C Programming Introduction

week 12: Arrays and Pointers

Pointers and Arrays

- Recall that an array **s** holds the address of its first element **s[0]**
- **s** is actually a pointer to **s[0]**

```
int s[10];
int *iptr;
iptr=s; /* From now iptr is equivalent to s  */
```

- Both **iptr** and **s** now point to **s[0]**

Pointer-array equivalence

- Arrays are actually a kind of pointers!
- When an array is defined, a fixed amount of memory (the size of the array) is allocated.
  - The array variable is set to point to the beginning of that memory segment
- When a pointer is declared, it is uninitialized (like a regular variable)
- Unlike pointers, the value of an array variable cannot be changed

Pointer arithmetic

- Pointers can be incremented and decremented
- If **p** is a pointer to a particular type, **p+1** yields the correct address of the next variable of the same type
- **p++**, **p+i**, and **p += i** also make sense

Pointer arithmetic

- If **p** and **q** point to elements in an array, **q-p** yields the number of elements between **p** and **q**.
- However, there is a difference between pointer arithmetic and "regular" arithmetic.

Pointer arithmetic - example

```
int main(void)
{
    int a[3] = {17,289,4913}, *p, *q;

    p = a;   /* p points to the beginning of a, that is &a[0] */
    q = p+2; /* q points to a[2]. Equivalent to q = &a[2]     */

    printf("a is %p\n", a);
    printf("p is %p, q is %p\n", p, q);
    printf("p points to %d and q points to %d\n", *p, *q);
    printf("The pointer distance between p and q is %d\n", q-p);
    printf("The integer distance between p and q is %d\n",
        (int)q-(int)p);
    return 0;
}
```

```
a is 0012FECC
p is 0012FECC, q is 0012FED4
p points to 17 and q points to 4913
The pointer distance between p and q is 2
The integer distance between p and q is 8
```

Passing arrays to function

- Another way to pass arrays to function is using pointer
- In fact, we pass just the array's address, or more precisely a pointer to the array.
- The function calculate the sum of all array elements.

```
#include <stdio.h>
int addNumbers(int *fiveNumber){
   int i,sum=0;
   for(i=0; i<5; i++, fiveNumbers++){
       sum+= *fiveNumbers
   }
   return sum;
}
```

Exercise 12.1

- Write a function countEven(int*, int) which receives an integer array and its size, and returns the number of even numbers in the array.

Solution

```
int counteven(int* arr, int size){
  int i;
  int count =0;
  for (i=0; i<size; i++)
     if (*(arr+i)%2==0) count++;
  return count;
}
```

Exercise 12.2

- Write a function that returns a pointer to the maximum value of an array of double's. If the array is empty, return NULL.

  double* maximum(double* a, int size);

Solution

```
double* maximum(double* a, int size){
  double *max;
  double *p;
  int i;
  max=a;
  if a==NULL return NULL;
  for(p=a+1; p<a+size; p++)
     if (*p > *max){
           max = p;
     }
  return max;
}
```

Exercise 12.3

Write a function getSale uses a pointer to accept the address of an array. It asks the user to enter the sales figures and stores those figures in the array.

Write a function totalSale return the total of the element int the array.

Use these two functions in a program to input the sales figure from different quarteurs and display the total. Using pointers instead of array in function's parameters.

Solution

```
#include <stdio.h>
void getSales(float *array, int size){
   int i;
   for(i=0; i<size; i++){
   printf("Enter the sale figure for quarter %d:",
   i+1);
   scanf("%f",array+i);
   }
}

float totalSales(float *array, int size) {
   double sum;
   int i; sum =0;
   for(i=0; i<size; i++){
   sum +=*array;
   array++;
   }
   return sum;
}
```

Solution

```
int main()
{
  float sales[6];
  getSales(sales,6);
  printf("The total sales for the
  year are:
  %0.1f\n",totalSales(sales,6));
  return 0;
}
```

Exercise 12.4

- Write a program to list all the sub array of an given array. For example the array 1 3 4 2 has the following sub array:

```
1
1 3
1 3 4
1 3 4 2
3
3 4
3 4 2
4
4 2
2
```

Solution

```
#include<stdio.h>

void main()
{
   int a[100],n;
   printf("n = "); scanf("%d",&n);
   for(int i=0;i<n;i++)
   {
   printf("\na[%d] = ",i);scanf("%d",&a[i]);
   }
   for(i=0;i<n-1;i++)
   {
   printf("\n%d",a[i]);
   for(int j=i;j<n-1;j++)
   {
   printf("\n");
   for(int k=i;k<=j+1;k++)
   printf("%d\t",a[k]);
   }
   }
}
```

Exercise 12.5

- Write a program to reverse an array in two different ways: using indexes and using pointers.

Solution: array

```
void reversearray(int arr[], int size){
   int i, j, tmp;
   i=0; j= size -1;
   while(i<j){
       tmp=a[i];
       a[i]=a[j];
       a[j]= tmp;
       i++; j--;
   }
}
```

Solution: pointer

```
void reversearray(int *arr, int size){
   int i, j, tmp;
   i=0; j= size -1;
   while(i<j){
      tmp=*(a+i);
      *(a+i)=*(a+j);
      *(a+j)= tmp;
      i++; j--;
   }
}
```