# Review of pointers in functions

A pointer is a data item whose value is the address in the memory of some other value.
Pointers allow us to refer to a large data structure in a compact way. No matter ***how large*** the data structure grows, it will reside somewhere in the computer’s memory and therefore would have an address.

We can use the address as a short hand for the complete value. All we need to store is an integer, irrespective of the size of the structure. So we may talk about a complete list with 5 nodes or 100 nodes with just one address.

Pointers facilitate sharing data between different parts of the program. When we pass the address of some data value from one function to another, both functions have access to the same data.

Pointers make it possible to reserve new memory during program execution (Dynamic allocation).

We have already seen that pointers can also be used to record relationship among data items (such as nodes in a linked list).
Suppose we want to point to an address where we have kept 45.

```
*p=45;
```

means that the object pointed to by p is assigned the value 45.

Now consider

```
q = &y;
```

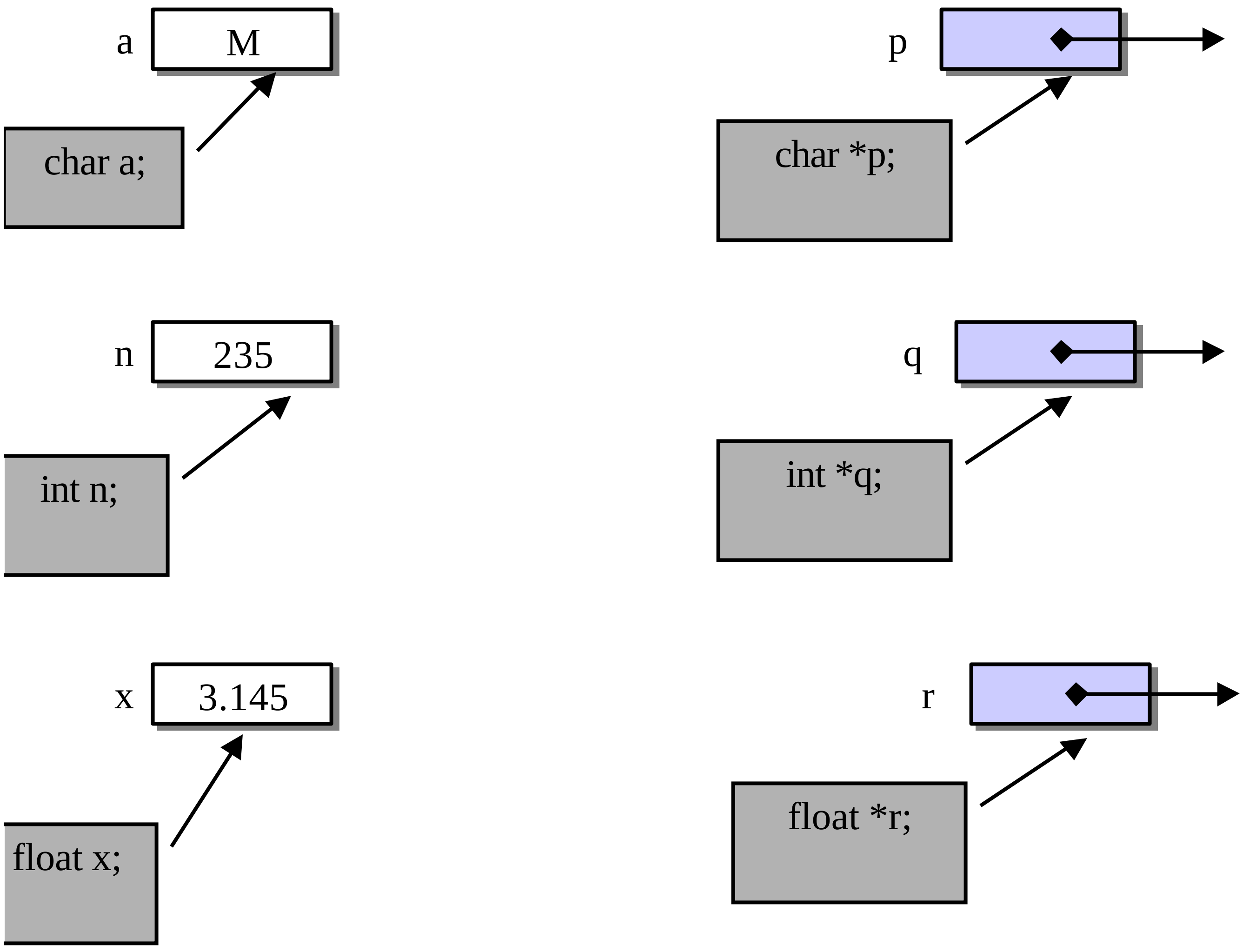
Here q is assigned the address of object y. So q points to object y.

In the following statement r is assigned the value of the object pointed to by q.

```
r= *q;
```

Basically this means that r and q are equivalent and referring to the same object.

The addresses of variables can be used as arguments to functions ( call-by- reference ).
Instead of ordinary variables , now we can use pointers to form the parameter list in the functions definition.


a
M
char a;
p
char *p;
n
235
int n;
q
int *q;
x
3.145
float x;
r
float *r;

**A good example is to carry out  swapping of values of two variables  through a function call.**

```
int main()
{
   int x=45,y=92;
   swap(&x, &y);
   return 0;
}
```

/* note that the main  program passes on the addresses of the variables x and y*/

```
void swap( int *p, int *q)
 {
     int temp;
     temp=*p;
     *p=*q;
     *q=temp;
}
```

The function  swap is receiving the objects pointed by the addresses  p and q. Here instead of the variables we are passing the pointers to the two variables.

The called function has full control on those variables and can manipulate them, so that when main function looks at them after function call, it would find them changed.

Function returning  pointers

```
int *smaller (int  *num1, int
*num2);
int main ( )
{ int a;
   int b;
   int *p;
   scanf (“%d %d”, &a, &b);
   p = smaller (&a, &b);
}
```

The main program passes addresses of  integers a and   b.
The function  returns a pointer to address holding the smaller of the two elements.

```
int *smaller (int *px, int *py)
{
     return (*px < *py  ? px : py);
}
```

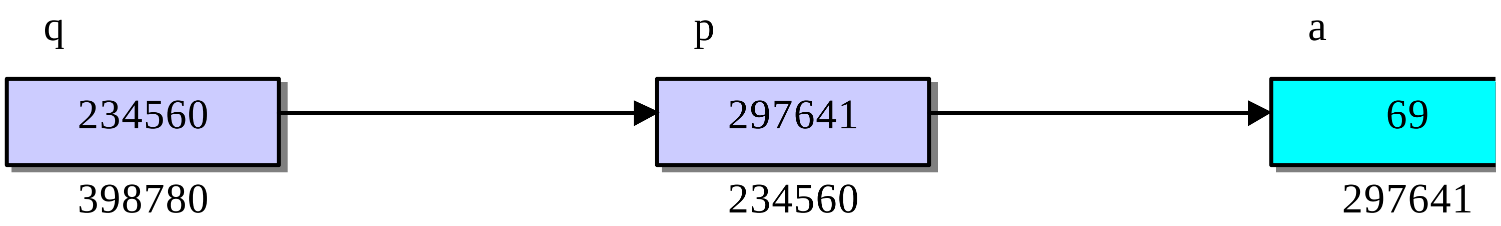
DOUBLE POINTERS
(pointer to a pointer )

```
int a;
int *p;
int **q;                  // pointer to p

a = 69;
p = &a;
q = &p;

printf(“ %3d %3d %3d”, a, *p, **q);

 // here *q is de-referenced as  p
```



## **Useful Programming tips**

When several values need to be returned to the calling function, use address parameters for all of them.

Do not return one value and use address parameters for the others.

Use the return for some other purpose, such as a status flag, or make the return void.

Create local variables when a value parameter will be changed within a function so that the original value will always be available for processing.

## *Florida Lottery System*

### The Problem

The system  automatically creates a file that contains the name of each ticket purchaser along with the combination of 6 ***distinct*** numbers picked by that person from the set {1,2,3,...,52,53}.

The payout for matching a certain number of values:

| Numbers Matched | Winnings |
|---|---|
| 3 | $10 |
| 4 | $100 |
| 5 | $10000 |
| 6 | $1000000 |

The program will ask the user for an input file with the data for the ticket purchases.

Then it will ask the user for the winning combination of numbers.

The user must enter these numbers in ascending order.

## Input File Format

The first line will contain **n**, the total number of tickets bought.

The next 2n lines will contain information about each ticket bought.
The information for a single ticket will be on two lines.

The first of the two lines will contain the last name of the ticket buyer, followed by the first name.

The following line will contain the six integers chosen by that buyer

Here is a sample file, input.txt:

```
5
Dorr Robin
1 15 19 26 33 46
Mong Drian
17 19 33 34 46 47
Pate David
1 4 9 16 25 36
Bayer Lisa
17 19 34 46 47 48
Gupta Alok
5 10 17 19 34 47
```

```
#include<stdio.h>

typedef struct
{
   char first[20];
   char last[20];
   int nos[6];
}gambler;

gambler *read_file(char file[],   int *numbuyers);
void print_winners( gambler *list,  nt numtickets );
void  match_one (gambler buyer ,  int win_nos[ ] );


int main()
{
   char filename[20];
   int numbuyers,win_nos[6];
   gambler *ticket_buyers;

    // reads the filename containing buyer information.
  printf("\nEnter the name of the file with the ticket
data \n");
  scanf("%s",filename);




    // reads file information into memory and prints the
winning numbers.
```

```
   ticket_buyers=read_file(filename,
&numbuyers);

   print_winners(ticket_buyers,
numbuyers);

  // free memory space used.
   free (ticket_buyers);
   return 0;
}

//Preconditions: file should be name of file storing the
ticket buyer
//information in proper    format.
//Postconditions: Returns a Pointer to the array storing all
ticket buyer
//information and total number of ticket buyers.

gambler *read_file (char file[],   int
*numbuyers)
{
   FILE *fp;
   int   i, j;

   gambler  *list  ;

   fp =  fopen( file,   "r");
   fscanf(   fp,  "%d",    numbuyers);

   list  =  (gambler* )  malloc( (  *numbuyers)
*sizeof(gambler));
```

```
     // read in individual ticket buyer information

   for (i=0;   i<*numbuyers;    i++)
           {
          //read in first and last name
        fscanf(  fp,   "%s"   ,  list[i].last );
        fscanf(  fp,   "%s",     list[i].first);


          //read in his choice of winning numbers

        for( j  = 0; j<6; j++)
        fscanf( fp,   "%d",    &( list[i].nos[j] ));
     }

    fclose(fp);
    return list;

}
```

**// FUNCTION LOOKS AT LIST AND GETS WINNING NUMBERS FROM USER//**

//Precondition: list points to a valid list of gamblers of length *numtickets*.

**//Postcondition: Will scan the complete file , read in the winning numbers and call the function to match with individual ticket information.**

```
void print_winners( gambler *list, int numtickets )
{
  int i,  k , win_nos[6];

    //get the winning lottery numbers from the user

  printf("\nEnter the winning lottery nos :\n ");
  for( i=0 ;i<6 ;i++)
  scanf("%d",&win_nos[i]);

    //for each ticket buyer , match the numbers with the
winning numbers.
  for  ( k = 0;    k< numtickets;  k++)
  match_one( list [k ] ,  win_nos  );
}

// FUNCTION TO MATCH DATA
//Precondition: buyer is a valid struct.
//Postcondition: If  3 or more numbers match , then print
ticket buyer's name with winning amount.

void  match_one (gambler buyer , int win_nos[] )
{
  int i=0 ,    j=0,    count=0;

     // Instead of matching all the winning numbers with
individual  number read from the file,we check only upto
next higher number When numbers match we look for
another match by moving down the list.

  while(i<6 && j<6)
```

```
        {
           if( win_nos[i] <  buyer.nos[j] )
                   i++;
           else  if (win_nos[i]  >  buyer.nos[j] )
                   j++;
           else
                 {
                     count++;
                     i++;
                     j++;
                 }
        }

//Depending on number matched print out the result
  switch(count)
      {
case 3:
        printf( "\n%s %s   matched  3 numbers and won
$10.\n",
        buyer.first, buyer.last);
        break;

case 4:
        printf( "\n%s %s   matched  4 numbers and won
$100.\n",
        buyer.first, buyer.last);
        break;
case 5:
        printf( "\n%s %s   matched  5 numbers and won
$10000.\n",
        buyer.first, buyer.last);
        break;

case 6:
        printf( "\n%s %s   matched  6 numbers and won 
$1000000.\n",
        buyer.first, buyer.last);
        break;
```