Конструкторы и деструкторы

Прикладное программирование

Конструктор и деструктор

При создании объектов одной из наиболее широко используемых операций является инициализация элементов данных объекта. Единственным способом, с помощью которого можно обратиться к частным элементам данных, является использование функций класса. Чтобы упростить процесс инициализации элементов данных класса, C++ использует специальную функцию, называемую конструктором, которая запускается для каждого создаваемого объекта.

Деструктор запускается при уничтожении объекта. Деструктор обычно используется, если при уничтожении объекта нужно освободить память, которую занимал объект.

Основные концепции:

- Конструктор представляет собой метод класса, который облегчает создаваемым программам инициализацию элементов данных класса.
- Конструктор имеет такое же имя, как и класс.
- Конструктор не имеет возвращаемого значения.
- Каждый раз, когда ваша программа создает переменную класса, C++ вызывает конструктор класса, если конструктор существует.
-

Основные концепции:

- Многие объекты могут распределять память для хранения информации; когда уничтожается такой объект, C++ будет вызывать специальный деструктор, который может освобождать эту память, очищая ее после объекта.
- Деструктор имеет такое же имя, как и класс, за исключением того, что необходимо предварять его имя символом тильды (~).
- Деструктор не имеет возвращаемого значения.

Создание простого конструктора

- Добавим в класс Students конструктор, который будет принимать имя и фамилию ученика, и сохранять эти значения в соответствующих переменных класса.

Создание простого конструктора

- // Конструктор Students
- Students::Students(std::string name, std::string last_name)
- {
- Students::set_name(name);
- Students::set_last_name(last_name);
- }

Создание простого конструктора

- При создании нового объекта, мы должны передать конструктору имя и фамилию студента. Иначе компиляция программы завершится с ошибкой.
- std::string name = “Иван”
- std::string last_name =“Иванов”
- Students *student = new Students(name, last_name);

Создание простого конструктора

- Теперь добавим прототип конструктора в файл students.h.
- /* students.h */
- #pragma once /* Защита от двойного подключения заголовочного файла */
- #include <string>

Создание простого конструктора

- class Students {
- public:
- // Конструктор класса Students
- Students(std::string, std::string);
- // Установка имени студента
- void set_name(std::string);
- // Получение имени студента
- std::string get_name();

Создание простого конструктора

- // Установка фамилии студента
- void set_last_name(std::string);
- // Получение фамилии студента
- std::string get_last_name();
- // Установка промежуточных оценок
- void set_scores(int []);

Создание простого конструктора

- // Установка среднего балла
- void set_average_ball(float);
- // Получение среднего балла
- float get_average_ball();
- private:
- // Промежуточные оценки
- int scores[5];
-

Создание простого конструктора

- // Средний балл
- 	float average_ball;
- 	// Имя
- 	std::string name;
- 	// Фамилия
- 	std::string last_name;
- };

Создание простого конструктора

- В файле students.cpp определим сам конструктор.
- /* students.cpp */
- #include <string>
- #include <fstream>
- #include "students.h"

Создание простого конструктора

- / Конструктор Students
- Students::Students(std::string name, std::string last_name)
- {
-     Students::set_name(name);
-     Students::set_last_name(last_name);
- }

Создание простого конструктора

- // Установка имени студента
- void Students::set_name(std::string student_name)
- {
-     Students::name = student_name;
- }

Создание простого конструктора

- // Получение имени студента
- std::string Students::get_name()
- {
-     return Students::name;
- }

Создание простого конструктора

```
// Установка фамилии студента
void Students::set_last_name(std::string
student_last_name)
{
   Students::last_name = student_last_name;
}
```

Создание простого конструктора

- // Получение фамилии студента
- std::string Students::get_last_name()
- {
-     return Students::last_name;
- }

Создание простого конструктора

```
// Установка промежуточных оценок
void Students::set_scores(int scores[])
{
    int sum = 0;
    for (int i = 0; i < 5; ++i) {
        Students::scores[i] = scores[i];
        sum += scores[i];
    }
```

Создание простого конструктора

- // Установка среднего балла
- void Students::set_average_ball(float ball)
- {
-     Students::average_ball = ball;
- }

Создание простого конструктора

```
// Получение среднего балла
float Students::get_average_ball()
{
    return Students::average_ball;
}
```

Создание простого конструктора

- В main() мы принимаем от пользователя имя и фамилию ученика, и сохраняем их во временных локальных переменных. После этого создаем новый объект класса Students, передавая его конструктору эти переменные.

Создание простого конструктора

- /* main.cpp */
- #include <iostream>
- #include "students.h“
- int main(int argc, char *argv[])
- {
- // Локальная переменная, хранящая имя ученика
- std::string name;
- // И его фамилию
- std::string last_name;

Создание простого конструктора

- // Ввод имени
- std::cout << "Name: ";
- getline(std::cin, name);
- // И фамилии
- std::cout << "Last name: ";
- getline(std::cin, last_name);

Создание простого конструктора

- // Передача параметров конструктору
- Students *student = new Students(name, last_name);
- // Оценки
- int scores[5];
- // Сумма всех оценок
- int sum = 0;

Создание простого конструктора

```
// Ввод промежуточных оценок
  for (int i = 0; i < 5; ++i) {
    std::cout << "Score " << i+1 << ": ";
    std::cin >> scores[i];
    // суммирование
    sum += scores[i];
  }
```

Создание простого конструктора

- // Сохраняем промежуточные оценки в объект класса Student
- student->set_scores(scores);
- // Считаем средний балл
- float average_ball = sum / 5.0;
- // Сохраняем средний балл
- student->set_average_ball(average_ball);

Создание простого конструктора

- // Выводим данные по студенту
- std::cout << "Average ball for " << student->get_name() << " "
- << student->get_last_name() << " is "
- << student->get_average_ball() << std::endl;
- // Удаление объекта student из памяти
- delete student;
- return 0;
- }

Сохранение оценок в файл

- Чтобы после завершения работы с программой, все данные сохранялись, необходимо записывать их в текстовый файл.
- Пример файла с оценками представлен в следующем виде:
- Андрей Сидоров 5 5 3 4 5
- Иван Иванов 5 3 3 3 3

Сохранение оценок в файл

- Для работы с файлами воспользуемся библиотекой fstream, которая подключается в заголовочном файле с таким же именем.
- #include <fstream>
- // Запись данных о студенте в файл
- void Students::save()
- {
-     std::ofstream fout("students.txt", std::ios::app);

Сохранение оценок в файл

```
fout << Students::get_name() << " "
       << Students::get_last_name() << " ";
    for (int i = 0; i < 5; ++i) {
        fout << Students::scores[i] << " ";
    }
    fout << std::endl;
    fout.close();
}
```

Сохранение оценок в файл

- Переменная fout — это объект класса ofstream, который находится внутри библиотеки fstream. Класс ofstream используется для записи каких-либо данных во внешний файл. У него тоже есть конструктор. Он принимает в качестве параметров имя выходного файла и режим записи.

Сохранение оценок в файл

В данном случае, мы используем режим добавления — std::ios:app (append). После завершения работы с файлом, необходимо вызвать метод close() для того, чтобы закрыть файловый дескриптор.

Сохранение оценок в файл

- тобы сохранить оценки студента, мы будем вызывать только что созданный метод save().
- Students student = new Students(“Иван", “Иванов");
- student->save();

Деструктор

- Логично было бы сохранять все оценки после того, как работа со студентом закончена. Для этого создадим деструктор класса Students, который будет вызывать метод save() перед уничтожением объекта.
- // Деструктор Students
- Students::~Students()
- {
-     Students::save();
- }

Деструктор

- Добавим прототипы деструктора и метода save() в students.h.
- /* students.h */
- #pragma once /* Защита от двойного подключения заголовочного файла */
- #include <string>

Деструктор

```
class Students {
    public:
        // Запись данных о студенте в файл
        void save();
        // Деструктор класса Students
        ~Students();
```

Деструктор

- // Конструктор класса Students
- Students(std::string, std::string);
- // Установка имени студента
- void set_name(std::string);
- // Получение имени студента
- std::string get_name();

Деструктор

- // Установка фамилии студента
- void set_last_name(std::string);
- // Получение фамилии студента
- std::string get_last_name();

Деструктор

- // Установка промежуточных оценок
- void set_scores(int []);
- // Получение массива с промежуточными оценками
- int *get_scores();
- // Получение строки с промежуточными оценками
- std::string get_scores_str(char);

Деструктор

- // Установка среднего балла
-         void set_average_ball(float);
-         // Получение среднего балла
-         float get_average_ball();
-     private:
-         // Промежуточные оценки
-         int scores[5];

Деструктор

```
// Средний балл
        float average_ball;
        // Имя
        std::string name;
        // Фамилия
        std::string last_name;
};
```

Деструктор

- И определим эти функции в students.cpp.
- /* students.cpp */
- #include <string>
- #include <fstream>

- #include "students.h"

Деструктор

- `// Деструктор Students`
- `Students::~Students()`
- `{`
- `    Students::save();`
- `}`

Деструктор

```
// Запись данных о студенте в файл
void Students::save()
{
    std::ofstream fout("students.txt", std::ios::app);
    fout << Students::get_name() << " "
         << Students::get_last_name() << " ";
    for (int i = 0; i < 5; ++i) {
        fout << Students::scores[i] << " ";
    }
    fout << std::endl;
    fout.close();
```

Деструктор

- // Конструктор Students
- Students::Students(std::string name, std::string last_name)
- {
- Students::set_name(name);
- Students::set_last_name(last_name);
- }

Деструктор

- // Установка имени студента
- void Students::set_name(std::string student_name)
- {
-     Students::name = student_name;
- }

Деструктор

- // Получение имени студента
- std::string Students::get_name()
- {
-     return Students::name;
- }

Деструктор

- // Установка фамилии студента
- void Students::set_last_name(std::string student_last_name)
- {
-     Students::last_name = student_last_name;
- }

Деструктор

- // Получение фамилии студента
- std::string Students::get_last_name()
- {
-     return Students::last_name;
- }

Деструктор

```
// Установка промежуточных оценок
void Students::set_scores(int scores[])
{
    int sum = 0;
    for (int i = 0; i < 5; ++i) {
        Students::scores[i] = scores[i];
        sum += scores[i];
    }

}
```

Деструктор

- // Получение массива с промежуточными оценками
- int *Students::get_scores()
- {
-     return Students::scores;
- }

Деструктор

- // Установка среднего балла
- void Students::set_average_ball(float ball)
- {
-     Students::average_ball = ball;
- }

Деструктор

- / Получение среднего балла
- float Students::get_average_ball()
- {
-     return Students::average_ball;
- }