Содержание

[ОБЩАЯ ИНФОРМАЦИЯ 2](#_heading=h.gjdgxs)

[Тема занятия](#_heading=h.30j0zll) 2

[Цели и задачи 2](#_heading=h.1fob9te)

[Ожидаемый результат 2](#_heading=h.3znysh7)

[Структура занятия 3](#_heading=h.tyjcwt)

[ОПИСАНИЕ ЗАНЯТИЯ 4](#_heading=h.3dy6vkm)

# ОБЩАЯ ИНФОРМАЦИЯ

### Тема занятия

Композиция объектов.

### Цели и задачи

* Рассказать о перегрузке основных операторов в С++
* Рассказать о деструкторах

### Ожидаемый результат

По результатам занятия Слушатель должен знать:

* Как использовать полиморфизм для взаимодействия объектов класса между собой и не только

По результатам занятия Слушатель должен уметь:

* Создавать иерархическую структуру с помощью наследования классов
* Структурировать проект (разбиение определение класса на \*.h и \*.cpp файлы)

### Структура занятия

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Тайминг занятия** | |  |  |  | Таблица 1 |
| № | Этапы | Что делает преподаватель | Что делает Слушатель | Время | Общее время |
| 1 | Приветственное слово преподавателя |  |  | 5 мин. | 5 мин. |
| 2 | Повторение пройденного |  |  | 5 мин. | 40 мин. |
| 3 | Теоретическая часть |  |  | 30 мин. |
| 4 | Вопросы по теоретической части |  |  | 5 мин. |
|  | *Перерыв* |  |  | 15 мин. | 15 мин. |
| 5 | Практическая часть |  |  | 40 мин. | 45 мин. |
| 6 | Рефлексия и вопросы |  |  | 5 мин. |

# ОПИСАНИЕ ЗАНЯТИЯ

**Теоретическая часть**

**Разделения объявления класса и его реализации**

Создание новых классов в отдельных файлах является хорошим тоном, так как это делает код более удобочитаемым и его легче поддерживать в дальнейшем. Возьмём за правило- для классов создавать 2 файла:

\* - название класса.

\*.h – заголовочный файл, содержащий в себе объявления функций (их прототипы) и объявления переменных (полей класса).

\*.cpp – исходный файл, который предоставляет реализацию методов, описанных (прототипами) в \*.h файле.

**Оператор разрешения контекста (оператор доступа к области видимости)**

Два двоеточия – оператор доступа к области видимости (используется для определения методов класса, которые уже были объявлены в пространстве имён класса в заголовочном файле)

Для создания объекта класса, объявленного в заголовочном файле (.h), достаточно подключить в текущий файл заголовочный файл класса.

**Дестркутор**

Помимо конструкторов, также в классе объявляется и деструктор. Как и конструктор, деструктор является специальной функцией/методом. Он вызывается при уничтожении или удалении объекта класса. Таким образом, «жизненный цикл» объекта класса выглядит следующим образом:

1. Создание объекта (вызов конструктора)
2. Работа с объектом (вызов его методов и т д)
3. Уничтожение объекта (вызов деструктора)

В отличие от конструктора (которых может быть сколь угодно много), деструктор только один. Он не принимает никаких аргументов и ничего не возвращает.

Объекты класса уничтожаются при выходе за пределы видимости вызова конструктора (например, тело функции main), или при применении выражения delete к указателю, направленному на объект класса.

**Имя у деструктора такое же, как и у класса, только с префиксом тильда (~).**

**Список инициализации**

C++ предоставляет удобный синтаксис для инициализации элементов класса, который называется списком инициализаторов (также называется инициализацией полей в конструкторе).

Этот класс имеет две переменные, Var и constVar. У него также есть конструктор, который принимает два параметра, которые используются для инициализации переменных-членов.

Запуск этого кода возвращает ошибку, потому что одна из его переменных-членов является константой, к которой не может быть присвоено значение после объявления. В таких случаях, список инициализаторов может быть использован для присваивания значений переменным-членам.

В этом синтаксисе список инициализации следует за параметрами конструктора. Список начинается с двоеточия (:), а затем следуют разделенные запятыми инициализируемые переменные вместе со значениями.

Cинтаксис - переменная(значение) для присваивания значений.

Список инициализации элементов может быть использован для регулярных переменных и должен быть использован для константных переменных.

**Пример композиции объектов**

В реальном мире, сложные объекты обычно состоят из маленьких, более простых объектов. Например, машина собрана с использованием металлической рамы, двигателя, колес и огромного количества других деталей. Этот процесс называется композицией.

В C++ объектная композиция подразумевает использование классов в качестве переменных-членов в других классах

**Дружественные функции**

Обычно, скрытые поля класса недоступны извне класса. Однако, если объявление функции как не-член класса с использованием ключевого слова friend позволяет получить доступ к скрытым полям класса. Это выполняется путем включения объявления этой внешней функции внутри класса, с предшествующим ключевым словом friend.

someFunc(), которая не является функцией-элементом класса, является дружественной функцией класса MyClass и имеет доступ к его скрытым полям.

Чтобы сделать элементы доступными, в классе в определении должна быть объявлена функция с использованием ключевого слова friend. Вы не можете "сделать" функцию дружественной для класса без "согласия" класса**.**

**Ключевое слово “this”**

Что если названия скрытых полей класса совпадут с аргументами, передаваемыми в конструктор?

**This**

Каждый объект в C++ имеет доступ к его собственному адресу через указатель, который называется this.

Внутри функции-члена this может быть использовано для ссылки на вызывающий объект.

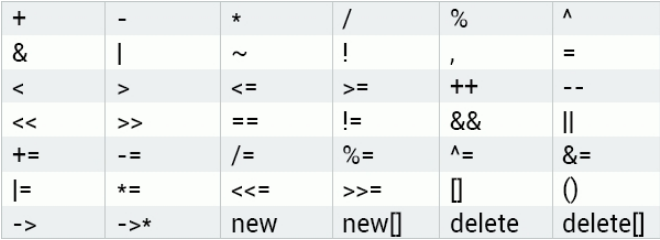
Дружественные функции не имеют указателя this, потому что дружественные функции **не являются элементами класса.**

**Перегрузка Операторов**

Большинство операторов в C++ могут быть переопределены или перегружены.

Таким образом, операторы могут быть использованы также с определенными пользователем типами (например, позволяют вам складывать два объекта вместе).

В данной таблице указаны операторы, которые могут быть перегружены.



Перегруженные операторы - это функции, определенные с помощью ключевого слова operator, за которым следует символ определяемого оператора.

Перегруженный оператор схож с другими функциями в том, что он тоже имеет возвращаемый тип и список параметров.

В нашем примере мы будем перегружать оператор +. Он будет возвращать объект нашего класса и принимать объект нашего класса в качестве параметра.

operator+

**Практическая часть**

**Задание 1**

Определить класс Human, который содержит такие поля (члены класса): имя, фамилию и возраст, публичные – методы ввода данных и отображения их на экран. Определить класс Pupil, который содержит такие поля (члены класса): имя, фамилию, возраст, класс, номер школы и средний бал. Публичные – методы ввода данных и отображения их на экран. Объявить два объекта класса, внести данные и показать их.

\*

Сделать конструктор по умолчанию и конструктор с помощью инициализатора

**Задание 2**

Определить класс fraction, который содержит такие поля (члены класса): закрытые – числитель и знаменатель, публичные – конструктор по умолчанию, конструктор, методы ввода данных (принимается числитель и знаменатель отдельно) и отображение дроби на экран в обычном виде и десятичном

**\***

Упрощение дробей (на этапе создания объекта)

Пример.

(25/100 –> 1/4)

(Файл с кодом лежит в папке code/)