Структуры, указатели, модули, файлы

Юрий Литвинов
yurii.litvinov@gmail.com

01.10.2019

Структуры

- Способ группировки родственных по смыслу значений
- Структура — это тип
  - В памяти представляется как поля, лежащие друг за другом, возможно, с “дырками” (padding)
  - Объявляется, гм, вне функции
- Объявление структуры:

```
struct Point
{
    int x;
    int y;
};
```

Указатели и ссылки

- Указатель — адрес ячейки в памяти
- Ссылка — “синоним”, просто другое название для ячейки в памяти
  - Можно считать, что ссылка — это указатель, который не надо разыменовывать (и нельзя менять)
- Структуры и указатели настолько часто используются вместе, что есть оператор -> (разыменовать указатель на структуру и обратиться к её полю)
  - ```
    Point *p = new Point { 10, 20 };
    printf("(%d, %d)", p->x, p->y);
    ```

    Или

    ```
    auto p = new Point { 10, 20 };
    printf("(%d, %d)", p->x, p->y);
    ```
  - То же самое, что (*p).x и (*p).y
    ```

Файлы

- Последовательность байтов на диске
  - Бывают “сырые” и “текстовые”
    - Самому файлу всё равно, это лишь способы интерпретации его содержимого
  - Режимы доступа: r, w, a, r+, w+, a+
  - Курсор
  - EOF
- Сишные функции
  - fopen, fclose, fprintf, fscanf, fseek, ftell, fgets
- Файлы надо не забывать закрывать

Пример, как писать в файл

```
FILE * out = fopen("ololo.txt", "w");
fwrite("Ololo", sizeof(char), 6, out);
fclose(out);
```

Пример, как читать из файла

```
#include <stdio.h>

int main() {
    FILE *file = fopen("test.txt", "r");
    if (!file) {
        printf("file not found!");
        return 1;
    }
    char *data[100] = {};
    int linesRead = 0;
    while (!feof(file)) {
        char *buffer = new char[100];
        const int readBytes = fscanf(file, "%s", buffer);
        if (readBytes < 0) {
            break;
        }
        data[linesRead] = buffer;
        ++linesRead;
    }
    fclose(file);
    ...
}
```

Тонкости

- Чтение строки целиком: fscanf(file, "%[^\n]", buffer);
- Или: fgets(buffer, **sizeof**(buffer), file);
- Working directory
  - Свойства проекта -> Отладка -> Рабочая папка
  - По умолчанию $(ProjectDir), папка с .vcxproj

Модули

- Способ группировки кода в логически обособленные группы
- В C++ это реализуется с помощью заголовочных файлов и файлов с реализацией
  - .h и .cpp
- В отдельный модуль выносятся объявления типов данных и функции, которые делают одно дело
  - Например, разные функции сортировки
  - Или всё для работы с матрицами
- В интерфейсную часть модуля выносится только то, что может использовать другой код
  - Меньше знаешь — крепче спишь
- Функции, используемые только для реализации, пишутся только в .cpp-файле
  - Например, функция разделения массива для быстрой сортировки или swap

Модули

Заголовочный файл:

```
#pragma once

// Комментарий к функции 1
int function1(int x, int y);

// Комментарий к функции 2
void function2();
```

.cpp-файл:

```
#include <имя заголовочного файла.h>

#include <все остальные библиотеки>

int function1(int x, int y)
{
    ...
}

void function2()
{
    ...
}
```