

MA 511: Computer Programming

Lecture 10: Pointers (Contd..)

http://www.iitg.ernet.in/psm/indexing_ma511/y10/index.html

Partha Sarathi Mandal

psm@iitg.ernet.ac.in

Dept. of Mathematics, IIT Guwahati

Semester 1, 2010-11

Passing Arguments to Function

- Passing **values** (call by value)
- Passing **pointers** (call by reference)
- Passing a **pointer to an array**
- Passing a **pointer to a character string**
- Passing a **pointer to a character**

Example: A function can return a pointer

```
int *func(int *p);

main(){
    int a[5]={10,20,30,40,50};
    int *maxptr;
    maxptr = func(a);
    printf("max=%d", *maxptr);
}
```

```
int *func(int *p){
    int i, imax, max = 0;
    for(i=0; i<5; i++){
        if(*(p+i) > max){
            max =*(p+i);
            imax = i;
        }
    }
    return(p+imax);
}
```

Example: Passing function to other function

```
float guest1(float x, float y);
float guest2(float x, float y);
float host(float (*pt) (float x, float y));
```

```
main(){
    float x, y;
    x = host(guest1);
    printf("x = %f\n",x);
    y = host(guest2);
    printf("y = %f\n",y);
} /* end of main */
```

```
float guest1(float x, float y){
    return(x+y);
}
float guest2(float x, float y){
    return(x*y);
}
float host(float (*pt) (float x, float y)){
    float a=5.0, b=3.5, c;
    c=(*pt)(a,b);
    return(c);
}
```

Dynamic memory allocation

```
int x[10];
```

Or

```
int *x;
```

```
x = (int *) malloc(10*sizeof(int));
```

- **malloc** library function reserves a block of memory size is equivalent to 10 integer for the above example.
- stdlib.h or **malloc.h** header file is required.
- **malloc** return a pointer to the beginning of the allocated space.
- in general **malloc(a)** allocate **a** bytes of memory.

Example: Dynamic memory in 1D

```
#include<stdio.h>
#include<stdlib.h>
main(){
    int *a, i, n;
    printf("Type n");
    scanf("%d",&n);
    a = (int *)malloc(n*sizeof(int));

    for(i=0; i<n; i++){
        printf("type %d th entry",i);
        scanf("%d", a+i);
    }
    for(i=0; i<n; i++){
        printf("%d", *(a+i));
    }
}
```

Example: Dynamic memory in 2D

```
#include<stdio.h>
#include<stdlab.h>
#define COL 50
void read_ip(int *a[COL], int nrow, int ncol);
void write_op(int *a[COL], int nrow, int ncol);
main(){
    int *a[COL], nrow, row;
    Printf("Type no of rows & cols");
    scanf("%d%d",&nrow, &ncol);
    for(row=0; row<=nrow; row++){
        a[row]=(int *) malloc(ncol*sizeof(int));
    }
    read_ip(a, nrow, ncol);
    write_op(a, nrow, ncol);
}
```

```
void read_ip(int *a[ROW], int m, int n){
    int row, col;
    for(row =0; row<m; row++)
        for(col =0; col<n; col++)
            scanf("%d", (a[row]+col));
}
void write_op(int *a[ROW], int m, int n){
    int row, col;
    for(row =0; row<m; row++){
        for(col =0; col<n; col++)
            printf("%4d", *(a[row]+col));
        printf("\n");
    }
}
```

Structures and pointers

```
typedef struct {  
    int    acct_no;  
    char   acct_type;  
    char   name[80];  
    float  balance;  
} account;  
account customer, *ptr;  
ptr = &customer
```

```
struct {  
    int    acct_no;  
    char   acct_type;  
    char   name[80];  
    float  balance;  
} customer, *ptr;  
ptr = &customer
```

Structures and pointers

```
typedef struct {  
    int      month;  
    int      day;  
    int      year;  
} date;
```

```
typedef struct {  
    int      acct_no;  
    char    acct_type;  
    char    name[80];  
    float   balance;  
    date    lastpayment;  
} customer, *ptr = &customer;
```

Example: Structures and pointers

```
main(){

    typedef struct {
        int month;
        int day;
        int year;
    } date;

    struct {
        int acct_no;
        char *acct_type;
        char *name;
        float balance;
        date lastpayment;
    } customer, *ptr = &customer;
```

```
ptr->acct_no = 12341;           //customer.aact_no = 12341;
ptr->acct_type = "S/B";         //customer.acct_type ="S/B";
ptr->name = "xyz";             //customer.name = "xyz";
ptr->balance = 2345.00;         //...
ptr->lastpayment.month=02;
ptr->lastpayment.day=15;
ptr->lastpayment.year=2009;

printf("Account= %s Name=%s\n", ptr->acct_type, ptr->name);
printf("AccNo=%d, Balance=%f, Year=%d\n",
       ptr->acct_no, ptr->balance, ptr->lastpayment.year );
}
```

Account= S/B Name=xyz
AccNo=12341, Balance=2345.000000

Example: Structures and pointers

```
main(){

    typedef struct {
        int month;
        int day;
        int year;
    } date;

    typedef struct {
        int acct_no;
        char *acct_type;
        char *name;
        float balance;
        date lastpayment;
    } account;

    account customer, *ptr = &customer;

    ptr->acct_no = 12341;           //customer.aact_no = 12341;
    ptr->acct_type = "S/B";        //customer.acct_type ="S/B";
    ptr->name = "xyz";            //customer.name = "xyz";
    ptr->balance = 2345.00;        //...
    ptr->lastpayment.month=02;
    ptr->lastpayment.day=15;
    ptr->lastpayment.year=2009;

    printf("Account= %s Name=%s\n", ptr->acct_type, ptr->name);
    printf("AccNo=%d, Balance=%f, Year=%d\n",
           ptr->acct_no, ptr->balance, ptr->lastpayment.year );

}
```

Account= S/B Name=xyz
AccNo=12341, Balance=2345.000000

Passing structures to functions

```
typedef struct {  
    int    acct_no;  
    char   *acct_type;  
    char   *name;  
    float  balance;  
} account;  
  
void get_data(account *p){  
    p->acct_no = 55555;  
    p->acct_type = "C/A";  
    p->name = "abc";  
    p->balance = 777777.00  
}
```

```
main(){  
    void get_data(account *p);  
  
    account customer, *ptr;  
    ptr = &customer;  
  
    ptr->acct_no = 12341; //customer.aact_no = 12341;  
    ptr->acct_type = "S/B"; //customer.acct_type ="S/B";  
    ptr->name = "xyz"; //customer.name = "xyz";  
    ptr->balance = 2345.00; //...  
  
    printf("Account= %s Name=%s\n", ptr->acct_type, ptr->name);  
    printf("AccNo=%d, Balance=%f\n", ptr->acct_no, ptr->balance);  
  
    get_data(ptr); //get_data(&customer);  
  
    printf("Account= %s Name=%s\n", ptr->acct_type, ptr->name);  
    printf("AccNo=%d, Balance=%f\n", ptr->acct_no, ptr->balance);  
}  
  
Account= S/B Name=xyz  
AccNo=12341, Balance=2345.000000  
Account= C/A Name=abc  
AccNo=55555, Balance=777777.000000
```

Passing structures to functions

```
# define N 2
typedef struct {
    int acct_no;
    char *acct_type;
    char *name;
    float balance;
} account;

account *get_data(account *p){
    p->acct_no = 55555;
    p->acct_type = "C/A";
    p->name = "abc";
    p->balance = 777777.00;
    return(p)
}
```

```
main(){
    account customer[N], *ptr;
    ptr = customer;
    int i;
    for(i=0; i<N; i++){
        ptr=get_data(ptr+i); //get_data(customer+i);

        printf("Account= %s Name=%s\n", ptr->acct_type,
               ptr->name);
        printf("AccNo=%d, Balance=%f\n", ptr->acct_no,
               ptr->balance);
    }
}
```

```
Account= C/A Name=abc
AccNo=55555, Balance=777777.000000
Account= C/A Name=abc
AccNo=55555, Balance=777777.000000
```