1. Время жизни объектов в C++ и C#:

C++:

Автоматические объекты (выделяемые в стеке): создаются в стеке при объявлении и автоматически уничтожаются при выходе из области видимости (завершении функции или блока кода). Время жизни определяется областью видимости.

Глобальные объекты: создаются до начала выполнения функции main() и уничтожаются после завершения main(). Время жизни — вся программа.

Динамические объекты (выделяемые в куче): создаются в куче с помощью оператора new. Уничтожаются только при явном вызове оператора delete. Если не использовать delete, произойдет утечка памяти. Время жизни контролируется программистом.

Статические объекты (внутри функции): создаются при первом вызове функции, в которой они объявлены, и уничтожаются после завершения работы программы.

C#:

Объекты, созданные с помощью new: Размещаются в управляемой куче (managed heap). Их время жизни управляется сборщиком мусора (garbage collector).

Локальные переменные: размещаются в стеке. Их время жизни ограничено областью видимости, но сборщик мусора может освободить память позже, если объект больше не используется.

Сборка мусора: сборщик мусора автоматически определяет, когда объект больше не используется, и освобождает занимаемую им память. Это позволяет избежать утечек памяти, но делает время жизни объектов менее предсказуемым.

IDisposable и using: Если объект реализует интерфейс IDisposable, его можно заключить в блок using, который гарантирует вызов метода Dispose() (аналогичного деструктору) при завершении блока. Это полезно для освобождения ресурсов, таких как файлы, соединения с базой данных и т. д.

2. Ответы на вопросы:

a) Какие случаи неявного копирования объектов существуют в C++? (их 2)

Передача объекта по значению в функцию: Когда объект передается в функцию как аргумент по значению, создается копия этого объекта.

Возврат объекта по значению из функции: Когда функция возвращает объект по значению, создается копия этого объекта для передачи его вызывающей стороне.

b) В каком порядке создаются составные объекты (потомки или агрегаты в C++)?

Потомки (наследование): При создании объекта дочернего класса (Child), сначала вызывается конструктор базового класса (Test), а затем конструктор дочернего класса (Child). При уничтожении объекта дочернего класса деструкторы вызываются в обратном порядке. Агрегаты (агрегация): При создании объекта класса-агрегата (Aggregate), сначала вызывается конструктор объекта-члена (m\_objTest), а затем конструктор самого класса-агрегата (Aggregate). При уничтожении объекта класса-агрегата деструкторы вызываются в обратном порядке.

c) Для чего предназначен интерфейс IDisposable в C#?

Интерфейс IDisposable в C# предназначен для освобождения неуправляемых ресурсов (unmanaged resources), таких как дескрипторы файлов, соединения с базами данных, сетевые сокеты и т. д., которые не управляются сборщиком мусора.

d) Будет ли вызван Dispose по завершению блока using, если на объект, указанный в using имеется еще одна ссылка?

Да, метод Dispose() будет вызван по завершении блока using, даже если на объект, указанный в using, имеется еще одна ссылка. Блок using гарантирует вызов Dispose() независимо от того, есть ли другие ссылки на объект.

e) Как вызвать конструктор предка с параметром в C++?

Чтобы вызвать конструктор предка с параметром в C++, необходимо использовать список инициализации членов в конструкторе дочернего класса.

f) Как вызвать конструктор предка с параметром в C#?

В C# используется аналогичный синтаксис, как и в C++.

g) Как вызвать конструктор члена класса с параметром?

h) Опишите разницу между понятием «ссылка» в C++ и в .Net.

C++: Ссылка ( & ) — это просто другое имя (псевдоним) для существующей переменной. Ссылка должна быть инициализирована при объявлении и не может быть переназначена (то есть она всегда ссылается на один и тот же объект). Ссылки не могут быть nullptr.

.NET (C#): Ссылка (reference) — это указатель на объект, находящийся в управляемой куче (managed heap). Ссылка может быть null (если она не указывает ни на какой объект) и может быть переназначена для указания на другой объект.

i) Как объявить и использовать шаблон класса?

Объявление шаблона класса в C++:

template <typename T>

class MyTemplateClass {

public:

MyTemplateClass(T value) : m\_value(value) {}

T getValue() const { return m\_value; }

private:

T m\_value;

};

cpp

Использование шаблона класса:

MyTemplateClass<int> intObject(10);

MyTemplateClass<std::string> stringObject("Hello");

std::cout << intObject.getValue() << std::endl; // Вывод: 10

std::cout << stringObject.getValue() << std::endl; // Вывод: Hello

cpp