**Отчёт**

В C++, время жизни объектов зависит от того, как они создаются. Если объект создается в стеке, то его время жизни совпадает со временем существования блока в стеке, в котором он был создан. Если объект создан с использованием динамического выделения памяти (например, с помощью new), то его время жизни зависит от того, когда он будет удален (удаление объекта происходит с помощью оператора delete). Если объект создан глобально, то он удалится в конце работы программы. В C#, время жизни объектов управляется сборщиком мусора. Он удаляет объект, когда на него перестают ссылаться.

a) Неявное копирование объектов возникает в C++ в двух случаях:

Когда объект передается по значению в функцию или возвращается из функции по значению

Когда происходит инициализация объекта другим объектом того же типа

b) При создании объекта класса сначала вызывается конструктор базового класса, затем конструкторы членов класса в порядке их объявления, а затем конструктор самого класса.

c) Да, метод Dispose() будет вызван при завершении блока using, даже если на объект, указанный в блоке using, есть еще одна ссылка. Это происходит потому, что метод Dispose() вызывается не по условию окончания существования объекта, а по условию завершения блока using.

d) Если на объект, указанный в using, имеется еще одна ссылка, то Dispose будет вызван только по завершению этой последней ссылки. Это можно проверить экспериментально, создав простой класс Test, реализующий интерфейс IDisposable, и используя его в блоке using.

e) Для вызова конструктора предка с параметром в C++ необходимо в теле конструктора наследника явно вызвать конструктор предка и передать в него нужный параметр.

class Child : public Test

{

public:

Child(int value) : Test(value)

{

// ...

}

};

f) В C# можно вызвать конструктор предка с параметром, используя ключевое слово base и передавая ему необходимые аргументы.

class Child : Test

{

public Child(int value) : base(value)

{

// ...

}

}

g) В C++ можно вызвать конструктор члена класса с параметром, используя список инициализации в конструкторе самого класса. Например, если у класса Aggregate есть член класса objTest, для которого нужно вызвать конструктор с параметром int, то это можно сделать следующим образом:  
class Aggregate

{

Test objTest;

public:

Aggregate(int value) : objTest(value)

{

// ...

}

};

h) В C++ ссылка - это альтернативное имя для объекта, который уже был определен. Ссылка создается с помощью оператора &, который ставится после типа данных. Ссылка может быть изменяемой или неизменяемой, и после создания ссылки нельзя изменить объект, на который она ссылается.

В .NET ссылка – это указатель на объект в управляемой куче. Ссылки в .NET всегда являются ссылками на объекты, а не на переменные, и могут быть переданы в качестве аргументов методов или возвращены из них.

i) объявление шаблонного класса начинается с ключевого слова template, за которым следует список параметров шаблона в угловых скобках, а затем определение класса. В определении класса можно использовать параметры шаблона как типы данных или константные значения.

Пример объявления шаблонного класса:

template <typename T>

class AggregateT {

public:

T m\_obj;

AggregateT() {

Test::nCount++;

cout << "Creating AggregateT object" << " " << Test::nCount << endl;

}

~AggregateT() {

Test::nCount--;

cout << "Deleting AggregateT object" << " " << Test::nCount << endl;

}

};