявлению класса в С++?

1. Он должен иметь по крайней мере одно поле данных
2. Он должен иметь по крайней мере один конструктор
3. Он не требует функций-членов
4. Он должен иметь по крайней мере один деструктор
5. Объявление класса должно быть в файле .h или .hpp

**18**

class X {

public:

X() {cout<<"X constructor was called"<<endl;}

X(int n) {cout<<"X(int) constructor was called"<<endl;}

~X(){cout<<"X destructor was called"<<endl;}

};

class A {

X x;

public:

A() {cout<<"A constructor was called"<<endl;}

A(int n) {cout<<"A(int) constructor was called"<<endl;}

~A(){cout<<"A destructor was called"<<endl;}

};

int main() { A a1(1);}

Каким будет вывод приведенного кода?

1) X constructor was called

A(int) constructor was called

A destructor was called

X destructor was called

2) A constructor was called

X(int) constructor was called

X destructor was called

A destructor was called

3) X destructor was called

A constructor was called

A destructor was called

4) A constructor was called

A(int) constructor was called

5) X(int) constructor was called

X(int) destructor was called

A constructor was called

A destructor was called

**19**

Когда вызывается деструктор?

1. Когда класс удаляется
2. Когда объект выходить из области определения или удаляется
3. Когда вызывается соответствующий конструктор
4. Когда дружественная функция завершает обработку
5. Когда удаляется ссылка на объект

**20**

class HasStatic {static int I;};

Как правильно определить переменную-член I и присвоить ей значение 10 вне объявления класса?

1. static I = 10;
2. static I(10);
3. HasStatic I = 10;
4. int HasStatic::I = 10;
5. int static I = 10;

**21**

Какое из приведенных утверждений о константной переменной верно?

1. Адресу константной переменной можно присвоить только переменную типа float
2. Указателю на неконстанту нельзя присвоить адрес константной переменной
3. Указателю на константу можно присваивать различные объекты на этапе компиляции
4. Указатели на константные объекты должны быть проинициализированы значением NULL перед обращением к ним
5. Указателю на неконстанту нельзя присвоить адрес volatile переменной

**22**

char a[] = "ABCD\r\n";

Сколько элементов будет в массиве, созданном приведенной командой?

1. 6
2. 7
3. 8
4. 9
5. 10

**23**

class A {int x;};

class B {protected: int y;};

class C {public: int z;};

class D : public A, B, protected C {};

Каким будет уровень доступа у переменных-членов класса D, унаследованных от базовых классов A, B, C?

1. x недоступен; y - private; z - public
2. x - private; y - protected; z - protected
3. x - private; y - private; z - protected
4. x - private; y - protected; z - public
5. x вне видимости; y - private; z - protected

**24**

class animal {};

class pet {};

class dog : public pet {};

class petdogs : public dog, public animal {};

petdogs lassie;

В каком порядке будут вызваны конструкторы при создании lassie?

1. dog(), pet(), animal(), petdogs()
2. animal(), dog(), pet(), petdogs()
3. petdogs(), dog(), pet(), animal()
4. pet(), dog(), animal(), petdogs()
5. animal(), pet(), dog(), petdogs()

**25**

class X{

public: X(int a, int b, int c) : a(a), b(b), c(c) {}

private:

int b;

int a;

int c;};

Укажите правильный порядок инициализации членов класса.

1. a, b, c.
2. c, b, a.
3. b, a, c.
4. c, a, b.
5. Код некорректен, так как члены класса создаются с помощью копирования собственных неинициализированных значений