Computer Science 1510

Lecture 25

Lecture Outline

- Arrays

Arrays

- Like in Fortran, arrays in C are consecutive locations in memory where data of the same type can be stored.

- However, there are some notable differences between arrays in C and Fortran. These include:
  - C array indices start from 0. (1 in Fortran).
  - Array elements in C are accessed using square brackets (ex. `a[2]`). (Round brackets in Fortran, ex. `a(2)`).

- Syntax (for a 1-dimensional array):

```
type identifier[size];
```

  where `type` is any valid C data type, `identifier` is any valid C variable name, and `size` is the number of elements in the array.

Array declaration

- The array indices run from 0 to `size-1`.
- Examples:

```
int assignments[8];
float failure[12];
```

- Array values can also be initialized on the declaration line as follows:

```
type identifier[size]={list of values};
```

- Examples:

```
int assignments[8]={72,89,65,79,82,97,80,86};
float failure[12]={3.2,2.6,4.0,2.6,3.2,3.5,3.7,
                   4.1,3.6,2.9,3.4,3.5};
```

Example 1: Arrays

Compute mean time to failure and display values greater than the mean.

```
// compute mean time to failure
// and print values less than the mean

#include <stdio.h>

int main(int argc, char *argv[]) {
   const int numtimes=10;
   int i;
   float ftime[numtimes], sum, mean;

   sum=0.0;

   for(i=0;i<numtimes;i++) {
      printf("Enter failure time %d:",i);
      scanf("%f",&ftime[i]);
      sum = sum + ftime[i];
   }
   mean = sum/numtimes;
   printf("Mean time to failure is %f\n",mean);

   printf("List of failure times greater than the mean:\n");
   for(i=0;i<numtimes;i++) {
      if (ftime[i] > mean) printf("%f\n",ftime[i]);
   }

   return 0;
}
```

Example 2: Arrays

```
// compute average, maximum, and minimum assignment grades

#include <stdio.h>

int main(int argc, char *argv[])
{
   int assign[99],i,n;
   double assign_avg,sum;
   int assign_max,assign_min;

   printf("How many assignment grades would you like to enter?\n");
   scanf("%d",&n);

   /* Read in the assignment marks and compute the sum */
   sum=0.0;
   for(i=0;i<n;i++) {
      printf("Enter assignment mark %d\n",i);
      scanf("%d",&assign[i]);
      sum=sum+assign[i];
   }
   /* Compute the average assignment mark */
   assign_avg=sum/n;
   printf("The average assignment mark is %lf\n",assign_avg);

   /* Compute the maximum and minimum assignment mark */
   assign_max=assign[0];
   assign_min=assign[0];
   for(i=1;i<n;i++) {
      if (assign[i]>assign_max) assign_max=assign[i];
      if (assign[i]<assign_min) assign_min=assign[i];
   }
   printf("The maximum assignment mark is %d\n",assign_max);
   printf("The minimum assignment mark is %d\n",assign_min);

   return 0;
}
```

Passing arrays into functions

- Recall that arguments passed into functions in C are passed-by-value.

- That is, a local copy of each argument is made, and any change made to an argument passed into a function is not seen by the calling function.

- If we wish to have the value of an argument that is changed within a function retained when the function returns, we must use a pointer to that variable as the argument, rather than the value itself.

- In C, array variables are actually constant pointers. Thus, when an array is passed into a function, that function is actually given the direct memory address of the array.

- Any changes to array elements within a function actually affect the original array in the calling function.

Example: Arrays and functions

```
// compute average, maximum, and minimum assignment grades
// using a function for average, maximum, and minimum

#include <stdio.h>

double compute_avg(int marks[99],int n);
int compute_max(int array[99],int n);
int compute_min(int array[99],int n);

int main(int argc, char *argv[])
{
   int assign[99],i,n;
   double assign_avg;
   int assign_max,assign_min;

   printf("How many assignment grades would you like to enter?\n");
   scanf("%d",&n);

   /* Read in the assignment marks */
   for(i=0;i<n;i++) {
      printf("Enter assignment mark %d\n",i);
      scanf("%d",&assign[i]);
   }

   assign_avg=compute_avg(assign,n);
   printf("The average assignment mark is %lf\n",assign_avg);

   assign_max=compute_max(assign,n);
   assign_min=compute_min(assign,n);
   printf("The maximum assignment mark is %d\n",assign_max);
   printf("The minimum assignment mark is %d\n",assign_min);

   return 0;
}
```

```
double compute_avg(int array[99],int n){
   double sum;
   int i;
   sum=0.0;
   for(i=0;i<n;i++) {
      sum+=array[i];
   }
   return sum/n;
}

int compute_max(int array[99],int n){
   int max;
   int i;
   max=array[0];
   for(i=1;i<n;i++) {
      if (array[i]>max) max=array[i];
   }
   return max;
}

int compute_min(int array[99],int n){
   int min;
   int i;
   min=array[0];
   for(i=1;i<n;i++) {
      if (array[i]<min) min=array[i];
   }
   return min;
}
```

Example: Insertion sort

```
// insertion sort

#include <stdio.h>

void shift(int array[999], int end, int start);

int main(int argc, char *argv[])
{
   int to_sort[999];
   int i,j,k,num,current,position;

   printf("How many numbers are to be read?\n");
   scanf("%d",&num);
   printf("Enter value\n");
   scanf("%d",&current);
   printf("List thus far:\n");
   printf("%d\n",current);

   to_sort[0]=current; // Put the first element in the array
   for(k=1;k<num;k++) {  // For each element
      printf("Enter value\n");
      scanf("%d",&current); // Get next element
      position=0;  // Assume that the element belongs at the beginning
      for(j=0;j<k;j++) {  // Check where element belongs
         if (current > to_sort[j]) position=j+1;
      }
      printf("Element belongs in position %d\n",position);
      if (position < k) {  // Need to shift part or all of the list
         printf("Shifting elements %d to %d ahead\n",position,k-1);
         shift(to_sort,k,position);  // Shift elements from position to k-1
         to_sort[position]=current; // Add current element in correct position
      }
      else {
         to_sort[position]=current; // Add element to the end of the list
      }
```

```
      printf("List thus far:\n");
      for(i=0;i<=k;i++) {
         printf("%d ",to_sort[i]);
      }
      printf("\n");
   }
   printf("\n");
   printf("Final sorted list:\n");
   for(i=0;i<num;i++) {
      printf("%d ",to_sort[i]);
   }
   printf("\n");

   return 0;
}

void shift(int array[999], int end, int start){
   /* Shift the elements of array from index start to index end-1
    * ahead by one, leaving a gap at index start.  */
   int i;
   for(i=end;i>start;i--) {
      array[i]=array[i-1];
   }
   return;
}
```