存储类别：关键字：auto static register extern

存储类别的作用：用来决定变量或函数的使用范围，生命周期，在内存中的存储位置。

**auto自动的**：仅能修饰局部变量，由auto修饰的变量，系统不给默认初始值。

被存储在内存的动态区。auto是局部变量前，系统默认加上的。

局部变量 int a; 等价于 auto int a; 二者等价

**static静态的**;既能修饰局部变量，也能修饰全局变量，还能修饰函数;

static修饰的变量，系统给默认初始值0，被存储在内存的静态区;

具有保存上一次结果值的功能。

static修饰函数，被称为内部函数。

static修饰的变量，仅在初始化一次，是在编译期间完成的。

static修饰的全局变量与全局变量的区别：功能都相同，但static修饰的全局变量他的使用范围会被限制在本文件内有效，其他文件无效。称为静态的局部变量，静态的全局变量。

**register寄存器的**：仅修饰局部变量，被存储在CPU的寄存器上。

**extern外部的**：仅能修饰全局变量或函数

extern修饰的全局变量，能起到扩展该变量的使用范围。不起定义变量的作用

extern n ：仅仅是扩展全局变量n的使用范围，不是定义n变量

extern修饰函数，被称为外部函数。

例题：局部变量

|  |
| --- |
| int fun(){int x=0;  static int y=2; //静态的局部变量，使用范围小，但生命周期特别长  y+=++x;  return x+y;}  void main(){int a,b;  a=fun();  b=fun();  printf(“%d%d”,a,b); // 输出 4 5 } |

例题：全局变量

|  |
| --- |
| int m=13; //全局变量  int fun(int x,int y){int m=2; return (x\*y-m);}  void main(){int a=7,b=6; printf(“%d”,fun(a,b)/m); // 40/13=3 (int) } |

**\*数组**：

数组的引入：当处理多个数据时，希望使用数组。

对多个数据进行统一的组织和处理，批处理。

数组是多个数据的集合，多个数据要求具有相同的数据类型。

定义时需要使用[] 一维数组 或 [][] 二维数组

1. 一维数组的定义：

语法：类型 数组名[长度];

int a[10]：定义了一个长度为10的数组，数组名为a，含义是在内存中开辟了10个int大小的房间。意味着可以存放10个整数。

定义时，长度值，必须是整型常量值，或常量表达式，不能是变量。

Error：int n=10; int a[n]; | int a(10); int a[0] ; int a[ ]

正确的：#define n 10 int a[n]; | int a[2+3]; int a[3\*2]

一维数组元素的初始化赋值：定义的同时直接给值。

初始化赋值：int a=a; | int a; a=2;

1. 给数组中所有元素赋值。int a[5]={1,2,3,4,5}，初始化赋值需要使用{}。

说明：一定要注意元素的个数不能超范围。超出为错误！

2.给数组中部分元素赋值：int a[5]={1,2,3}; 剩余元素都是0

说明：数组元素至少 给其中一个初始化赋值，剩余的元素才是0;

int a[5]; 5个值不是0,不会进行初始化。

3.给数组元素都赋值0;

int a[5]={0,0,0,0,0}; 或int a[5]={0};

int a[5]={10\*0}; //给第一个元素 赋值0

4.定义数组时，如果直接给元素初始化赋值，则数组长度可以省略。

int a[]={1,2,3}; 自动开启3个空间

一维数组元素的引用（使用）。

理解两个概念：

(1)下标值：就是元素的编号，从0开始，编号时元素相对于首元素的偏移量

int a[5]={1,3,5,7,9}; 0 1 2 3 4 下标值 13579为元素值

(2)下标法取元素：利用数组名a和下标值的组合，就是下标法：

例如：a[2]----5 a[4]----9 printf(“%d”,a[2]) ; //输出5

下标法中的下标值，可以是变量，常量，或表达式，必须是整型的

下标法引用数组元素，相当于一个普通的变量来使用。

a[0]就是一个普通变量名;

例题：

|  |
| --- |
| int sum; void main(){int a[5]={1,3,5,7,9};  for(int i=0;i<5;i++){sum=sum+a[i];}} |

说明：如果编程是针对数组，则需要利用单个元素处理法。

单个元素处理法：就是一个一个元素处理，利用循环结构，结合下标法，针对数组中的每个元素分别进行处理;

数组元素的输入：

scanf(“%d”,地址值); 数组中每个元素的地址值为：

int a[5]={1,3,5,7,9} 地址值的第一种方法：取地址 &a[0] &a[1] ….

地址值的第二种方法：用数组名，数组名表示了首元素的地址（是常量值）

a a+1 a+2 …..

例题：从键盘输入10个元素存入数组，统计和，输出。

|  |
| --- |
| void main() {int a[10]; int sum=0;  for(int i=0;i<10;i++) { scanf(“%d”,&a[i]); }  for(int i=0;i<10;i++) sum = sum+a[i]; printf(“%d”,sum); } |

总结：

1. 定义数组时长度值必须是常量，表明数组房间的个数，不能再修改了

2. 下标值，是每个元素的编号，从0开始。

3. 数组中内存房间都是连续的，并且数组名能表示首元素的地址（是常量值）

a a+1 a+2 各个房间的地址值。 a++ error 因为 a=a+1

4.下标法取元素值，表示一个普通变量a[i];

例如：

|  |
| --- |
| void main(){ int p[7]={5,6,7,8,16,17,18}; int i=0,j=0;  while(i<7 && p[i]%2 ==1) {j=j+p[i++];} printf(“%d”,j); //5} |

算法：打擂法：专门适合求最大值和最小值

方法：首先找一个人当初始擂主（数组中的第一个元素a[0]）

然后让剩余的人和擂主比较，根据需要，更新擂主。

|  |
| --- |
| void main(){ int a[5]={6,1,10,7,8}; int min,i; min=a[0];  for(i=1;i<5;i++){ if(min>a[i]) min=a[i]; } printf(“%d”,min); } |

排序算法：直接选择排序法

思想：5 8 3 1 6→1 3 5 6 8 每次都从序列中选择一个最小值,放在本趟的前方

for(int i=1;i<=n-1;i++){1 选最小的值（打擂） 2 放在本趟前方（交换）}

|  |
| --- |
| void main(){int a[5]={5,8,3,1,6}; int j,i,m;  for(i=0;i<=n-1;i++)  { m=i; for(j=i+1;j<n;j++) { if(a[m]>a[j]) m=j;}  t=a[i];a[i]=a[m] ; a[m]=t; } } |

一维数组做函数的形参; 长度可以省略

例如：

|  |
| --- |
| int fun(int a[],int n)  {int i; for(i=0;i<n;i++){if(a[i]%2==0) k++;}return k;}  void main(){ int a[5]={10,3,8,7,9}; printf(“%d”,fun(a,5)); } |

规则2：如果函数的形参是数组或指针，则实参传递给形参，传递的是实参的地址值，实参与形参共享同一内存单元，此时形参与实参互相影响。

二维数组：

定义需要使用[][];

例如：int b[3][4]; 二维数组名是b，3和4表示行长度和列长度

说明：定义二维数组时，长度值必须是整型常量，或常量表达式

这是一个3行4列组成的二维平面结构。

二维数组中的每行都是一个一维数组，这个一维数组中有4个元素。

因此，二维数组是多个一维数组的集合。

二维数组元素的初始化赋值：

(1)按元素在内存中的存储形式赋值，（存储方式采用行优先）

int b[3][4]={1,2,3,4,5,6,7,8,9,10,11,12};

说明：赋值时，要注意数组的长度，不能超范围

(2)按数组行形式赋值，每行需要使用{}

int b[3][4]={{1},{3},{5}}; 1 0 0 0 3 0 0 0 5 0 0 0

(3)定义数组时，如果直接为数组初始化赋值，则数组行的长度可以省略，但列长度一定不能省略。

int b[][4]={1,2,3,4,5,6,7,8} int b[][4]={{1},{3},{5},{7}}

二维数组元素的引用

概念：

1. 行下标：就是行的编号，从0开始，是想对于首行的偏移量

列下标：就是列的编号，从0开始，是相对于首列元素的偏移量

2. 下标法：就是用数组名和行列下标值，表示每个元素值。下标法表示元素值，相当于一个普通变量。b[0][0] b[1][1] b[0][0]++表示一个普通变量。

二维数组中的地址表示方式：

1. 行地址，每行中的地址，就是楼层地址值。

2. 列地址，每行中每个元素的地址，就是楼层中房间的地址值。

字符型数组：

例如：char st[10]; 一维的字符型数组

char st[3][10]; 二维的字符型数组

字符型一维数组的定义：char st[10];

字符型一维数组元素的初始化赋值：

(1)可以用单个字符为他赋值：

char st[5]={‘a’,’b’,’c’};

(2)可以使用字符串：

char st[5]={“abcd”}; 注意，一定不能超范围。

说明：C语言中，所有字符串都是以字符数组的形式存放的。

字符型一维数组元素的引用

(1)按单个字符形式应用：需要使用下标法：

char st[]=”abcde”;

使用单个元素处理法：利用循环，结合下标法。针对数组中的每个元素分别进行处理。

字符数组中常用的两种循环结构;

for(int i=0;i<strlen(st);i++) st[i]=st[i]-32;

for(int i=0;st[i]!=’\0’;i++) st[i]=st[i]-32;

(2)按字符串的形式处理，此时需要给出串的首元素地址值。

char st[]=”abcde”; 例如：printf(“%s”,st); strlen(st); // 5 strlen(st+2) // 3

char st[]=”abcde\0def”; strlen(st); //5 遇到\0结束

例题：

|  |
| --- |
| void main(){ char st[]=”a\*b\*c\*def”;int i,j=0; for(i=0,st[i]!='\0';i++)  { if(st[i] != '\*');{st[j]=st[i]; j++;}}st[j]='\0'; printf("%d",st) } |

删除指定字符的算法：从前后，一次扫描字符串中的每个字符，如果遇到删除字符不予处理，如果遇到非删除的字符，则向前放。 最后放\0;

二维字符数组：

char st[3][10]; 数组中包含有3个一维数组，每个一维数组都可以存放在一个字符串

例如：char st[3][10]={{“abc”},{“def”},{“123”}}; printf(“%s”,st[1]);def

说明按串形式处理，后面需要给出待处理串的首元素地址值。

字符串的处理函数;

函数，包，如果使用函数，必须要包含包

一般都在stdio.h 和 string.h 家中的函数

#include<stdio.h>

1. gets()

功能：字符串的输入函数，能输入带空格的字符串

语法：gets(地址值 或 数组名);

例如：char st[100]; gets(st); 从键盘输入的字符串，存入st数组中

scanf(“%s”,st); 此时不能输入带空格的字符串

2. puts()

功能：字符串输出函数（显示器）。

语法：puts(地址值 或 数组名);

例如：char st[]=”abc”; puts(st); //输出 abc

#include<string.h>

1. strlen()

功能：计算字符串的长度： 长度：从指定的位置开始（默认从头开始），向后数，首次遇到\0前的字符个数。

语法：strlen(地址值 或 数组名);

例如：char st[]=”abcd”; strlen(st); //输出4 strlen(st+2); // 输出2

2. strcat()

功能：两个字符串的连接;

语法：strcat(地址1 /数组名1;地址2/数组名2); 将串2的内容连接到串1尾部

例如：char s1[100]=”abc”; char s2[]=”123”; strcat(s1,s2); puts(s1);

结果为 abc123 注意！串1的空间要足够大。

连接算法：就是解决问题的方法和步骤

1. 寻找串1的尾部（\0）;

2. 将串2中的字符一个一个的拷贝到串1的尾部，直到串2遇到\0

3. 在新串的尾部加\0;