**Отчёт:**

Время жизни объектов в C++ и C#:

В C++ время жизни объектов определяется областью видимости, в которой они были созданы. При выходе из этой области объект уничтожается. В языке C# время жизни объектов определяется сборщиком мусора, который автоматически освобождает память, занимаемую объектами, когда на них нет ссылок.

a) Случаи неявного копирования объектов в C++:

* Когда объект передается в функцию как параметр по значению.
* Когда объект возвращается из функции по значению.

b) В C++ составные объекты создаются в порядке, указанном в списке инициализации (constructor initialization list), где сначала создаются базовые классы (потомки), а затем члены-агрегаты.

c) Интерфейс IDisposable в C# предназначен для освобождения неуправляемых ресурсов, таких как файлы, подключения к базам данных, сетевые соединения и т.д., для предотвращения утечки памяти и других проблем.

d) Если на объект, указанный в блоке using, имеется еще одна ссылка, то Dispose не будет вызван по завершению блока using. Это можно проверить на примере:

*using System;*

*public class Test : IDisposable*

*{*

*public Test()*

*{*

*Console.WriteLine("Test constructor");*

*}*

*public void Dispose()*

*{*

*Console.WriteLine("Dispose method");*

*}*

*}*

*class Program*

*{*

*static void Main()*

*{*

*Test test = new Test();*

*using (test)*

*{*

*Console.WriteLine("Inside using block");*

*}*

*Console.WriteLine("Outside using block");*

*}*

*}*

На выходе будет выведено сообщение "Test constructor", "Inside using block", и "Outside using block", но "Dispose method" не будет вызван, так как объект test еще имеет ссылку за пределами блока using.

e) Чтобы вызвать конструктор предка с параметром в C++, можно передать аргументы через список инициализации в конструкторе потомка. Пример:

*#include <iostream>*

*class Parent*

*{*

*public:*

*Parent(int value)*

*{*

*std::cout << "Parent constructor with value: " << value << std::endl;*

*}*

*};*

*class Child : public Parent*

*{*

*public:*

*Child(int value) : Parent(value)*

*{*

*std::cout << "Child constructor" << std::endl;*

*}*

*};*

*int main()*

*{*

*Child child(5);*

*return 0;*

*}*

В этом примере конструктор потомка передает аргумент value в конструктор предка при создании объекта Child.

f) Для вызова конструктора предка с параметром в C# используется ключевое слово base, например:

*class ChildClass : ParentClass*

*{*

*public ChildClass(int param) : base(param)*

*{*

*// Код конструктора*

*}*

*}*

g) Для вызова конструктора члена класса с параметром можно использовать инициализацию в теле конструктора класса или через конструктор инициализации:

*class MyClass*

*{*

*private AnotherClass member = new AnotherClass(param);*

*public MyClass(int param)*

*{*

*member = new AnotherClass(param);*

*}*

*}*

h) В C++ ссылка представляет собой указатель на объект, который не может быть переопределен или указывать на нулевой объект. В .Net ссылка также является указателем на объект, но в .Net ссылки могут указывать на null и могут быть переназначены на другой объект.

i) Шаблон класса объявляется с использованием угловых скобок и ключевого слова template. Пример объявления шаблона класса:

*template <class T>*

*class MyTemplateClass*

*{*

*public:*

*T value;*

*MyTemplateClass(T val)*

*{*

*value = val;*

*}*

*};*

*int main()*

*{*

*MyTemplateClass<int> myIntClass(5);*

*MyTemplateClass<string> myStringClass("Hello");*

*}*