20160308学习笔记

指针

1.地址和指针的概念

- 按变量地址存取变量值的方式称为“直接访问”方式
- 将变量i的地址存放在另一个变量中称为“间接访问”方式

2.变量的指针和指向变量的指针变量

基类型 *指针变量名
&取地址 引用
*取值 解引用

```
    int i = 3;
    int  *i_pointer;
    i_pointer = &i;
    printf (“the i value is %d\n”,*i_pointer) ;
    /*
            在内存i的地址0x31F934
            i_pointer的地址0x31F928,值是i的地址
            打印*i_pointer为3
    */
```

一个变量的地址称为该变量的“**指针**”。
一个专门用来存放另一个变量的地址的变量称为“**指针变量**”
(*i_pointer)++ 等价于 a++

```
void func(int *a,int *b){
    int tmp;
    tmp = *a;
    *b = *a;
    *b = tmp;
}
//指针传递的不是值而是地址值，从而交换
void main(){
    int i=3,j=4;
    int *p1 = &i;
    int *p2 = &j;
    func(*p1,*p2);
}
```

3.数组与指针

```c
void cal_new(int *p,int size){
    int i;
    for(i=0;i<size;i++){
        *(p+i) =  *(p+i) *10;
    }
}

void main(){
    int a[10] = {1,2,3,4,5,6,7,8,9,10};
    int *p;
    //p = a;//一般用这种
    //p = &a[0];*p的值为1  *(p+1)=*(a+1)的值为2 偏移sizeif(int)
    cal_new(a,10);
    for(i=0,i<10;i++){
        printf("%4d",*(a+i));
    }
    printf("\n");
}
```

```c
void main(){
    char *p = "hello world";
    *p = 'H';//错误,字符串存在常量区
}
```

```c
void main(){
    char a[20] = "hello world";
    char *p;
    p = c;
    *p = 'H';//正确
}
```

```c
#include <string.h>
#define OUT
void find_word(char *start,OUT int *pos){
    int i;
    for(i=0;i<strlen(start);i++){
        if(start[i]==' ' && start[i+1] >= 'a' && start[i+1]<='z'){
            *pos = i + 1;
            return;
        }
    }
}
void main(){
    char a[20] = "hello world";
    char *p;
    int i;
    p = c;
    find_word(p,&i);
    printf("the i is %d\n",i);
    printf("the word is %s\n",&p[i]);
}
```

4.动态分配

[**malloc&free**] (http://www.cnblogs.com/hanyonglu/archive/2011/04/28/2031271.html)
void* malloc(size_t size);
动态分配的空间存放在堆
(char*)malloc(8)
在整个进程执行时都不释放
malloc之后要free

```c
#include <stdlib.h>
void main(){
    char *p;
    p = (char *)malloc(10);
    strcpy(p,"hello");//不能直接赋值
    printf("%c",*p);//h
    *p = 'H';
    printf("%c",*p);//H
    free(p);//free以后要把指针置为NULL
    p = NULL;
}
```

```c
#include  <string.h>
char *find_word(char *s){
    //char a[20];//不能拿局部变量当返回值,用堆
    char *a;
    int i;
    a = (char *)malloc(10);
    for(i=0;i<strlen(s);i++){
        if(s[i]==' '&& s[i]>='a' && s[i]<='z'){
                strcpy(a,s+i+1);
                return a;
        }
    }
    free(a);
    a=NULL;
}
void main(){
    char str[20];
    char *p;
    while(memset(str,0,sizeof(str)),gets(str)!=NULL){
        p = find_word(str);
    }
}
```

5.函数指针
函数在编译时被分配给一个入口地址。这个函数的入口地址就称为函数的指针

```c
void main(){
    int max1(int,int);
    int (*p)();//也可以写上int,int
    int a,b;
    a = 5;
    b = 10;
    p = max1;
   printf( "the max is %d\n",(*p)(a,b));//10
}
int max1(int a, int b){
    return a>b?a:b;
}
```

应用qsort

```c
#include <stdio.h>
#include <stdlib.h>

int compare(const void* left,const void *right){
    //升序
    int *p1 = (int*)left;
    int *p2 = (int*)right;
    if(*p1<*p2){
        return -1;
    }else if(*p1==*p2){
        return 0;
    }else{
        return 1;
    }
}
void main(){
    int a[10] = {4,78,20,10,5,-15,100,99,17,29};
    int (*p)(const void*,const void*);
    int i;
    p = compare;
    qsort(a,10,sizeof(int),p);
    for(i=0;i<10;i++){
        printf("%4d",a[i]);
    }
    printf("\n");
    system("pause");
}
```

6.二级指针

int i = 5;

int j = 10;

int *p1;

p1 = &i;

若想让指针p指向j，则用到二级指针

传递：要在子函数里修改主函数里的一级指针变量的值，用二级指针

```
void change(int **p,int *p2){
    *p = p2;
}
void main(){
    int i=3,j=4;
    int *p1 = &i;
    int *p2 = &j;
    change(&p1,p2);
    printf("*p1 is %d\n",*p1);
    system("pause");
}
```

7.多维数组与指针

```
void main(){
    int a[3][4] = {1,3,5,7,9,11,13,15,17,19,21,23};
    //sizeof(a[1]) = 16
    system("pause");
}
```

&a + 1 偏移到a的结尾

指针数组

char *p[10];//存放的全是指针的数组*

数组指针

int (p)[4];//指向数组的指针