Программирование на языке С

Модуль 8.
КЛАССЫ ПАМЯТИ

- Время жизни и область видимости объекта
- Декларации на внутреннем и внешнем уровнях
- Модификаторы – `auto`, `register`, `static`, `extern`
- Динамическое распределение памяти
- Определяемые типы `typedef`

ТЕКСТ ПРОГРАММЫ

содержит:

1. **Директивы препроцессора**.
2. **Объявления объектов:**

В Си ***3 типа объектов программ***:

1. Переменные

2. Функции

3. Типы данных

3. **Определения** – задают прямо или косвенно начальные состояния объектов программ (для переменных).
4. **Инструкции компилятору** – определяют последовательность и набор операций над объектами программы.

В Си все объявления, определения, инструкции компилятору завершаются **;**

Время жизни и область видимости объектов

- **Блок** — расширение составного оператора, включающее в себя описание или определение объектов (например, переменных). Начало и конец блока маркируется парными фигурными скобками { }

- **Область действия**
- **Область видимости** — фрагмент исходного кода, в котором известны идентификатор и тип объекта. Примерами областей видимости являются блок, единичный файл, все файлы программы

- **Время жизни** — период, на протяжении которого объект существует (для переменной — выделена оперативная память)

Декларации на внутреннем и внешнем уровнях

Если объект объявлен на внутреннем уровне (внутри блока), он видим только в этом блоке и в блоках, вложенных в него.

Если объект объявлен на внешнем уровне, он видим во всех блоках от точки объявления до конца **данного файла**.

Классы памяти

В языке Си четыре класса памяти:

auto - автоматический.
extern - внешний.
static - статический.
register - регистровый.

Классы памяти переменных (начало)

Квали-фикатор	Способ определе-ния	Время жизни	Область видимости	Место выделения памяти
auto	локально (в блоке)	до выхода из блока	блок	стек
register	локально (в блоке)	до выхода из блока	блок	регистры ЦП (если возможно), стек
static	локально (в блоке)	до завершения программы	блок	сегмент данных программы
static	глобально	до завершения программы	до конца файла	сегмент данных программы

Классы памяти переменных (окончание)

Квали-фикатор	Способ определе-ния	Время жизни	Область видимости	Место выделения памяти
`extern`	локально (в блоке, только описание)	до выхода из блока	блок	не выделяется
`extern`	глобально (только описание)	до завершения программы	до конца файла	не выделяется

Примечания:

- по умолчанию локальным переменным назначается класс памяти auto, глобальным — класс памяти static;
- к регистровым переменным неприменима операция &;
- переменные, имеющие тип памяти static, могут инициализироваться константными выражениями и не могут инициализироваться, например, адресами автоматических переменных; в отсутствие явной инициализации статическим переменным присваивается нулевое значение; при наличии выражения инициализации глобальной статической переменной квалификатор `static` разрешается опускать

Классы памяти функций

Квали-фикатор	Способ определе-ния	Время жизни	Область видимости	Место выделения памяти
`static`	глобально	—	до конца файла	—
`extern`	глобально	—	во всех файлах программы	—

Примечание:

- по умолчанию функциям назначается класс памяти extern

РАБОТА С ДИНАМИЧЕСКИМИ ОБЪЕКТАМИ ПРОГРАММ

ДИНАМИЧЕСКОЕ РАСПРЕДЕЛЕНИЕ ПАМЯТИ ПОЗВОЛЯЕТ ВЛИЯТЬ НА ВРЕМЯ ЖИЗНИ ПЕРЕМЕННЫХ!

1.Объявить указатель для динамического объекта (переменной, массива).
2.Распределить память под динамический объект (переменную, массив):
для этого используются функции стандартной библиотеки:
malloc – распределить блок памяти из заданного числа байтов;
calloc – распределить блок памяти из заданного числа элементов, каждый из заданного числа байтов и инициализировать все элементы нулями;
realloc – перераспределить ранее выделенную память, добавив или удалив её часть.
3.Проверить успешность выполнения этапа №2. Указатель на выделенный блок памяти должен быть не Ноль.
4.Если память действительно распределена, то можно работать с объектом (переменной, массивом).
5.Освободить память после того, как динамический объект (переменная, массив) стал не нужен. Для этого используется функция стандартной библиотеки **free.**

Создание псевдонимов для типов

- **Как создать?**

  typedef *<существующий типа> <псевдоним>*;

- **Зачем?**
  - Наглядность кода:

    double function();

    или

    typedef double area_t;
    area_t function();

  - Внесение изменений в код:
  - Упрощение кода:

Список литературы

- [Кнут08] Кнут Д.Э. Искусство программирования / Пер. с англ. — Т. 3. Сортировка и поиск. — 2-е изд. — М.: Вильямс, 2008. — 824 с.
- [КР92] Керниган Б., Ритчи Д. Язык программирования Си / Пер. с англ. — М.: Финансы и статистика, 1992. — 272 с.
- [КР06] Керниган Б., Ритчи Д. Язык программирования С / Пер. с англ. — М.: Вильямс, 2006. — 304 с.
- [Под04] Подбельский В.В., Фомин С.С. Программирование на языке Си. – 2-е доп. изд. – М., Финансы и статистика, 2004. – 600 с.
- [Уэз82] Уэзерелл Ч. Этюды для программистов / Пер. с англ. — М.: Мир, 1982. — 288 с.