数组:一组相同数据类型变量的集合

定义:类型 数组名[大小];大小:>=1整数常量或常量表达式

内存大小:所有元素内存大小之和;连续内存

数组元素:数组名[下标];下标从0到数组大小-1

数组:静态初始化:{}

动态赋值:循环+下标

数组越界:下标超出了表示范围

排序:冒泡、插入、选择

冒泡:相邻元素两两相互比较，按照排序准则交换元素的值

for (int j = 1; j <=9; j++)//趟数

{

for (int i = 0; i <10-j; i++)//每一趟比较的次数

{

//交换两个元素的值(不满足排序准则)

if (num[i]>num[i + 1])

{

int t = num[i];

num[i] = num[i + 1];

num[i + 1] = t;

}

}

}

选择：

#include<iostream>#define N 13using namespace std;int main(){ float a[] = { 2, 5, 13, 1, 10, 6, 3, 4, 12, 8, 11, 9, 7 }; for (int i = 0; i < N - 2; i++) { int k = i;//先保存最开始的数 for (int j = i; j < N ; j++) { if (a[j] < a[k]) k = j; }

If(k!=i){

int temp = a[k]; a[k] = a[i]; a[i] = temp; }

} for (int i = 0; i < N; i++) cout << a[i] << " "; return 0;}  
插入：

具体算法描述如下：

1、将待排序序列第一个元素看做一个有序序列，把第二个元素到最后一个元素当成是未排序序列；

2、取出下一个元素，在已经排序的元素序列中从后向前扫描；

3、如果该元素（已排序）大于新元素，将该元素移到下一位置；

4、重复步骤3，直到找到已排序的元素小于或者等于新元素的位置；

5、将新元素插入到该位置后；

6、重复步骤2~5

int num[5] = {3, 7, 1, 8, 5};

int cur，length = sizeof(num)/sizeof(num[0]);

for (int i = 1; i < length; i++)

{

cur = num[i]; //待排序元素

for (int j = i - 1; j >= 0 && num[j] > cur; j--)

{

num[j + 1] = num[j];

}

num[j + 1] = cur;

}

二维数组：

定义： 类型 数组名 [维数1][维数2]

连续存储，大小为所有元素内存之和

访问方式：数组名[下标1][下标2]

静态初始化：

1 跟一维数组一样

2 把每一行看成一维数组初始化｛｝｛｝｛｝｛｝｛｝｛｝；

Int 数组名={{1,2}//给第一行初始化，｛1｝//给第二行初始化}；

给二维数组初始化的时候行数可以省略

多数组：n维数组可以看成多个一维数

字符数组：

字符串相关函数：

1 输入输出： gets puts 专门针对字符数组（字符数组名默认是以字符串的形式输出）

Gets 输入的时候可以接收间隔符

基本的5个字符串函数：

拷贝 strcpy、链接strcat、比较strcmp、测试长度strlen、字符串转化为整数atoi:

strcpy 一一对应赋值包括’\0’(以‘\0’结尾)

for(int i=0;(DstChar[i]=SrcChar[i])!=’\0’;i++);

strcat:先查找链接的位置’\0’

strcmp：int n=strcmp(数组名1，数组名2) 若有n=0 字符串相等 如果n=1 第一个大于第二个 n=-1 第一个小于第二个数组。

原理：字符串中的字符一一对应进行比较，直到出现不相同的字符。那个字符大对应的字符串大。知道‘\0’ 则两个相等

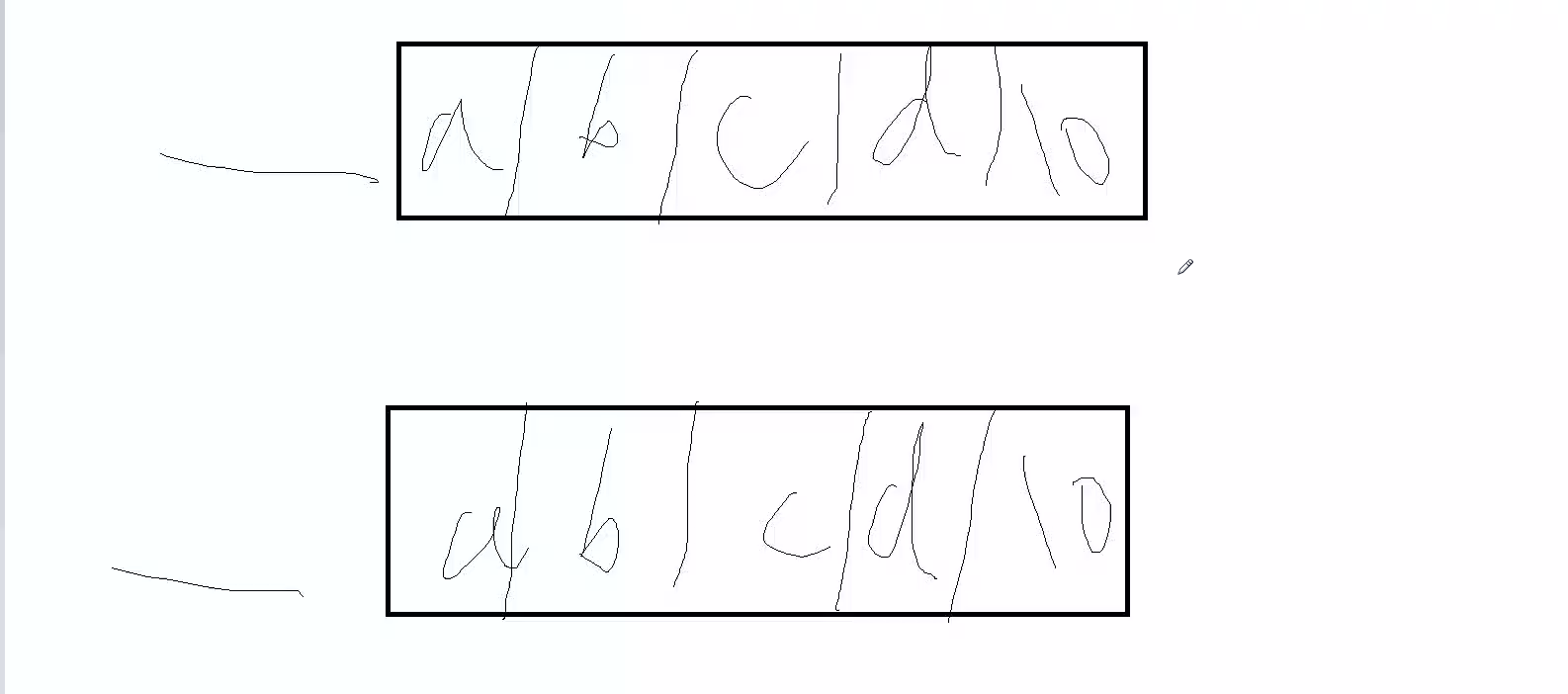
Strlen： int len = strlen(数组名或者字符串);

原理：求字符串中有多少个字符（不包括‘\0‘）

Atoi： int num=atoi(数组名或者字符串)

用字符串给字符数组初始化

1. 字符串有自己独立的内存空间
2. 把字符串中每一个字符拷贝给数组的每一个元素



上是字符数组内存 下面是字符串内存 都是一一对应的

数组的值可以修改 字符串的值不能修改