Московский государственный университет им. М. В. Ломоносова
Факультет вычислительной математики и кибернетики

Алгоритмы и алгоритмические языки
Лекция 10

9 октября 2019 г.

Преобразование типа указателя

Разрешено также преобразование целого в указатель и наоборот (поведение определяется реализацией). Однако пользоваться этим нужно очень осторожно.

```
aux = (void *) -1;
```

Допускается присваивание указателя типа `void *` указателю любого другого типа (и наоборот) без явного преобразования типа указателя. Это позволяет использовать `void *`, когда тип объекта неизвестен.

> Использование типа `void *` в качестве параметра функции позволяет передавать в функцию указатель на объект любого типа.

Допускается неявное преобразование *менее* квалифицированного типа указателя к *более* квалифицированному (`int *` $\rightarrow$ `const int *`).

Константа `NULL` неявно преобразуется к любому другому типу указателя.

Указатели и массивы

Указатель на первый элемент массива можно создать, присвоив переменной типа «указатель на тип элемента массива» имя массива без индекса:

```
int array[15];
int *p, *q;
p = array;
q = &array[0];
```

`p` и `q` указывают на начало массива `array[15]`

Значение `array` изменить нельзя, а значение `p` — можно

`array` не является *l*-значением, а `p` — является

- `array = p; array++` — писать нельзя (это ошибки)
- `p = array; p++` — писать можно (и нужно)

Указатели и массивы

Индексирование указателей

```
int *p, a[10]; /*  два способа присвоить 100 */
               /*  6-ому элементу массива a[10] */
p = a;
*(p + 5) = 100; /*  адресная арифметика  */
p[5] = 100;     /*  индексирование указателя  */
```

Сравнение указателей

Если `p` и `q` являются указателями на элементы одного и того же массива и `p < q`, то:

`q - p + 1` равно количеству элементов массива от `p` до `q` включительно.

Можно написать:

```
if (p < q)
  printf ("p ссылается на меньший адрес, чем q");
```

Массивы указателей

Указатели могут быть собраны в массив.

```
int *mu[27]; /* это массив из 27 указателей на int */
int (*um)[27]; /* это указатель на массив из 27 int */

static void error (int errno) {
  static char *errmsg[] = {
    "переменная уже существует",
    "нет такой переменной",
    <...>
    "нужно использовать переменную-указатель"
  };
  printf ("Ошибка: %s\n", errmsg[errno]);
}
```

Имя массива указателей — пример многоуровневого указателя.
Массив `errmsg` можно представить как `char **errmsg`.

Функции

Объявление функции: `return-type func(type1 arg1, type2 arg2, ..., typen argn);`

```
int atoi (char s[]);
void QuickSort (char *items, int count);
```

> Тип возвращаемого значения **void** означает, что функция не возвращает значения.

Определение функции: `func-decl { body }`

Областью действия функции является **весь** программный файл, в котором она объявлена, начиная со строки, содержащей её объявление.

> Если в программном файле вызывается какая-либо функция, она *обязательно должна быть объявлена в этом программном файле до её вызова.*

Директива препроцессора `#include <имя_библиотеки.h>` вставляет в программу объявления всех функций соответствующей библиотеки

Вызов функции

Если функция возвращает значение, то её результатом можно пользоваться в выражениях: `v = f(); a = f(y) + 2;`

Если функция не возвращает значений, то вызов выглядит просто как `f(args);`

В языке Си все аргументы передаются по значению (т.е. передаются только значения аргументов, и эти значения копируются в локальную область памяти функции).

Если аргументом является указатель, его значением может быть адрес объекта вызывающей функции, что обеспечивает вызываемой функции доступ к объекту.

Указатели и аргументы функций

Используя аргументы-указатели, функция может обращаться к объектам вызвавшей её функции.

Использование указателей позволяет избежать копирования сложных структур данных: вместо этого передаются указатели на эти структуры.

Пример. Функция `void swap(int x, int y);` меняет местами значения переменных `x` и `y`.

Неправильно:

```
void swap (int x, int y)
{
  int tmp;
  tmp = x;
  x = y;
  y = tmp;
}
```

Правильно:

```
void swap (int *px, int *py)
{
  int tmp;
  tmp = *px;
  *px = *py;
  *py = tmp;
}
```

Вызов функции

Массив всегда передается с помощью указателя на его первый элемент.

```
int asum1d (int a[], int n) {
  int s = 0;
  for (int i = 0; i < n; i++)
    s += a[i];
  return s;
}
```

Можно объявить массив `a` в списке параметров как `const a[]`.

Функции с переменным числом параметров:

```
int scanf (const char *, ...);
```

Всегда должен быть явно задан хотя бы один параметр.
После многоточия не должно быть других явных параметров.

Обработка переменных параметров — файл `stdarg.h`, макросы `va_start/va_arg/va_end`.