**To do list week of 4/1/14 after the test**

Passing an array

1. include a prototype which indicates that the parameter passed is an array, without actually showing the number of elements in the array - just put [].

int getscores (int scores[])

1. The function heading also has similar declaration.
2. The variable declaration however, should show the dimension of the array.

int scores[90]

1. Just pass the name of the array variable when calling a function

n = getscores(scores)

It is important to note that the array parameter passed to the function is neither a **call-by-value**, or a **call-by-reference**, but a new kind of parameter known as an **array parameter**. It follows that when passing an array neither a copy of the entire array is passed nor every memory location of entire array are passed, instead the memory location of the first element of the array is passed. Therefore, it is not necessary to pass the size of the array, just include square brackets with nothing in it. Both the calling module and called module calculate the memory location of a particular element using a base and offset (calculated by index). More about addressing modes will be taught in future courses.

Searching Array

Linear. Array is also called a list. Search through an array to look for a match. Once you find it return the index number. In a linear search you have to look at every element until you find it. Worst case scenario is that it may be the last item on the list. Think of looking through a telephone directory, starting with the first name until you find the name you are looking for.

Binary – The list has to be ordered (sorted) first. Cut the list into half each time a comparison is made.

Passing an array

|  |
| --- |
| #include <iostream.h>  int getscores (int scores[]); //prototype for passing an array  int main ()  {  int scores[90]; //array of 90 integers named scores  int n, i; // n for number of scores. i is a counter.  n = getscores(scores); //go get the scores and how many  for (i=1; I<=n; i++) cout<<scores[i]<<endl; //show scores.  Return(0);  }  int getscores(int scores[])  {  int n=1;  cout << "ENTER A SCORE AND PRESS ENTER. YOU QUIT ANY TIME BY ENTERING A NEG NUMBER!\n";  cout << "Enter score# " << n << " ";  cin >> scores[n];  while (scores[n] >= 0)  {  n++;  cout << "Enter score# " << n <<" ";  cin >> scores[n];  }  return n-1; //n-1 actual scores read.  } |

Linear Search Example. Given the name of a month, print month number. The searchMonth is compared to each element in the array, starting with the first, when found the index number (which happens to be the month number) is returned.

|  |
| --- |
| #include "stdafx.h"  #include <iostream>  #include <string>  using namespace std;  int main()  {  int monthNum;  bool found=false;  string searchMonth;  string month[] = {"","JAN","FEB","MAR","APR","MAY","JUN",  "JUL","AUG","SEP","OCT","NOV","DEC"};    cout<<"\nEnter a month name using uppercase three character format ('QUIT' to exit the program) ";  cin>>searchMonth;  while (searchMonth !="QUIT")  {  monthNum=1;    while (monthNum <=12 && !found)  {  if (searchMonth==month[monthNum])  {  cout <<"\nThe number of the month you entered is: "<<monthNum;  found=true;  }  else monthNum++;  }  if (monthNum>12) cout<<"\nPlease recheck the spelling!";  cout<<"\nEnter a month name using uppercase three character format ('QUIT' to exit the program) ";  cin>>searchMonth;  found=false;  }      return 0;  } |