能源与动力工程学院

C语言基本知识

陈　斌

目录

指针与地址

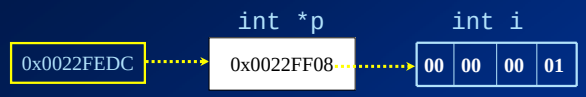
- 物理地址：计算机主存储器每个字节的实际地址。程序的指令、常量和变量等都要存放在计算机的内存中。
  - 8086CPU有20根地址线，它可以直接寻址的最大内存空间是2的20次方（1MB），0x00000到0xFFFFF。
  - 386CPU出来之后，采用了32根地址线，直接寻址的最大内存空间增加到2的32次方（4G），0x00000000到0xFFFFFFFF。
  - 64位机的寻址空间可以达到2的64次方，4G×4G
- 逻辑地址：相对于某个应用程序而言的。当执行一个编译好的可执行程序（产生一个进程）时，系统分配给该进程的入口地址可以理解为逻辑地址的起始地址，程序中用到的相关变量、数据或者代码相对于这个起始地址的位置。

指针

- 直接存取方式：无需知道变量的地址，利用变量名对内存中的内容进行存取
- 间接存取方式：将变量的地址存放在一个指针变量中，通过指针中存放的地址查找到该变量并对其进行存取操作
- 指针：存储变量的逻辑内存地址

```
int i;
int *p;
p = &i;
```

指针



- 指针的类型：
  - int * or int* ?
  - int* x, y or int *x, y
- 指针所指向的类型：int
- 指针的值或者指针所指向的内存：从指针的值所代表的那个内存地址开始，长度为sizeof(int)的一片内存区
- 指针本身所占据的内存：4个字节

指针

- 指针变量的一般形式为：

  数据类型 *标识符
- 标识符代表指针变量的名字，“*”表示该变量为指针变量
- 指针变量只能指向同类型的变量，数据类型定义了指针变量的类型及其所指向的变量的类型。但不管指向什么类型，指针里面存放的都是地址

```
double x, *p
p = &x
```

- 此时，*p 与 x 等价

指针运算

- 间接寻址或间接引用运算符 *
  - *运算表示通过指针变量存储的地址间接访问指针指向的变量
- &表示变量的地址
  - 只能应用于内存中的对象，如变量和数组元素，不能应用于表达式和常量
- *和&运算的优先级相同，结合性 从右至左

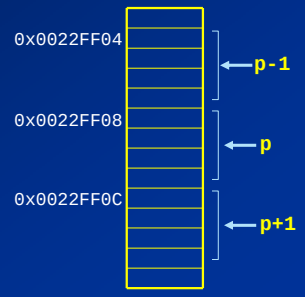
```
int i, *p;
p=&i
*&i   ?          i
&*p   ?          p
```

指针运算

- p++(p--) / p+=1(p-=1)
- 指针变量的加减不是简单的加减一个整数，而是将指针变量的地址增加或减少它所指向的变量所占用的字节数

这样可保证*(p++)指向下一个数据，而不是p本身的第二个字节。该过程由系统自动完成，所有的指针都会自动考虑它所指向对象的长度

```
int i, *p;
p=&i
(*p)++
*(p++)
```



指针变量的赋值与初始化

- 把一个变量的地址赋予指向与其数据类型相同的指针变量

```
float x, *p
p = &x;
```

- 把一个指针变量的值赋予指向相同类型变量的另一个指针变量

```
char s;
char *p1, *p2;
p1 = &s;
p2 = p1;
```

- 声明指针变量的同时初始化

```
double y, *p = &y;
```

指针变量的赋值与初始化

- 不要给未进行初始化的指针直接赋值，否则会引起错误。例如：

```
double y, *p;
*p = 1.0;
```

- NULL指针
  - 如果声明的同时没有初始化，则指针变量被初始化0。此时指针不指向任何有效数据，有时也称指针为空指针（NULL指针）。
  - 要使指针变量为NULL，可以给它赋一个0值。
  - 指针变量未赋值，说明它尚未指向任何变量或内存地址，不能使用，否则将造成意外错误。当指针变量赋为0值后，可以使用，但它没有指向任何具体的变量。

指针

例10-15 用指针变量查看内存分配

```
#include <stdio.h>
int main()
{
    int i = 1, j = -2, k = 3;
    float x = 0.123, y = -0.456;
    char s[26] = "abcdefghijklmnopqrstuvwxyz";

    unsigned char *p;

    p =(unsigned char *)(&i);

    printf("%p, %p, %p\n", &i, &j, &k);
    printf("%p, %p\n", &x, &y);

    printf("%p, %x\n", p, *p);
    printf("%p, %x\n", p+1, *(p+1));
    printf("%p, %x\n", p+2, *(p+2));
    printf("%p, %x\n", p+3, *(p+3));

    for (i = 0; i < 26; i++) printf("%p, %x\n", &s[i], s[i]);

    return 0;
}
```

指针

地址（&）	变量名	值（*）	二进制	地址（&）	变量名	值（*）	二进制
0022FF0B			00000000	0022FEFF			00111101
0022FF0A		1	00000000	0022FEFE		0.123	11111011
0022FF09			00000000	0022FEFD			11100111
0022FF08	← i		00000001	0022FEFC	← x		01101101
0022FF07			00000000	0022FEFA			00111110
0022FF06		2	00000000	0022FEFB		-0.456	11101001
0022FF05			00000000	0022FEF9			01111000
0022FF04	← j		00000010	0022FEF8	← y		11010101
0022FF03			00000000	0022FEF7	s[25]	‘z’ (122)	01111010
0022FF02		-3	00000000	0022FEF6	s[24]	‘y’(121)	01111001
0022FF01			00000000	…			
0022FF00	← k		00000011	0022FEDE	s[0]	‘a’ (97)	01100001

数组

- 数组的说明格式是:

  类型 数组名[第n维长度].....[第1维长度];

```
int a[10];        声明了一个具有10个整型数元素的数组a;
float x[10][10];  声明了一个10×10的二维单精度实数型数组x;
char s[10];       声明了一个能容纳5个字符的字符数组s;
```

- 数组以0作为第一个元素的下标
- 数组必须先定义，后使用。不能像Fortran一次引用整个数组，必须逐个引用数组元素。引用时，下标可以是整型变量或者整型表达式
- 多维数组行主存储，即最右边的下标变化最快
- C语言不允许使用动态数组

数组

- 数组元素的初始化

  类型说明符 数组名[常量表达式] = {值, 值……值};

```
float x[5] = { 0.1, 0.2, 0.3, 0.4, 0.5 };
```

- 当{ }中值的个数少于数组的元素个数时，只按顺序给前面部分元素赋值

```
int a[10]={0, 1, 2, 3, 4};
```

- 不能对数组进行整体赋值:

```
int a[10]={1, 1, 1, 1, 1, 1, 1, 1, 1, 1};
```

- 多维数组按照行主存储顺序给数组元素赋值:

```
int a[2][2]={0, 1, 2, 3};
int a[2][3]={{1, 2, 3}, {4, 5, 6}};
int a[3][3]={{1}, {0, 2}, {0, 0, 3}};
```

数组

例10-16打印10行杨辉三角形

```
#include <stdio.h>
#define N 10
int main()
{
    int i, j, a[N][N];
    for (i=0; i<N; i++)
    {
        a[i][0]=1;
        a[i][i]=1;
    }
    for (i=2; i<N; i++)
        for (j=1; j<i; j++)
            a[i][j] = a[i-1][j-1] + a[i-1][j];
    for (i=0; i<N; i++)
    {
        for (j=0; j<=i; j++)
            printf("%4d",a[i][j]);
        printf("\n");
    }
    return 0;
}
```

指针与数组

- 根据C语言的定义，数组类型的变量（即数组名）是该数组第0个元素的内存地址，因此，对于一个数组a[10]，a和&a[0]的值是一样的，都存放了a[0]这个元素的内存地址
- 定义一个与该数组同样数据类型的指针：int *p
- p = a;
- p = &a[0]; ➡ *p = a[0]
- p+i ➡ &(a+i) //指向a[i]的地址
- *(p+i) ➡ a[i]

指针与数组

值	*a	*(a+1)	*(a+2)	*(a+3)		*(a+i)		*(a+9)
	*p	*(p+1)	*(p+2)	*(p+3)		*(p+i)		*(p+9)
	a[0]	a[1]	a[2]	a[3]		a[i]		a[9]
a								
地址	&a[0]	&a[1]	&a[2]	&a[3]		&a[i]		&a[9]
	p	p+1	p+2	p+3		p+i		p+9
	a	a+1	a+2	a+3		a+i		a+9

指针与数组

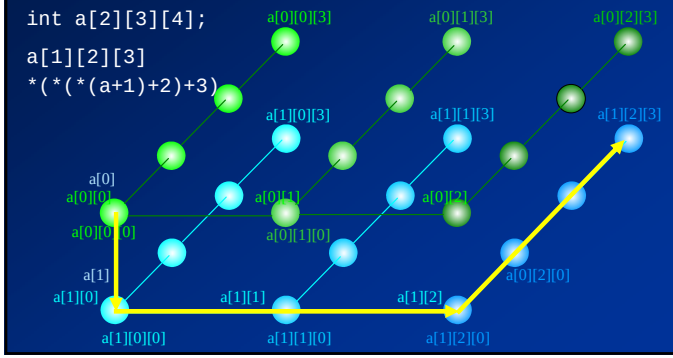
例10-17 找出数组中的最大值并记录其位置

```
#include <stdio.h>
#define N 5
int main()
{
    int a[N] = {1, 9, 4, 3, 8};
    int i, *p;

    p = a;

    for (i=1;i<N;i++)
        if (a[i] > *p) p = &a[i];

    printf("a[%d] is the maximum: %d\n", p - a, *p);

    return 0;
}
```

多维数组

- C语言中所谓的多维数组，实际是数组的数组，就是说上一维把下一维看做下一级数组，引用某个元素时需要层层嵌套

```
int a[2][3][4];
a[1][2][3]
*(*(*(a+1)+2)+3)
```



指针多维数组

例10-18指针与多维数组

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int a[5] = {1, 2, 3, 4, 5};
    int b[2][3][4] = {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24};
    int i, j, k;

    printf("memory address of a is: %p %p\n", a, &a[0]);
    printf("value at memory address %p is %d\n", a, *a);

    for (i=0;i<5;i++)
    printf("value at memory location %p (%p) is %d (%d)\n", &a[i], a+i, a[i], *(a+i));

    getch();
    printf("---------------------------------------\n");

    printf("memory address of b is: %p %p %p %p\n", b, &b[0], &b[0][0], &b[0][0][0]);

    for (i=0;i<2;i++)
        for (j=0;j<3;j++)
            for (k=0;k<4;k++)
                printf("value at memory location %p is %d (%d)\n",
                       &b[i][j][k], b[i][j][k], *(*(*(b+i)+j)+k));

    return 0;
}
```

内存动态分配

- C语言不支持动态数组，数组的长度必须预先加以定义，在整个程序执行过程中不能动态改变。
- 对于所需的内存空间预先未知的情形，可以通过C语言提供的内存管理函数按实际需要动态地分配内存空间，实质是利用指针变量向操作系统申请一块新的内存，使用完毕后再回收待用，从而有效地利用内存资源。
- 内存管理函数包括内存的分配和释放，内存的分配可以使用malloc()、calloc()、realloc()等函数，释放则使用free函数。使用mallac函数和free函数都需要包含头文件stdlib.h。

内存动态分配

- 根据不同的系统状况，申请内存可能成功也可能失败，失败时返回空指针NULL。因此，动态申请内存时，一定要判断结果是否为空；
- malloc只管分配内存，并不能对得到的内存进行初始化，新内存的值是随机的，要正确使用该内存必须进行初始化；
- malloc()的返回值类型是“void *”，使用时不要忘记类型转换；
- 内存被free()函数释放后，并不表示指针会消亡或者成了NULL 指针。因此，释放内存后最好将指针置空。

内存动态分配

```
int *p1 = (int *) malloc ( 100 * sizeof(int) );
/* 分配用于存放100个整数的内存空间*/

float *p2 = (float *) malloc (50 * sizeof(float) );
/* 分配用于存放50个单精度浮点数的内存空间*/

char *p3 = (char *) malloc (10 * sizeof(char) );
/* 分配用于存放10个字符的内存空间*/


free(p);          /* 释放以p为首地址的空间 */
p = NULL;         /* 将p置空，防止野指针 */
```

例10-22 输入任意个整数后排序输出

能源与动力工程学院

Thank You !