Лекция 1

А.Ф. ЗУБАИРОВ

Классы памяти

- Каждый идентификатор в программе имеет имеет *класс памяти*, *период хранения (время жизни)*, *область действия* и *тип компоновки*.
- Четыре класса памяти обозначаются спецификаторами класса памяти: `auto`, `register`, `extern`, `static`.
- Класс памяти определяет период хранения идентификатора (время, в течение которого идентификатор существует в памяти), область действия (возможность обращения к нему из различных частей программы), тип компоновки (возможность обращения из разных файлов).

Автоматический период хранения

- Могут иметь только переменные.
- Идентификаторы классов `auto` и `register`.
- Переменные создаются, когда управление получает блок, в котором они объявлены; существуют, пока блок активен; уничтожаются при выходе из блока.
- По умолчанию локальные переменные функций относятся к классу `auto`.

```
float x = 1.0, y = 0.0;
auto float x = 1.0, y = 0.0;
```

Автоматический период хранения

- Спецификатор класса памяти `register` позволяет загрузить автоматическую переменную в один из регистров процессора.
- Объявление `register` может быть проигнорировано, если не окажется достаточного числа регистров.
- Обычно объявление `register` не требуется, т.к. в процессе оптимизации компиляторы сами распознают частоиспользуемые переменные и размещают их в регистрах.

```
register int counter = 1;
```

Статический период хранения

- Могут иметь переменные и функции.
- Идентификаторы классов extern и static.
- Переменные и функции существуют с того момента, как программа начинает выполняться – память распределяется и инициализируется один раз, когда программа запускается.
- Идентификаторы со статическим периодом хранения:
  - внешние идентификаторы (глобальные переменные и имена функций);
  - локальные переменные со спецификатором класса памяти `static`.

Статический период хранения

- Глобальные переменные и имена функций имеют по умолчанию класс extern.
- Глобальные переменные создаются при объявлении их вне любого определения функции и сохраняют значение в течение всего времени выполнения программы.
- Локальные переменные с классом `static` известны только той функции, в которой они определены, но сохраняют свои значения и после выхода из функции.
- Все числовые переменные со статическим хранением инициализируются нулём (`0` либо `NULL`).

ссылка

Модификатор const

- Модификатор const даёт возможность сообщить компилятору о том, что значение переменной не должно изменяться.

```
// допустимо
int i = 0;
i += 2;

// недопустимо
const int i = 0;
i += 2;
```

Передача параметров по значению

- Если передаваемое функции значение не изменяется (или не должно быть изменено) в теле функции, оно должно объявляться с модификатором const, чтобы гарантировать невозможность даже случайного изменения.

```
int foo(const int a, const int b) {
    return a + b;
}
```

Передача функции указателя

- Изменяемый указатель на изменяемые данные;
- Изменяемый указатель на неизменяемые данные – указатель может изменяться, но элемент данных, на которые он указывает, не может изменяться;
- Неизменяемый указатель на изменяемые данные – указатель всегда указывает на одно и то же место, а данные, расположенные по этому адресу могут изменяться (например, массивы);
- Неизменяемый указатель на неизменяемые данные – указатель указывает на одно место в памяти, и данные по этому адресу не могут изменяться.

Аргументы командной строки

- Главная функция main (_tmain) может получать исходные данные через аргументы командной строки.
- Для этого необходимо включить в функцию параметры int argc, char *argv[] (_TCHAR* argv[]).
- argc – число аргументов в командной строке;
- argv – массив строк, в котором сохраняются имеющиеся в командной строке аргументы.
- Обычное использование аргументов включает вывод аргументов на печать, передачу опций, передачу программе имён файлов.
- Нулевой элемент argv[0] массива указателей ссылается на строку символов, содержащую имя самой команды и поэтому параметр argc всегда имеет значение большее или равное единице.

Аргументы командной строки

- concat пано рама
- argc - 3
- argv[0] – concat
- argv[1] – пано
- argv[2] – рама