Quizizz

object oriented programming quiz

23 Questions

NAME :

CLASS :

DATE :

1.

```
class Employee {
    enum class Seniority : char {
        Junior = 'J', Middle = 'M', Senior = 'S'
    };

public:

    Employee(Seniority seniority = Seniority::Junior, const std::string &name = "")
            : m_seniority{seniority}, m_name{name} {
    }

    Employee(const std::string &name) : Employee{ seniority: Seniority::Junior, name} {}

    operator std::string() {
        return m_name + " is a " + static_cast<char>(m_seniority);
    }

private:
    Seniority m_seniority{};
    std::string m_name{};
};
```

Определение класса *Employee предоставлено на картинке. Что выведет следующий код:*
Employee employee("Ivan");
std::cout << std::string(employee);

A Ivan is a S

B Код не скомпилируется

C Ivan is a J

D is a

E Ivan is a Junior

2. Ниже приведены рекомендации относительно размещения кода в .h и .cpp файлах. Выберите правильные.

A. классы, используемые только в одном файле, и которые, как правило, нельзя использовать повторно, определяйте непосредственно в том файле .cpp, где они используются

B. классы, используемые в нескольких файлах или предназначенные для повторного использования, объявляйте в файле .h, который имеет такое же имя, что и класс

C. тривиальные функции-члены (тривиальные конструкторы или деструкторы, функции доступа и т.д.) могут быть определены внутри класса

D. нетривиальные функции-члены должны быть определены в файле .cpp, имеющем такое же имя как и класс

3.

```
class IdManager {
    const int m_id;

public:
    IdManager(int id);
};
```

Выберите правильную реализацию конструктора инициализирующую поле *m_id*

A Конструктор невозможно написать так, чтобы проинициализировать m_id.

B
```
IdManager::IdManager(int id) const {
    m_id = id;
}
```

C
```
IdManager::IdManager(int id)
    : m_id{id} {
}
```

D
```
IdManager::IdManager(int id) {
    m_id = id;
}
```

4.

```
class Base {
private:
    int my_private_variable;
protected:
    int my_protected_variable;
public:
    int my_public_variable;
};

class Derived : <key word> Base {

};
```

Какое ключевое слово/слова можно вставить вместо <key word>, чтобы получить доступ к
my_private_variable
в классе Derived

A protected

B public

C можно ничего не вставлять и будет доступ

D private

E какое бы ключевое слово не вставили доступа не будет

5.
```
class Base {
private:
    int my_private_variable;
protected:
    int my_protected_variable;
public:
    int my_public_variable;
};

class Derived : protected Base {
};
```

К какому полю/полям класса Base мы имеем доступ в классе Derived

A my_private_variable

B my_protected_variable

C my_public_variable

D не имеем доступа ни к каким переменным

6.
```
class A {
public:
    A() {std::cout << "A";}
    ~A() {std::cout << "~A";}
};

class B: public A {
public:
    B() {std::cout << "B";}
    ~B() {std::cout << "~B";}
};

class C : public B {
public:
    C(){std::cout << "C";}
    ~C(){std::cout << "~C";}
};
```
Что выведется в консоль?
int main() { C(); return 0; }

7.
```
class Base {
public:
    Base();
};

class Derived : public Base {
public:
    Derived();
};
```

Допустим у нас есть класс базовый класс Base. Выберите правильный порядок выполнения операций компилятором при вызове конструктора его наследника Derived.
1. выполняется тело конструктора
2. список инициализации инициализирует переменные
3. управление возвращается вызывающей функции
4. выделяется память для Derived (достаточная и для части Base, и для части Derived)
5. вызывается соответствующий конструктор Derived
6. создается объект Base с использованием соответствующего конструктора Base

A 546213

B 312546

C 542613

D 456213

8.

```
class Base {
protected:
    int m_num;
public:
    Base() : m_num{2} {}
    int GetNum() {
        return m_num;
    }
};

class Derived : public Base {
public:
    Derived(int num) : m_num{num} {
    }
};
```

Что выведет программа?
Derived d(3);
std::cout << d.GetNum();

A 3

B 0

C программа не скомпилируется

D 2

9.

```
class Base {
private:
    int m_num;
public:
    Base(int num) : m_num{num} {}
};

class Derived : public Base {
public:
    Derived(<1>) : <2> {
        <3>
    }
};
```

В каком из мест нужно вызвать конструктор класса Base, чтобы программа заработала?

A ни в каком, т.к. m_num является private членом

B 2

C 3

D в любом

E 1

10.

```
class A {
public:
    int a;
protected:
    int a1;
private:
    int a2;
};

class B : private A {
public:
    int b;
protected:
    int b1;
private:
    int b2;
};

class C : protected B {

};
```

Какие члены родительских классов доступны в классе C

A a, b, b1

B a, a1, b, b1

C b, b1

D a, a1, b, b1, b2

E b, b1, b2

11.

```
class SpyBase {
protected:
    void Identify() {std::cout << "The sky is blue!\n";}
};

class Spy : public SpyBase {
public:
    void Identify() {
        // <?>
        // some additional actions
    }
};
```

Что нужно написать вместо //<?>, чтобы вызвать метод Identify базового класса SpyBase

A SpyBase::Identify();

B this->Identify();

C SpyBase().Identify();

D Identify();

12.

```
class Base {
public:
    void PrintName() { std::cout << "Base"; }
};

class Derived : public Base {
public:
    void PrintName() { std::cout << "Derived"; }
};
```

Что выведется в консоль, если в main() написать
Base * obj = new Derived;
obj->PrintName();

A Derived

B Base

C Программа не скомпилируется

D BaseDerived

13.

```
class Animal {
public:
    virtual void Speak() { std::cout << "???"; }
};

class Cat : public Animal {
public:
    void Speak() override { std::cout << "Meow"; }
};

int main() {
    Cat cat;
    Animal & animal = cat;
    animal.Speak();
    return 0;
}
```

Что выведется на экран после выполнения программы?

A Ничего не выведется, но программа завершится успешно

B Программа не скомпилируется, т.к. вызов метода Speak() неоднозначен

C ???

D Meow

14.

```
class Animal {
public:
    Animal() {
        Report();
    }
    virtual void Report() {std::cout << Name() << " says " << Speak(); }
    virtual std::string Name() { return "Animal"; }
    virtual std::string Speak() { return "???"; }
};

class Cat : public Animal {
public:
    Cat() {}
    virtual void Report() {std::cout << Name() << " says " << Speak(); }
    virtual std::string Name() { return "Cat"; }
    virtual std::string Speak() { return "Meow"; }
};

int main() {
    Cat cat;
    Animal & animal = cat;
    return 0;
}
```

Что выведется в консоль после выполнения данной программы?

A Animal says Meow

B Cat says ???

C Animal says ???

D Cat says Meow

15.

```
#include <iostream>

class Base {
public:
    virtual std::string GetName() { return "Base"; }
};

class Derived : public Base {
public:
    std::string GetName() const override { return "Derived"; }
};

int main()
{
    Derived d{};
    Base* b{ &d };
    std::cout << b->GetName();
    return 0;
}
```

Что выведется в консоль при выполнении данной программы?

A программа не скопилируется

B Base

C Derived

D BaseDerived

16.

```
#include <iostream>
class Base
{
public:
    ~Base() { std::cout << "Calling ~Base()\n"; }
};

class Derived: public Base
{
private:
    int* m_array;

public:
    Derived(int length) : m_array{ new int[length] } {}

    ~Derived()
    {
        std::cout << "Calling ~Derived()\n";
        delete[] m_array;
    }
};

int main()
{
    Base *base  = new Derived(5);
    delete base;
    return 0;
}
```

Произойдёт ли утечка памяти при работе данной программы?

A Нет, т.к. деструктор базового класса не является виртуальным

B Нет, т.к. в деструкторе класса Derived мы удаляем массив m_array

C Нет, т.к. массив m_array удалится автоматически

D Да, т.к. деструктор базового класса не является виртуальным

17. Выберите правильные утверждения про чисто виртуальные функции

A. Любой класс с одной или несколькими чисто виртуальными функциями становится абстрактным базовым классом, что означает, что его экземпляр не может быть создан

B. Любой производный класс должен определять тело этой функции, иначе этот производный класс также будет считаться абстрактным базовым классом

C. Сигнатура такой функции, может выглядеть, например, следующим образом:
virtual int f() = 0;

D. Чисто виртуальная функция может иметь тело функции. Т.е. может быть как объявлена, так и определена

18. Наиболее распространённое использование некоторого оператора - преобразование указателей базового класса в указатели производного класса (downcasting). При этом он также может выполнять и upcasting, а также горизонтальное преобразование. Кроме указателей он может преобразовывать и ссылки. При неудачном преобразовании указателей он возвращает nullptr, а при неудачном преобразовании ссылки выбрасывает исключение std::bad_cast.
Напишите название этого оператора динамического приведения.

19.

```
int max(int x, int y)
{
    return (x > y) ? x : y;
}
```

Мы имеем функцию определения максимума двух чисел. Выберите способ(ы) позволяющие переписать эту функцию для работы с произвольным типом переопределяющим operator> .

A

```
template <typename T>
T max(T x, T y)
{
    return (x > y) ? x : y;
}
```

B

```
template <typename StringStream>
StringStream max(StringStream x, StringStream y)
{
   return (x > y) ? x : y;
}
```

C

```
typename <template T>
T max(T x, T y)
{
    return (x > y) ? x : y;
}
```

D

```
template <class T>
T max(T x, T y)
{
    return (x > y) ? x : y;
}
```

20.

```
#include <iostream>

template <typename T>
int count(T x)
{
    static int c { 0 };
    return ++c;
}

int main()
{
    std::cout << count(1);
    std::cout << count(1);
    std::cout << count(2.3);
    std::cout << count<double>(1);

    return 0;
}
```

Что выведется в консоль при выполнении данной программы?

21.

```
template <int i>
void func() {
    i = 20;
    std::cout << i;
}

int main()
{
    func<10>();
    return 0;
}
```

Что выведет в консоль программа при выполнении?

A 20

B произойдёт ошибка компиляции

C ничего из перечисленного

D 10

22.

```
template <typename T, int count>
void Test(T x) {
    T val[count];
    for (int i = 0; i < count; ++i) {
        val[i] = x++;
        std::cout << val[i] << " ";
    }
}

int main()
{
    float data = 2.1;
    Test<float, 3>( x: data);
    return 0;
}
```

Что выведется в консоль при выполнении данной программы?

A 3.1

B 2.1 3.1

C 2.1

D 2.1 3.1 4.1

23.

```
template <int T>
struct X {
    enum val { v = T * X<T-1>::v };
};

template <>
struct X<0>
{
    enum val { v = 1 };
};

int main()
{
    std::cout << X<5>::v;
    return 0;
}
```

Что выведется в консоль при выполнении программы?

A 120

B 5

C Ошибка компиляции

D Ошибка линковки

Answer Key

1. c	2. a, b, c, d	3. c	4. e
5. b, c	6. ABC~C~B~A	7. d	8. c
9. b	10. c	11. a	12. b
13. d	14. c	15. a	16. d
17. a, b, c, d	18. dynamic_cast	19. a, b, d	20. 1212
21. b	22. d	23. a	