Welcome to 6.096

Lecture 4
January 12, 2009
Arrays

• A collection of a fixed number of variables of the same type stored sequentially in the memory

• Element $\rightarrow$ an item in the array

• Dimension $\rightarrow$ size of the array
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>34</td>
<td>23</td>
<td>2</td>
<td>76</td>
<td>12</td>
<td>45</td>
</tr>
</tbody>
</table>

Memory units

Figure by MIT OpenCourseWare.
Declaring arrays in C++

- type varName [size];
- int arr[10] ; char alphabet [26];
- An array of the int datatype called ‘arr’ composed of 10 elements
Initializing arrays

• Elements must be initialized before usage

• int arr [5] = { 2, 43, 32, 34, 13 };

• char arr [] = { ‘r’, ‘T’, ‘g’, ‘B’ };

• char arr[50];
  for (int i=0; i<50 ;i++)
    arr[i] = ‘ ‘;
Entering data in an array

- #include <iostream>
  
  using namespace std;
  
  int main()
  {
    int arr[5];
    
    cout<<"Enter 5 integers: " <<endl;
    
    for (int i=0; i<5; i++)
    {
      cin>>arr[i];
    }
    
    cout<<"Data has now been recorded!";
    return 0;
  }
Working with arrays

• Number inside [] must be a positive integer less than the dimension of the array

• Indexes: first element ----> 0
  last element ----> N-1
  (where N is the total number of elements)

• arr [i+2]
  arr [i*j]

• Treat arr[i] just like any other variable
#include <iostream>
using namespace std;
int main()
{
    int arr[5] = {23, 234, 1234, 14, 11} ;
    cout<<"The elements of the array are:"<<endl;
    for (int i=0; i<5; i++)
    {
        cout<<arr[i]<<' ';
    }
    return 0;
}
Example

// Program to copy the contents of an array into the other

#include <iostream>
    using namespace std;
    int main()
    {
        int iMarks[4] = {78, 64, 66, 74};
        short newMarks[4];

        for(int i=0; i<4; i++)
            newMarks[i]=iMarks[i];

        cout<<"The new array is :"<<endl;

        for(int j=0; j<4; j++)
            cout<<newMarks[j]<<endl;

        return 0;
    }
Linear search in arrays

```cpp
#include <iostream>
using namespace std;

int main()
{
    int num;
    int Account[10] = {5658845, 4520125, 7895122, 8777541, 8451277, 1302850, 8080152, 4562555, 552012, 5050552};

    cout << "Enter Number \n";
    cin >> num;

    for(int i=0; i<10; i++)
    {
        if(Account[i] == num)
        {
            cout << "Element found at index number " << i; 
            break;
        }
        else
        
            cout << "Element not found!";
    }
    return 0;
}
```
Bubble sort algorithm

• Sorts numbers in ascending/descending order

• How does it work:

First Pass:
(5 1 4 2 8) (1 5 4 2 8) Here, algorithm compares the first two elements, and swaps them.
(1 5 4 2 8) (1 4 5 2 8)
(1 4 5 2 8) (1 4 2 5 8)
(1 4 2 5 8) (1 4 2 5 8) Now, since these elements are already in order, algorithm does not swap them.

Second Pass:
(1 4 2 5 8) (1 4 2 5 8)
(1 4 2 5 8) (1 2 4 5 8)
(1 2 4 5 8) (1 2 4 5 8)
(1 2 4 5 8) (1 2 4 5 8)
Now, the array is already sorted, but our algorithm does not know if it is completed. Algorithm needs one whole pass without any swap to know it is sorted.

Third Pass:
(1 2 4 5 8) (1 2 4 5 8)
(1 2 4 5 8) (1 2 4 5 8)
(1 2 4 5 8) (1 2 4 5 8)
(1 2 4 5 8) (1 2 4 5 8)
# Bubble sort code snippet

```cpp
#include <iostream>
using namespace std;

int main()
{
    int array [5] = {12, 234, 345, 1234, 51};
    int i, j, k;
    for(i=0; i<5; i++)
    {
        for(j=0; j<i; j++)
        {
            if(array[i] > array[j])
            {
                int temp = array[i]; // swap
                array[i] = array[j];
                array[j] = temp;
            }
        }
    }

    cout << "The array is : " << endl;
    for(k=0; k<5; k++)
    {
        cout << array[k] << " ";
    }

    return 0;
}
```
Binary search algorithm

- Searching for an element in an array
- Faster and more efficient than linear search
- The array needs to be sorted first
Figure by MIT OpenCourseWare.
Binary search code snippet

```cpp
#include<iostream>
using namespace std;

int main()
{
    int arr[5] = {1,2,3,4,5};
    int N = 6; // Search this number
    int low = 0, middle, high=5;

    bool found = false;

    while (low <= high)
    {
        middle = (low+high)/2;
        if(arr[middle]==N)
        {
            found = true;
            cout<<"Element found";
            break;
        }
    }
}```
else if (N < arr[middle])
    high = middle-1; //search low end of array

else
    low = middle+1; //search high end of array
}

if(found==false)
    cout<<"Not found";

    return 0;
}

• Additionally, you can also determine the location of the element you were looking for.
Multidimensional arrays

- Multidimensional array: ‘an array of arrays’
- char century [100][365][24][60][60];
- int arr[3][5];
Example

- Here is a sample program that stores roll numbers and marks obtained by a student side by side in matrix

```cpp
int main ()
{
    int stud [4] [2];
    int i, j;
    for (i =0; i <=3; i ++)
    {
        cout<< "Enter roll no. and marks";)
        cin>>stud [i] [0]>>stud [i] [1] ;
    }

    for (i = 0; i <= 3; i ++)
        cout<<stud [i] [0]<< stud [i] [1]);

    return 0;
}
```
• Multidimensional arrays are just an abstraction for programmers, since we can obtain the same results with a simple array just by putting a factor between its indices.

• int arr [3][5]; // is equivalent to int arr [15]; // (3 * 5 = 15)