Использование массивов

Определение массива

Массив – это линейный тип данных или последовательность ячеек памяти одинакового типа. Массивы используются практически в любой программе. Массивы могут быть одномерными, двумерными и многомерными. Одномерные массивы соответствуют строке, двумерные – матрице.

Объявление массивов

При объявлении массива указывают тип его элементов и в квадратных скобках размер массива.

Для одномерного массива объявление запишется в виде:

```
int a[10];
```

Одномерные массивы

Удобно использовать при объявлении массива директиву препроцессора `#define`:

```
#define   SIZE   10   //   определили
поименованную константу, задающую
     размер массива
void main()
{
int a[SIZE];
…
}
```

Одномерные массивы

Для доступа к элементу массива указывают в квадратных скобках его номер. При этом следует отметить, что в языке С индексация элементов массива начинается с нуля.

То есть для инициализации первого элемента массива единицей необходимо записать оператор:

```
a[0]=1;
```

Одномерные массивы

Для работы с массивами обычно используют цикл `for`.

Инициализировать массив можно разными способами.

Если требуется можно инициализировать элементы массива сразу после объявления:

```
int a[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
```

Одномерные массивы

Можно заполнить элементы массива непосредственно в программе:

```
int a[10];
a[5] = 5;
for (int i=0; i<10; i++)
   a[i]=0;
```

В этом примере сначала шестому элементу массива присваивается значения пять, затем все элементы массива обнуляются.

Одномерные массивы

Инициализация одномерного массива пользователем запишется в виде:

```
int a[10];
for (int i=0; i<10; i++) {
    printf("Введите значение a[%d]\n", i);
    scanf("%d", &a[i]);
}
```

В этом примере сначала элементам массива присваиваются значения введенные пользователем.

Примеры

Рассмотрим пример нахождения количества отрицательных элементов массива.

```
int a[10], count=0;
…
for (int i=0; i<10; i++)
    if (a[i]<0) count++;
…
```

Примеры

Пример нахождения суммы отрицательных элементов массива.

```
int a[10], sum=0;
…
for (int i=0; i<10; i++)
    if (a[i]<0) sum=sum+a[i];
…
```

Примеры

Пример нахождения минимального элемента массива и его индекса.

```
int a[10], min, mini;
…
min=a[0];
for (int i=1; i<10; i++)
    if (a[i]<min) {
min=a[i];
mini=i;
}
…
```

Использование многомерных массивов

При создании двухмерного массива после его имени ставится два значения – количество строк и столбцов:

```
int a[10][12];
```

Многомерные массивы инициализируются так же, как и одномерные. В следующем примере массив sqrs инициализируется числами от 1 до 10 и их квадратами:

```
int sqrs[10][2] = {
        1, 1,
        2, 4,
        3, 9,
        4, 16,
        5, 25,
        6, 36,
        7, 49,
        8, 64,
        9, 81,
        10, 100
        };
```

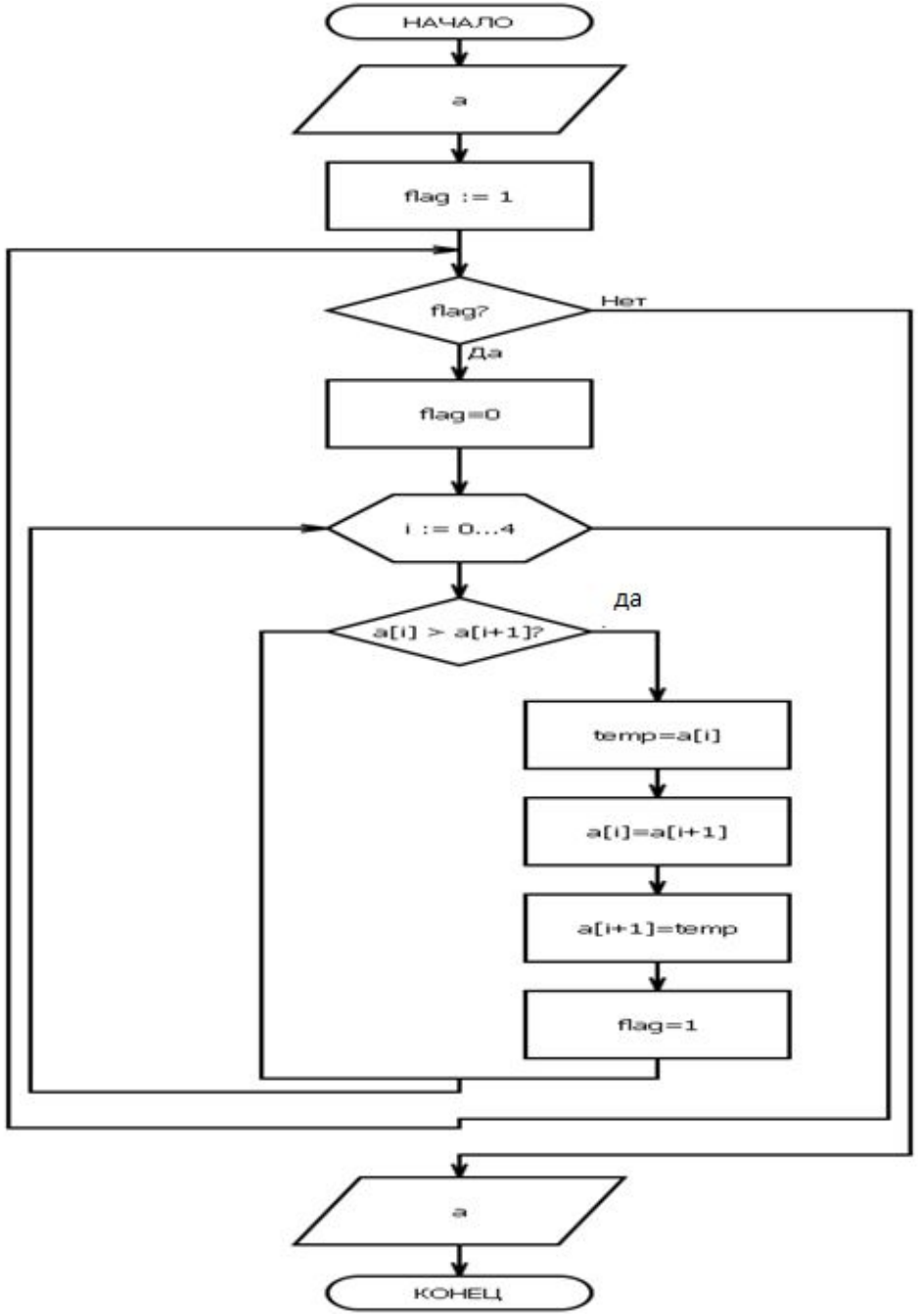
Далее на примере показаны инициализация элементов двухмерного массива.

```
int a[10][20];
for (int i=0; i<10; i++)
    for (int j=0; j<20; j++)
    a[i][j] = 0;
```

```
for (int i=0; i<10; i++)
    for (int j=0; j<20; j++)
    if(i==j)
            a[i][j] = 1;
```

```
for (i=0; i<10; i++)
{
    for (int j=0; j<20; j++)
    printf(“%i”, a[i][j])
    printf(“\n”);
            }
```

Алгоритм сортировки пузырьком



6 2 7 3 1 5

flag=1

flag=0, i=0: 2 6 7 3 1 5, flag=0

flag=0, i=1: 2 6 7 3 1 5

flag=0, i=2: 2 6 3 7 1 5, flag=1

flag=1, i=3: 2 6 3 1 7 5, flag=1

flag=1, i=4: 2 6 3 1 5 7, flag=1

flag=1

flag=0, i=0: 2 6 3 1 5 7

flag=0, i=1: 2 3 6 1 5 7, flag=1

flag=1, i=2: 2 3 1 6 5 7, flag=1

flag=1, i=3: 2 3 1 5 6 7, flag=1

flag=1, i=4: 2 3 1 5 6 7, flag=1

flag=1

flag=0, i=0: 2 3 1 5 6 7, flag=0

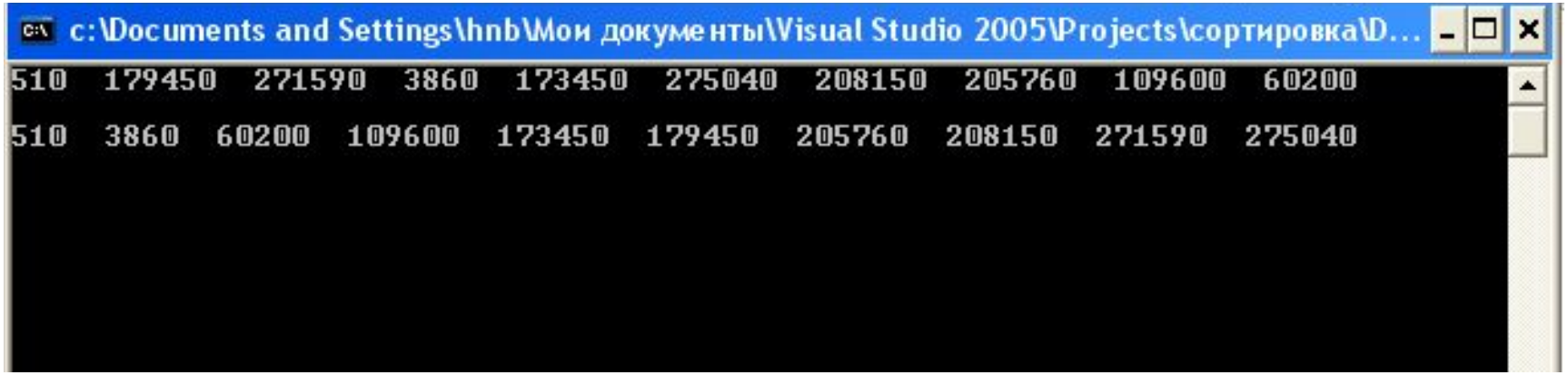
flag=0, i=1: 2 1 3 5 6 7, flag=1

Сортировка методом пузырька

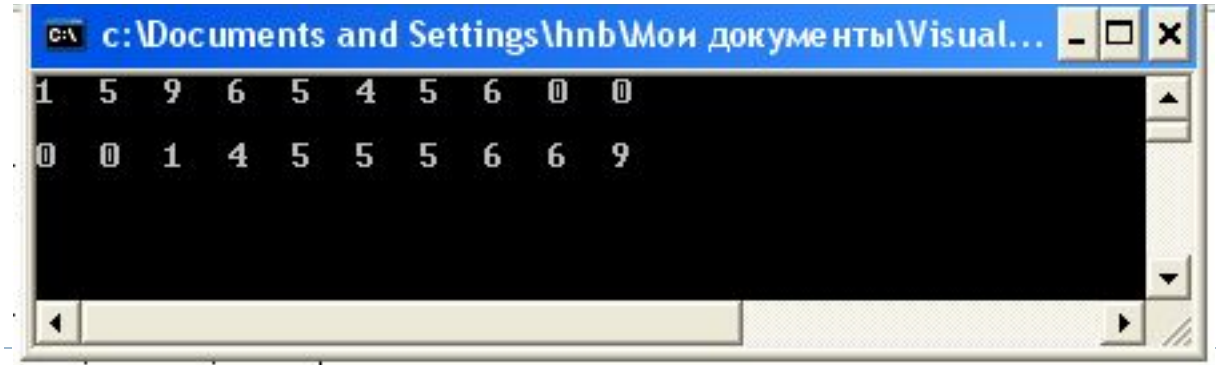
```
#include "stdafx.h"
#include "conio.h"
#include "stdlib.h"
#include "time.h"
#define randomize()
srand((unsigned)time(NULL));
#define random(p)
((int)(rand()*p/RAND_MAX))
 void main() {
 randomize();
int a[10];
char flag;
srand(4);
for (int i=0; i<10; i++)
        a[i] = rand()*10;
for (int i=0; i<10; i++)
        printf("%d ", a[i]);
printf("\n");
flag = 1;
while(flag)
{
        flag = 0;
        for (int i=0; i<9; i++)
                if(a[i]>a[i+1])
                {
                  int temp = a[i];
                  a[i] = a[i+1];
                  a[i+1] = temp;
                  flag =1;
                        }
}
        for (int i=0; i<10; i++)
                  printf("%d ", a[i]);
   getch();
}
```

Результат работы

- srand(4); for (int i=0; i<10; i++) a[i] = rand()*10;



- srand(4); for (int i=0; i<10; i++) a[i] = rand()*10;



- Функция srand() устанавливает исходное число для последовательности, генерируемой функцией rand(). (Функция rand() возвращает псевдослучайные числа.)
- Часто функция srand() используется, чтобы при различных запусках программа могла использовать различные последовательности псевдослучайных чисел, — для этого она должна задавать различные исходные числа. Кроме того, с помощью функции srand() можно многократно генерировать одну и ту же последовательность псевдослучайных чисел, — для этого нужно задавать в качестве исходного числа одно и то же значение. Иными словами, чтобы многократно генерировать одну и ту же последовательность псевдослучайных чисел, нужно вызывать данную функцию с одним и тем же значением параметра *seed* до начала генерации этой последовательности.

- Функция rand() генерирует последовательность псевдослучайных чисел.
- При каждом обращении к функции возвращается целое в интервале между нулем и значением RAND_MAX, которое в любой реализации должно быть не меньше числа 32 767.

Символьные строки

Строковые константы

Строковая константа – это последовательность символов кода ASCII, заключённая в кавычки "..." и имеющая тип char.

Примеры:

```
char str="This is character string"
char str1="Это строковая константа"
char str2="A"
char str3="1234567890"
char str4="0"
char str5="$"
```

В конце каждой строки компилятор помещает нулевой байт '\0', отмечающий конец данной строки

- Каждая строковая константа, даже если она идентична другой строковой константе, сохраняется **в отдельном месте памяти**
- Если необходимо ввести в строку символ кавычек ("), то перед ним надо поставить символ (\)
- В строку могут быть введены любые специальные символьные константы, перед которыми стоит символ \. При этом символ \ и следующий за ним символ новой строки игнорируется
- Строковые константы размещаются в **статической** памяти. Вся фраза в кавычках является указателем на место в памяти, где записана строка. Это аналогично использованию имени массива, служащего указателем на расположение массива.

```
/* Строки в качестве указателей */
main( )
{
 printf("%s, %u, %c\n", "We", "love‟,*"Pascal");
}
```

В примере, формат %s выводит строку We.

Формат %u выводит целое без знака. Если слово "love" является указателем, то выдается его значение, являющееся адресом первого символа строки.

Наконец, *"Pascal" должно выдать значение, на которое ссылается адрес, т.е. первый символ строки "Pascal".

Вот что выдаст программа:

```
We, 34, P
```

Массивы символьных строк и их инициализация

При определении массива символьных строк необходимо сообщить компилятору требуемый размер памяти

Первый способ - инициализация массива при помощи строковой константы

Например:

```
char m1[]="Только ограничьтесь одной строкой.";
```

Здесь оператор инициализировал внешний по умолчанию массив `m1` для указанной строки

```
char m1[ ]={
'Т','о','л','ь','k','о',' '
',' ,'о','г','','р','а','н','и','ч','ь','т'
,'е','с','ь',' '
',о','д','н','о','й',' '
',' ,'с','т','р','о','k','о','й','.','\0'
}
```

Без символа ‘\0’ мы имеем массив символов, а не строку. Для той и другой формы компилятор подсчитывает символы и таким образом получает размер памяти.

Как и для других массивов, имя m1 является указателем на первый элемент массива:

m1==&m1[0]

*m1=='T', и

*(m1+1)==m1[1]=='o’

* - оператор разыменования

& - оператор взятия адреса

Функции ввода-вывода строк

№	Функция	Прототип функции	Операция
1	fgets	char *fgets (s, n, stream) char *s; int n; FILE *stream;	прочитать строку из входного потока, включая символ новой строки
2	gets	char *gets (s) char *s;	прочитать строку из стандартного файла ввода stdin
3	fputs	int fputs (s, stream) char *s; FILE *stream;	записать строку в поток stream
4	puts	int puts (s) char *s;	записать строку в стандартный файл вывода stdout. В конце строк записывается символ новой строки.

Чтение и запись строк

- Функция gets() читает строку символов, введенную с клавиатуры, и записывает ее в память по адресу, на который указывает ее аргумент

```
char *gets(char *cmp);
```

Здесь *cmp* - это указатель на массив символов, в который записываются символы, вводимые пользователем, gets() также возвращает *cmp*.

- Функция puts() отображает на экране свой строковый аргумент, после чего курсор переходит на новую строку. Вот прототип этой функции:

```
int puts(const char *cmp);
```

Пример использования

- Следующая программа читает строку в массив str и выводит ее длину:

```
#include <stdio.h>
 #include <string.h>
 int main(void) {
char str[80];
gets(str);
printf("Длина в символах равна %d",
  strlen(str));
puts(str);
return 0;
}
```

Функции для обработки строк

- Для выполнения описанных в этом подразделе функций необходимо подключить файл string.h командой
- #include <string.h>

Функция	Прототип функции	Операция
strcat	char *strcat(char *s1, char *s2) char *s1, *s2;	сцепить две строки
strncat	char *strncat(s1,s2,n) char *s1, *s2; int n;	сцепить две строки, причем из второй строки копировать не более n символов
strcmp	int strcmp(s1,s2) char *s1, *s2;	сравнить две строки в лексикографическом порядке
strncmp	int strncmp(s1,s2, n) char *s1, *s2; int n;	сравнить первые n символов двух строк
strcpy	char *strcpy(s1,s2) char *s1, *s2;	копировать строку s2 в строку s1
strncpy	char *strncpy(s1,s2,n) char *s1, *s2; int n;	копировать не более n символов строки s2
strlen	int strlen(s) char *s;	определить длину строки (число символов без завершающего нулевого символа)
strchr	char *strchr(s,n) char *s; int n;	найти в строке первое вхождение символа c
strrchr	char *strrchr(s,c) char *s; int c;	найти в строке последнее вхождение символа c
strpbrk	char *strpbrk(s1,s2) char *s1, *s2;	найти в строке s1 любой из множества символов, входящих в строку s2
strspn	int strspn(s1,s2) char *s1, *s2;	определить длину отрезка строки s1, содержащего символы из множества, входящих в строку s2
strcspn	int strcspn(s1,s2) char *s1, *s2;	определить длину отрезка строки s1, не содержащего символы строки s2

Пример 1:

```
/* сцепить две строки */
#include <string.h>
#include <stdio.h>
#include <conio.h>
int main(void)
{
   char destination[25];
   char *blank = " ", *c = "C++", *turbo = "Turbo";
   strcpy(destination, turbo);
   strcat(destination, blank);
   strcat(destination, c);
   printf("%s\n", destination);
   getch();
   return 0;
}
```

```
  char destination[25];
    char blank[10], c[10], turbo[10];
    printf ("введите строку");
    gets(blank);
    gets(c);
    gets(turbo);
    strcpy(destination, turbo);
    strcat(destination, blank);
    strcat(destination, c);
    puts(destination);
    getch();
    return 0;
}
```

Пример 2:

```
/* сцепить  две строки, причем из второй строки копировать не более n символов*/
#include <string.h>
#include <stdio.h>
#include <conio.h>
int main(void)
{
char destination[25];
   char *source = "structured ";
   strcpy(destination, "programming");
   strncat(destination, source, 11);
   printf("%s\n", destination);
   getch();
   return 0;
}
```

```
int  main(void)
{
   char *buf1 = "aaabbb", *buf2 = "bbbccc", *buf3 = "ccc";
   int ptr;
   clrscr();
   ptr =;
   if (strncmp(buf2,buf1,3)> 0)
      printf("buffer 2 is greater than buffer 1\n");
   else
      printf("buffer 2 is less than buffer 1\n");
    ptr = strncmp(buf2,buf3,3);
   if (ptr > 0)
      printf("buffer 2 is greater than buffer 3\n");
   else
      printf("buffer 2 is less than buffer 3\n");
   getch();
   return(0);
}
```

```
#include <string.h>
#include <stdio.h>
#include <conio.h>
int main(void)
{
char string[20];
   char *ptr, c = 'r';
   strcpy(string, "This is a string");
   ptr = strchr(string, c);
   if (ptr)
        printf("The character %c is at position: %d\n", c, ptr);
   else
    printf("The character was not found\n");
    getch();
   return 0;
}
```