Учебный Центр компьютерного обучения «СПЕЦИАЛИСТ» при МГТУ им. Н.Э.Баумана

Язык программирования C

В.Г.Тетерин – Microsoft Solution Developer (Visual C++)
teterin@specialist.ru

Центр компьютерного обучения «СПЕЦИАЛИСТ» при МГТУ им. Н.Э.Баумана®

МОДУЛЬ 6

МАССИВЫ

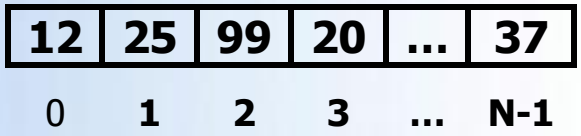
Модуль 6. Массивы

- Декларация массивов и их размещение в памяти
- Индексация элементов массива
- Алгоритмы суммирования, поиска и сортировки

Массивы

- **Массив (индексный)** — простая статическая структура данных, предназначенная для хранения набора элементов данных, каждый из которых идентифицируется индексом или набором индексов.

- **Индекс** — обычно целое число, либо значение типа, приводимого к целому, указывающее на конкретный элемент массива.



- Количество используемых индексов массива может быть различным. Массивы с одним индексом называют *одномерными*, с двумя — *двумерными* и т. д.

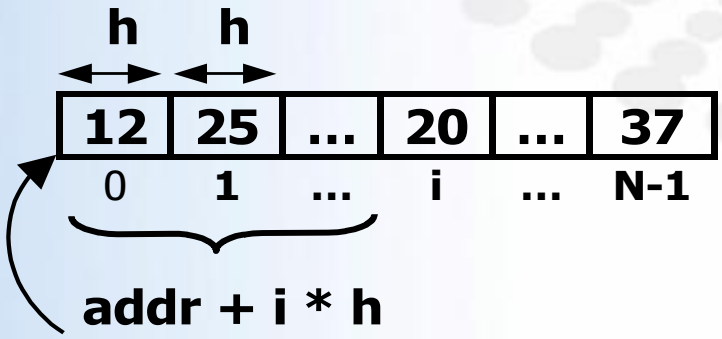
- Одномерный массив соответствует вектору в математике, двумерный — матрице.

Массивы (продолжение)

- **Специфические типы массивов**
  - **Статические массивы.**
    - Статическим называется массив, размер которого не может изменяться во время исполнения программы.
    - *В языке С встроенные массивы являются **статическими**.*
  - **Динамические массивы.**
    - Динамическим называется массив, размер которого может меняться во время исполнения программы.
    - Для реализации динамики язык программирования должен предоставлять встроенную поддержку.
    - *Язык С не имеет встроенной поддержки динамических массивов, но ее можно реализовать **функциями управления динамической памятью**.*
  - **Гетерогенные массивы.**
    - Гетерогенным называется массив, элементы которого могут содержать значения, относящиеся к различным типам данных.
    - Гетерогенные массивы удобны как универсальная структура для хранения наборов данных произвольных типов, но требуют усложнения механизма поддержки массивов в трансляторе языка.
    - *Язык С не поддерживает гетерогенных массивов (но их тоже можно эмулировать средствами языка – **объединениями**, или **смесями**).*

Массивы (продолжение)

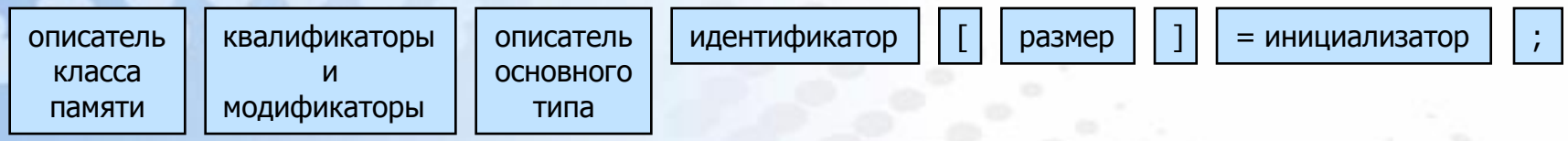
- **Достоинства массивов**
  - легкость вычисления адреса элемента по его индексу (поскольку элементы массива располагаются в памяти один за другим и имеют одинаковую длину)



  - одинаковое время доступа ко всем элементам - *набор с произвольным доступом*
  - малый размер элементов: они состоят только из информационного поля

Декларация массивов и их размещение в памяти

- Оператор описания массива состоит из следующих компонентов:



- Обязательными являются
  - ***идентификатор с квадратными скобками,***
  - ***хотя бы один из предшествующих описателей,***
  - ***точка с запятой***
- Если несколько идентификаторов имеют одинаковый набор описателей, то их можно объединить в одном операторе описания, причем каждый из них может иметь свой инициатор (в т.ч. можно объединять описания массивов и переменных).
  - Пример:

```
int i, a[5];
```

    для хранения элементов этого массива резервируется непрерывный участок памяти объемом **`5*sizeof(int)`** байт.

- Количество элементов массива в С задается *целой константой*, обычно определенной при помощи директивы препроцессора **`#define`**:

```
#define  SIZE  5
int a[SIZE];
```

Декларация массивов и их размещение в памяти (продолжение)

- Массивы можно *инициализировать при их определении*, заключая список инициаторов в фигурные скобки:

```
int a[SIZE]={2,4,6,8,10}, a0[SIZE]={0};
```

  - Если инициаторов в списке больше, чем элементов в массиве, - это *ошибка*.
  - Если инициирующих значений в списке меньше, чем элементов в массиве, то оставшиеся элементы массива получают *нулевое* значение.

- При инициализации массива можно в определении *не указывать размер* массива, тогда компилятор сам определит размер исходя из числа инициирующих значений:

```
int b[]={1,1,2,3,5,8};
```

  В этом случае массив b будет иметь 6 элементов.

  - Для дальнейшего удобно вычислить размер такого массива (*только в области видимости его определения*):

```
const int n = sizeof b / sizeof (int);
```

  или

```
const int n = sizeof b / sizeof b[0];
```

Индексация элементов массива

- Имя массива в С трактуется как *указатель-константа* на начало памяти, отведенной для хранения элементов массива.
- Доступ к отдельным элементам осуществляется *индексированием* имени, причем индексация начинается с *нуля*. Следовательно,
  - первый элемент имеет обозначение **a[0]**,
  - второй - **a[1]**,
  - последний - **a[SIZE-1]**.
- В С нет встроенных средств анализа *выхода индекса за границы массива* ни на этапе компиляции, ни на этапе выполнения программы - вся ответственность за организацию правильной работы возлагается на программиста.
  - Наиболее частая ошибка состоит в присваивании значения несуществующему элементу **a[SIZE]**, что приводит к ***порче других объектов программы***.
  - Типичный цикл для перебора всех элементов массива:

```
for(i=0; i<SIZE; ++i)
{
        … a[i] …
}
```

Символьные массивы

- Массивы, состоящие из элементов типа **char**, играют в С особую роль – они используются для представления *символьных строк*, т.к. самостоятельного типа "строка символов" (string) в С нет.
- Каждый символ строки представляется отдельным элементом символьного массива, причем непосредственно после последнего символа строки в массиве должен находиться нулевой код **'\0'**, который служит ограничителем строки.
  - Таким образом, для представления строки, состоящей из **n** символов, массив должен иметь размер не менее чем **n+1**:

```
char text[6]={'H', 'e', 'l','l','o','\0'};
```

    Массив **text** содержит строку **"Hello".**

- Поскольку инициализация символьных массивов значениями строки символов используется очень часто, то в С для нее разрешена более простая и удобная форма:

```
char text[]="Hello";
```

  - Как и ранее, размер массива определяется в этом случае компилятором по результату инициализации, причем нулевой код автоматически добавляется в массив для ограничения строки в соответствии с принятым соглашением.

Символьные массивы (продолжение)

- Для поддержки работы с символьными массивами (строками) библиотека языка С содержит большой набор функций.
- **<stdio.h>** - ввод / вывод строк
  - **int puts(const char *string);**
    - вывод строки с заменой символа **'\0'** на символ новой строки.
  - **char *gets(char *string);**
    - ввод строки с заменой символа новой строки на **'\0'**.
  - **int printf(const char *format ...);**
  - **int scanf (const char *format ...);**
    - поддерживают спецификацию **%s** для ввода / вывода строк.
  - **int sprintf(char *buffer, const char *format,...);**
  - **int sscanf(const char *buffer, const char *format,...);**
    - то же, что и **printf** / **scanf**, но обмен данными производится с символьным массивом в памяти, на который указывает **buffer**
  - и другие функции.

Символьные массивы (продолжение)

- **<stdlib.h>** - преобразование числовых значений в символьное представление и наоборот
  - **char *itoa(int value, char *string, int radix);**
  - **char *ltoa(long value, char *string, int radix);**
    - преобразование значения **value**, рассматриваемое в системе счисления с основанием **radix** (от 2 до 35), и запись его символьного представления в **string**, с символом **'\0'** в конце. Длина строки должна быть не менее 33 байт.
  - **char *gcvt(double value, int ndec, char *str);**
    - преобразует **value** в строку **str**; результат содержит **ndec** значащих цифр и представлен в формате **%f** (если возможно) или **%E**.
  - **int atoi(const char *string);**
  - **long atol(const char *string);**
  - **long strtol(const char *string, char **endptr, int radix);**
  - **double strtod(const char *string, char **endptr);**
    - функции преобразуют символьное представление числа в строке **string** в числовое значение.
  - и другие функции

Символьные массивы (продолжение)

- **<string.h>** - манипуляции со строками
  - **size_t strlen(const char *s);**
    - вычисление длины строки s.
  - **int strcmp (const char *s1, const char *s2);**
  - **int strncmp(const char *s1, const char *s2, size_t maxlen);**
    - лексикографическое сравнение строк s1 и s2 (целиком или не более **maxlen** первых символов), функции возвращают значение:
      - **< 0, если s1 < s2;**
      - **0, если s1==s2;**
      - **> 0, если s1 > s2.**
  - **char *strcpy (char *dest, const char *src);**
  - **char *strncpy(char *dest, const char *src, size_t maxlen);**
    - копирование строки **src** в строку-приемник **dest**, причем **strncpy** копирует ровно **maxlen** символов, дополняя их до **maxlen** символами **'\0'** или отсекая лишние, в последнем случае **dest** не будет ограничена.
  - **char *strcat (char *dest, const char *src);**
  - **char *strncat(char *dest, const char *src, size_t maxlen);**
    - конкатенация строк - добавление строки **src** или не более **maxlen** ее первых символов в конец строки **dest**.
  - и другие функции (поиск и замена символов и т.п.)

Многомерные массивы

- В С допускается описание не только одномерных, но также и многомерных массивов.
  - Наиболее часто используются двумерные массивы, которые можно трактовать как *матрицы*, а массивы размерности выше двух применяются гораздо реже, поскольку требуют для своего хранения значительного объема памяти.
- При описании многомерных массивов размер по каждому измерению заключается в *отдельные квадратные скобки*, например:

```
int a[3][4];
```

  - Такая конструкция рассматривается в С следующим образом: массив **a** имеет три элемента **a[0]**,**a[1]** и **a[2]**, каждый из которых в свою очередь является массивом из четырех целых чисел.
- Трактовка многомерных массивов как массивов, состоящих из других массивов, имеет следствием то, что в программе может появиться и иметь смысл любая из таких конструкций:

```
a, a[i], a[i][j].
```

  первые две из них имеют тип "массив", а третья **int**.

Многомерные массивы (продолжение)

- Как и одномерные, многомерные массивы можно *инициализировать при их определении:*

```
int a[3][4]={ {5,3,-21,42}, {44,15,0,6}, {97,6,81,2} };
```

- Внутренние фигурные скобки можно опустить, кроме того, как и ранее, разрешено опускать размер массива *по первому измерению*:

```
int a[][4]={5,3,-21,42,44,15,0,6,97,6,81,2};
```

тогда пропущенный размер определяется компилятором автоматически по длине инициирующего списка.

- Допускается инициализация лишь *части элементов* многомерного массива

```
int a[3][4]={{1,2},{3,4},{5,6}};
```

В этом примере инициализированы первые два столбца матрицы:
элементы первого столбца имеют нечетные значения,
элементы второго - четные,
а оставшиеся два столбца имеют нулевые значения.

- Если внутренние скобки опустить:

```
int a[3][4]={1,2,3,4,5,6};
```

то элементы первой строки получат значения 1,2,3,4,
первые два элемента второй строки - значения 5 и 6,
а остальные элементы матрицы будут нулевыми.

Итоги

- В этом модуле Вы изучили:
  - Правила декларации массивов и их размещение в памяти
  - Особенности реализации и работы с массивами в С
  - Роль символьных массивов в языке и функции стандартной библиотеки для работы со строками
  - Понятие многомерных массивов в С особенности работы с ними

Вопросы?

- В.Г.Тетерин – Microsoft Solution Developer (Visual C++)
- teterin@specialist.ru

ПРИЛОЖЕНИЕ

ЗАДАЧИ

Задачи

1. Содержит ли массив заданное значение X. Задачу решить методом:
   a. линейного поиска
   b. бинарного поиска (дихотомии) в отсортированном массиве
2. Найти:
   a. максимальный элемент массива
   b. два наибольших элемента массива
3. Напечатать элементы массива, превосходящие среднее арифметическое всех элементов
4. Вычислить среднее арифметическое элементов массива без учета максимального и минимального значений

Задачи

5. Проверить, упорядочен ли массив по возрастанию
6. Отсортировать массив по возрастанию методом:
   a. «пузырька»
   b. вставок
   c. поиска максимума
7. Выполнить слияние двух отсортированных массивов
8. Инвертировать массив «на месте»
9. Вычислить значение в точке X многочлена, коэффициенты которого хранятся в массиве
10. Напечатать все простые числа до N
11. Посчитать количество неповторяющихся элементов массива

Задачи

12. Ввести фамилии N студентов и их экзаменационные оценки по M предметам
    a. Вычислить средние баллы по студентам и по каждому предмету
    b. Напечатать списки «отличников», «хорошистов», «троечников» и «двоечников»
13. Написать реализацию элемента пользовательского интерфейса «EditBox» - редактируемое поле ввода
14. Написать реализацию элемента пользовательского интерфейса «PopUp Menu» - меню с выделением позиции инверсным цветом
15. Написать игру «Змейка»
16. Написать игру «Пятнадцать»