1. 0Что будет в результате выполнения данного кода?

struct A

{

virtual ~A()

{

}

virtual void Foo(const int value) = 0;

};

struct B : public A

{

~B()

{

}

void Foo(const int value)

{

value ? throw 'B' : throw 1;

}

};

struct C : public A

{

~C()

{

}

void Foo(const int value)

{

value ? throw 'C' : throw 1;

}

};

int main()

{

int count = 1;

A\* p = new B;

do

{

try

{

p->Foo(count);

}

catch (int i)

{

cout << i;

delete p;

if (i == 0)

{

p = new C;

count = 2;

}

else

{

break;

}

}

catch (char c)

{

cout << c;

}

--count;

}

while (true);

return 0;

}На экране мы увидим B1. Так как сначала сработает catch который принимает элемент типа данных char,после же в нем сработает cout, выведет на экран B, ,после же сработает count--, и мы попадем в catch который принимает элемент типа данных int, и там сработает cout, который выведет на экран нам 1

1. Что будет в результате выполнения данного кода?

int main()

{

try

{

throw 100;

}

catch (double e)

{

cout << "catched, e - " << e << endl;

}

return 0;

}//Не сработает так как мы в блоке try выкидываем еще один throw с ошибкой, то есть одновременно происходит раскрутка 2 стека, и программа крашется.

1. Что будет в результате выполнения данного кода?

class A

{

};

class B : public A

{

};

int main()

{

try

{

throw new B;

}

catch (A\* exception)

{

cout << "Error A" << endl;

}

catch (B\* exception)

{

cout << "Error B" << endl;

}

return 0;

}Ловит ошибку A, так как идет полиморфная обработка, плюс еще класс B наследуется от А

1. Что будет в результате выполнения данного кода?

void Foo()

{

throw 12;

}

int main()

{

try

{

Foo();

}

return 0;

}//Будет ошибка так как нет хотя бы одного catch к блоку try

1. Будут ли утечки памяти в результате выполнения данного кода? Если да, то где и почему?

class Foo

{

int m\_value;

public:

Foo()

{

throw 1;

}

};

int main()

{

try

{

Foo x;

Foo\* y = new Foo;

}

catch (...)

{

}

return 0;

}Да будут, в месте Foo \* y = new Foo; Так как создается новая динамическая память которую мы не удаляем(не освобождаем)

1. Что будет в результате выполнения данного кода?

class Foo

{

public:

Foo()

{

cout << "Constructor" << endl;

throw 1;

}

~Foo()

{

cout << "Destructor" << endl;

}

};

int main()

{

try

{

Foo x;

Foo\* y = new Foo;

}

catch (...)

{

}

return 0;

}Вызовется конструктор, и будет утечка в памяти, так как мы создаем новую динамическую память, которую мы не удаляем

1. Будут ли утечки памяти в результате выполнения данного кода? Если да, то где и почему?

class Foo

{

char\* m\_array;

int m\_value;

public:

Foo() : m\_array(new char[255])

{

throw 1;

}

~Foo()

{

delete[] m\_array;

}

};

int main()

{

try

{

Foo x;

Foo\* y = new Foo;

}

catch (...)

{

}

return 0;

}Да так как мы в блоке try создаем новую динамическую память

1. Что будет в результате выполнения данного кода?

class Foo

{

public:

Foo()

{

Initialize();

}

virtual void Initialize()

{

cout << "Foo" << endl;

}

};

class Bar : public Foo

{

public:

virtual void Initialize()

{

cout << "Bar" << endl;

}

};

int main()

{

Bar b;

return 0;

}На экране будет надписать Foo так как это виртуальный метод с ранним связыванием

1. Что будет в результате выполнения данного кода?

class Foo

{

public:

virtual void Show(int value = 0)

{

cout << value << endl;

}

};

class Bar : public Foo

{

public:

virtual void Show(int value = 1)

{

cout << value << endl;

}

};

int main()

{

Foo\* ptr1 = new Foo;

ptr1->Show();

Foo\* ptr2 = new Bar;

ptr2->Show();

return 0;

}На экране мы увидим 0 0 так как срабатывает позднее связывание, плюс 2 указатель имеет тип данных Foo\*.

1. Что будет в результате выполнения данного кода?

class Foo

{

public:

virtual void Show(int value)

{

cout << "Foo" << value << endl;

}

};

class Bar : public Foo

{

public:

virtual void Show(int value1, int value2 = 0)

{

cout << "Bar" << value1 << value2 << endl;

}

};

int main()

{

Foo\* ptr1 = new Foo;

ptr1->Show(1);

Foo\* ptr2 = new Bar;

ptr2->Show(2);

return 0;

} На экране мы увидим Foo1 Foo2 так как срабатывает позднее связывание, так как 2 указателя имеют тип данных Foo\*.

1. Что будет в результате выполнения данного кода?

class Foo

{

public:

~Foo()

{

throw 1;

}

};

int main()

{

try

{

Foo obj;

}

catch (int exception)

{

cout << exception << endl;

}

return 0;

}Высветится 1 так как в catch выводим ошибку 1, но имея в коде другие throw, у нас программа автоматически крешнится так как будут вызыватся 2 потока раскрутки исключений

1. Что будет в результате выполнения данного кода?

class Foo

{

public:

~Foo()

{

throw 1;

}

};

int main()

{

try

{

Foo obj;

throw 2;

}

catch (int exception)

{

cout << exception << endl;

}

return 0;

} у нас программа автоматически крешнится так как будут вызыватся 2 потока раскрутки исключений