Arrays

- Lecture 11

What is an Array?

- A Collection of Data all of which is of the same type

Array of Numbers

Index	Score
0	100
1	75
2	80
3	66

Array of Letters

Index	Grade
0	A
1	C
2	B
3	D

Declaring and Using Arrays

- Declaration
  - Syntax:

    Type_name Array_Name[Declared_size]
  - Example:

    int score[4];

Array of Integers

Index	Score
0	100
1	75
2	80
3	66

score[**0**]
score[1]
score[2]
score[3]

Declaring Arrays

Syntax

```
datatype name [size];                                  /* not initialized */

datatype name [size] = {initialization list} /* initialized */
```

optional

Examples

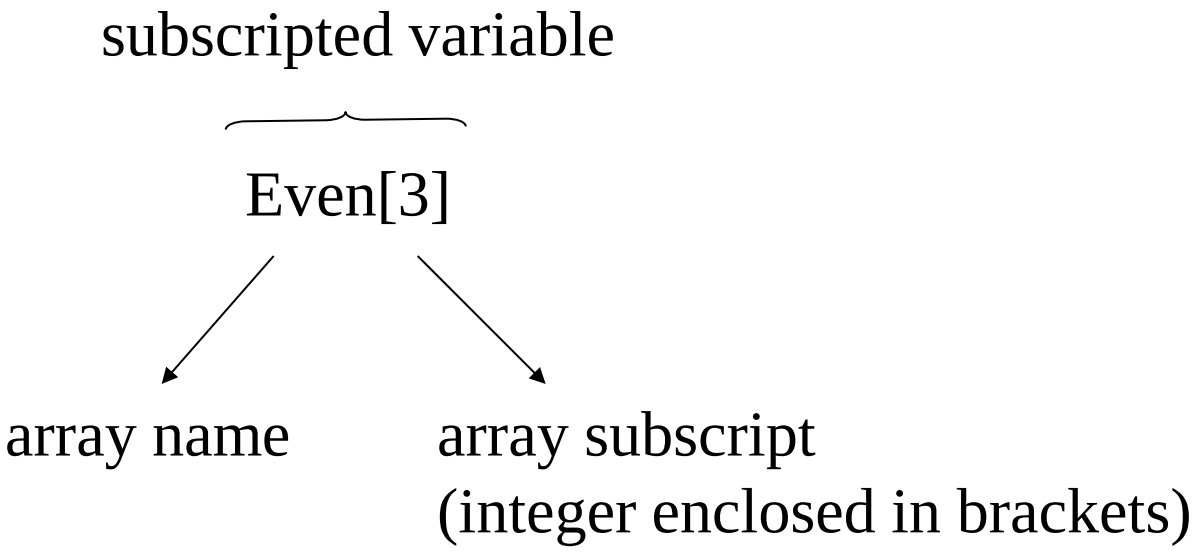
```
int even[5];

int even [5] = {2,4,6,8,10};

int even[5], num=0, x[10], y;

double j[] = {1.0,4.2,6.5,8.1,10.4}, d, k[3];
```

Referencing Arrays



Examples

even[3] ⟶ read as: even sub three

x[0] ⟶ read as: x sub zero

Declaring and Referencing Arrays

Int even[5]; ⟶

even[0]
even[1]
even[2]
even[3]
even[4]

subscripts always: starts from 0 until size-1

Int even[] = {2,4,6,8,10}; ⟶

even[0] = 2
even[1] = 4
even[2] = 6
even[3] = 8
even[4] = 10

Exercise 1

Char grades [5];

- *How many memory cells are reserved in memory?*

  5 memory cells

- *What type of data will be stored there?*

  character

- *How do we refer to the first array element?*

  grades[0]

- *How fo we refer to the last array element?*

  grades[4]

Exercise 2

- *Declare one array for storing the square roots of the integers*
- *from 0 through 10.*

Double sr[11];

- *Declare one array for storing the cubes of the same integers.*

Int cb[11];

Declaring and Using Arrays

- Using & Referencing Arrays
  - Elements of the array are called indexed variables
  - Can be used anyplace that an ordinary variable of the same

    type is used

Read: cin >> score[2];

Write: cout << score[1];

Assign: score[3]= score[1] + score[2];

score[student_id] = score[2];

score

Index	Score
0	100
1	75
2	80
3	66

Reading from & Writing to Arrays

- For (i=0; i<size, i++)
  cout << , x[i];

- For (i=0; i<=size-1, i++)
  cin >> x[i];

- For (i=0; i<size, i++)
  x[i] = i *5;

Example 1

```
/* program to compute the sum and the sum of the squares of all
data elements in an array */

#include <iostream>
#include <cmath>
 using namespace std;
int main (void)
{ int x[5] = {1,2,3,4,5}, i, sum=0, sum_sqr=0;
   for (i=0; i<5; i++)
         { sum+=x[i];
            sum_sqr+=pow(x[i],2); }
   cout << "The sum of the array elements are“ <<sum << endl;
   cout << "The sum of their squares are “ << sum_sqr << endl;
   return 0;}
```

Example 2

```
/* program that reads 10 integers and prints them in
reverse order */
#include<iostream>
 using namespace std;
int main()
{
 int number [10], i;

 for (i=0; i<=9; i++)
      cin >> number[i]);

 for (i=9; i>=0; i--)
      cout <<  number[i]);
 return(0);
}
```

Example 3

```
/* Program that calculates the price of four items */

#include<iostream>
 using namespace std;
int main()
{ int i, quantity;
   float total=0, price[4]={5.77, 3.15, 2.50, 1.35};
   for (i=0; i<=3; i++)
        {
          cout << "Enter the quantity of item : “ << i << endl;
          cin >> quantity; /*read quantity of item i */
          total+=quantity*price[i];
        }
   cout << "The total cost is “<<total <<endl;
   return 0;}
```

Example 4

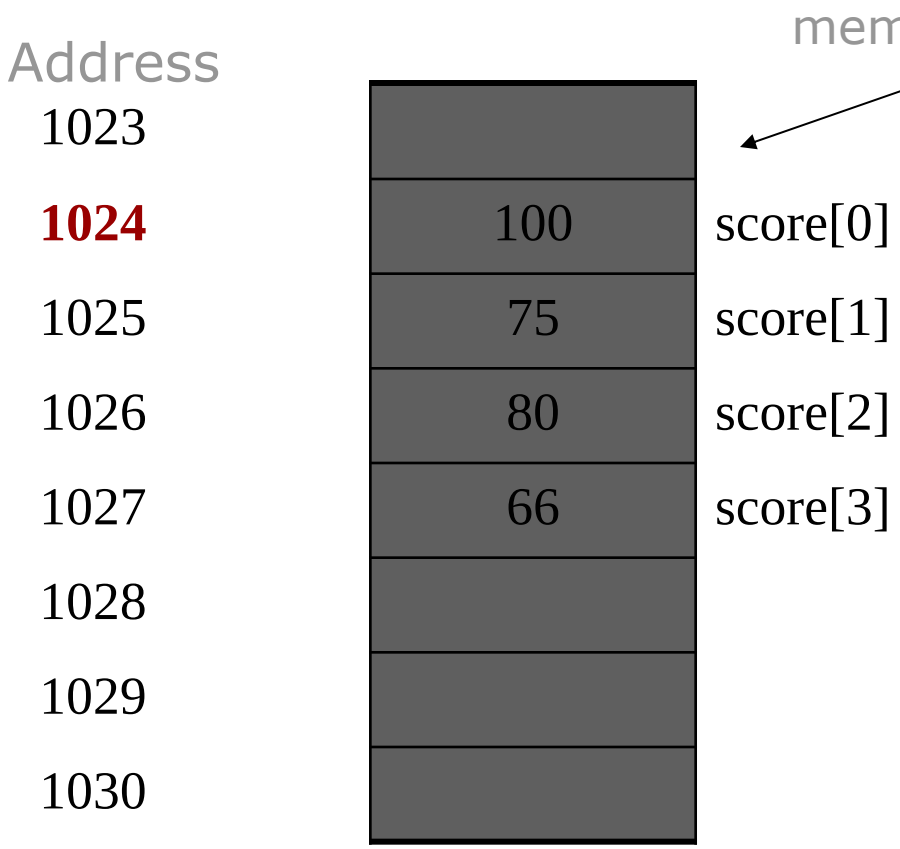
```
/* Program that determines the maximum element in an array */

#include <iostream>
 using namespace std;
int main()
{
 int x [5] = {12, 5, 21, 6, 4},i,max;
 max = x[0]; /* assume that max is the first element */
 for (i=1; i<=4; i++) /* finding the maximum */
     if (x[i] > max)
                          max = x[i];
 cout << "The maximum is“ << max << endl;
 return 0;
}
```

Arrays in Memory

- Contiguous locations:

```
int score[4];
```



score

Index	Score
0	100
1	75
2	80
3	66

What can we do with Arrays?

- Read/Write
- Initialize
- Search
- Sort
- Manipulate values

Initializing Arrays

- int score_w[4] = {60, 60, 60, 60};
- char score_x[4] = {‘A’, ‘A’,’A’,’A’};
- int score_y[4] = {100, 100};
- int score_z[] = {100, 100, 100};

score_w

Index	Score
0	60
1	60
2	60
3	60

score_x

Index	Score
0	A
1	A
2	A
3	A

score_y

Index	Score
0	100
1	100
2	0
3	0

score_z

Index	Score
0	100
1	100
2	100

Using Arrays

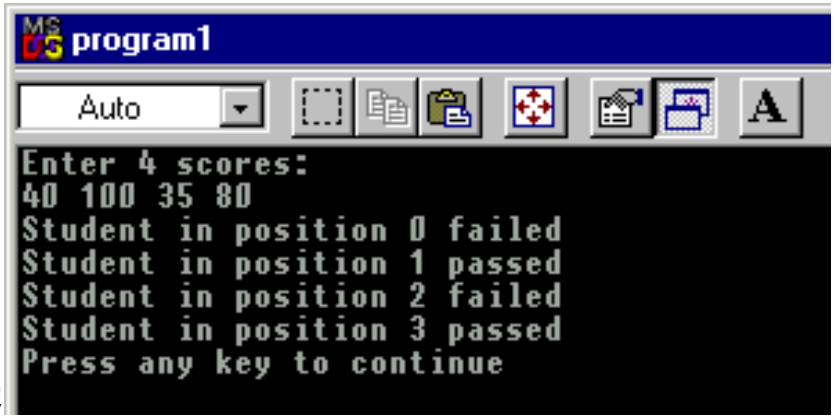
- To step through all indexed variables of an array, use a for statement:

```
for (int index = 0;
        index < Declared_size_of_Array;
        index++)
{
   Do something to your_array[index]
}
```

Using Arrays

- Example

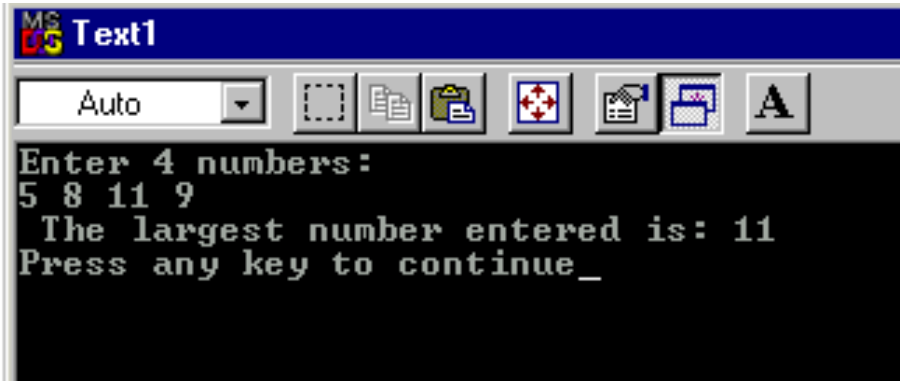
```
#include <iostream>
int main()
{       using namespace std;
        int i, score[4];
        cout<< "Enter 4 scores:\n";
        for (i=0; i<4; i++)
        {       cin >> score[i];
                if (score[i] >= 60)
                cout << "Student in position " << i <<" passed\n";
                else
                cout << "Student in position " << i <<" failed\n";
        }
}
```



Using Arrays

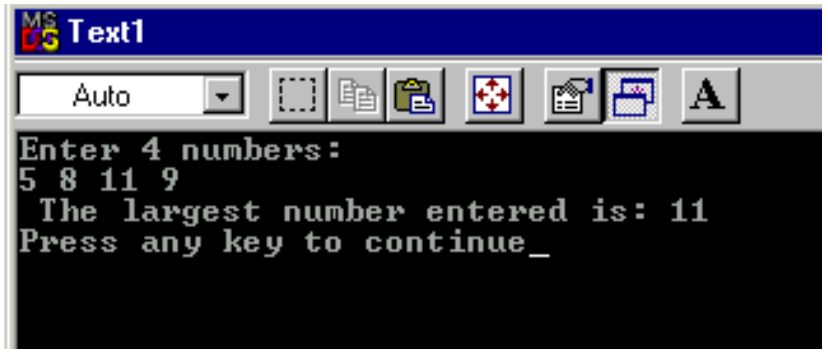
- Try This:

  Write a program that reads 4 positive integers from the user, stores them into an array then finds the maximum number



Using Arrays

- Sample Solution:




```
#include <iostream>
int main()
{       using namespace std;
        int i, my_list[4], max=-1;
        cout<< "Enter 4 numbers:\n";
        for (i=0; i<4; i++)
        {
                cin >> my_list[i];
                if (my_list[i] >= max)
                max = my_list[i];
        }
        cout<< " The largest number entered is: " <<max <<endl;
}
```

Arrays in Functions

- You can use indexed variables or entire arrays as arguments to functions
- Indexed Variables
  - my_function(array[3]);
- Entire Arrays
  - get_scores(score, number_of_scores);

Arrays in Functions

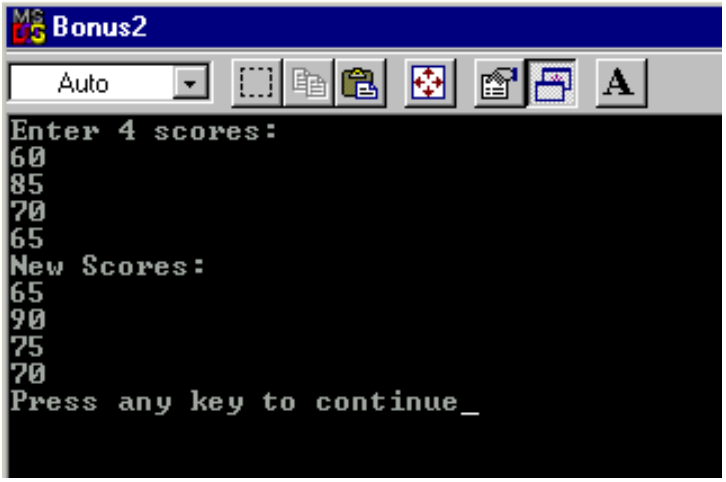
- Indexed Variables

```
include<iostream>
int give_bonus(int old_score);

main()
{     using namespace std;
      int score[4], number;

      cout << "Enter 4 scores: \n";
      for (number = 0; number < 4; number++)
      cin>> score[number];
      for (number = 0; number < 4; number++)
      score[number] = give_bonus(score[number]);
      cout<< "New Scores: \n";
      for (number = 0; number < 4; number++)
      cout<< score[number]<< endl;;}

int give_bonus(int old_score)
{return (old_score+5);}
```



Arrays in Functions

- Entire Arrays

Syntax:

Type_returned Function_name (.., BaseType Array_Name[], …);

- Example

void sum_array (double a[], int size);

→ call: sum_array (a, 5)

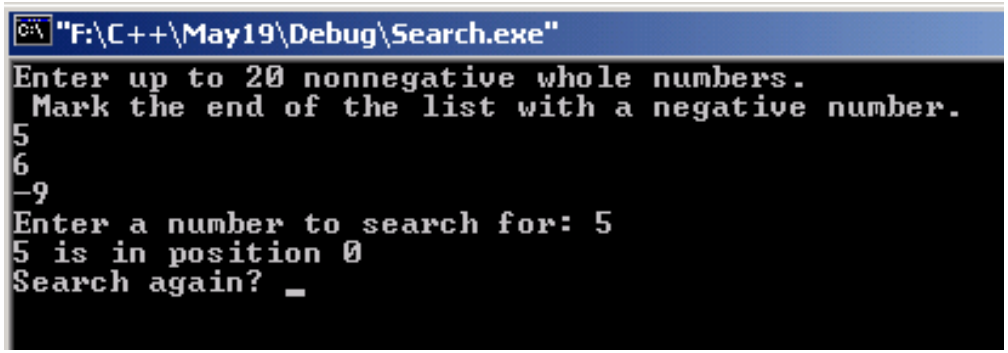
Include the size because only the address of the 1st element is passed

- Using *Const*

void sum_array (const double a[], int size);

Searching Partially Filled Arrays

- Example:




```
#include<iostream>
const int Declared_Size=20;

void fill_array(int a[], int size, int& number_used);

int search(const int a[], int number_used, int target);
```

Searching Arrays Example…continued

```
int main()
{    using namespace std;
     int arr[Declared_Size], list_size, target;
     fill_array(arr, Declared_Size, list_size);
     char ans;
     int result;
     do
     {
          cout<<"Enter a number to search for: ";
          cin>> target;
          result = search(arr, list_size, target);
          if (result==-1)
               cout<< target<< " is not on the list\n";
          else
               cout<< target << " is in position " << result <<endl;
          cout<< "Search again? ";
          cin>>ans;
     } while((ans!='n')&&(ans!='N'));
     cout <<"End of program\n";
     }
```

Searching Arrays Example…continued

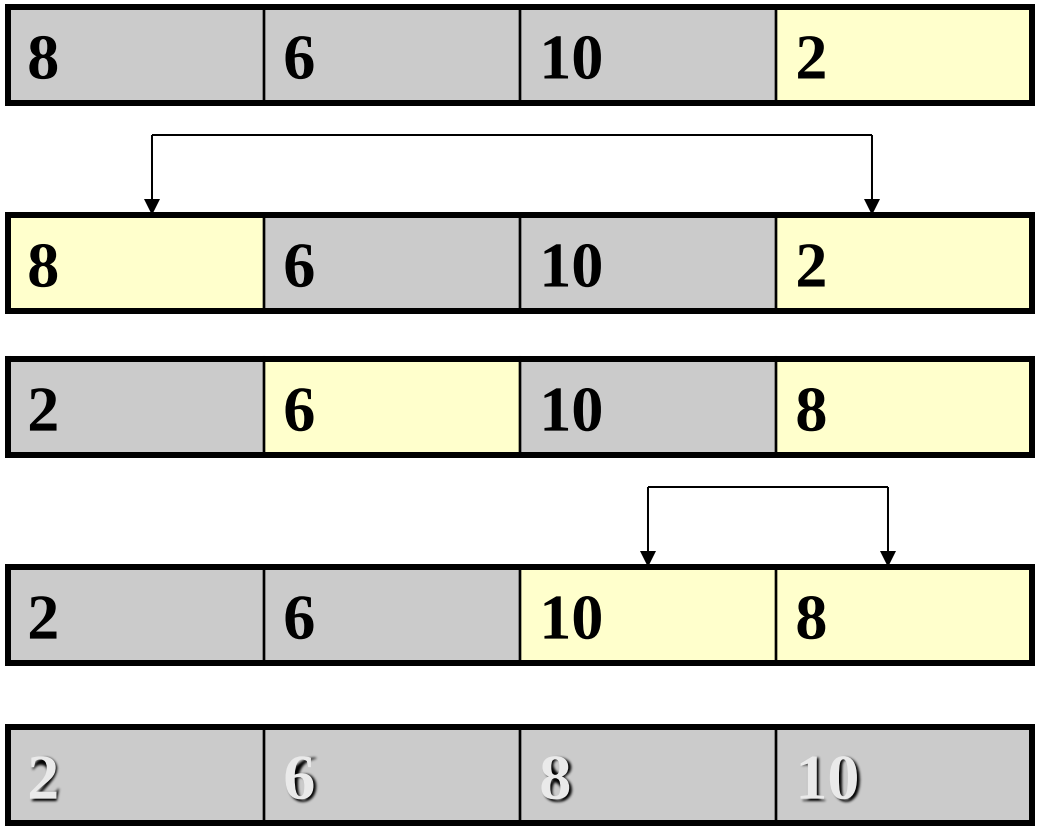
```
int main()
{    using namespace std;
     int arr[Declared_Size], list_size, target;
     fill_array(arr, Declared_Size, list_size);
     char ans;
     int result;
     do
     {
          cout<<"Enter a number to search for: ";
          cin>> target;
          result = search(arr, list_size, target);
          if (result==-1)
               cout<< target<< " is not on the list\n";
          else
               cout<< target << " is in position " << result <<endl;
          cout<< "Search again? ";
          cin>>ans;
     } while((ans!='n')&&(ans!='N'));
     cout <<"End of program\n";
     }
```

Searching Arrays Example…continued

```
int search(const int a[], int number_used, int target)
{
    int index =0;
    bool found = false;
    while ((!found)&& (index<number_used))
    if (target==a[index])
            found = true;
    else
            index++;
    if (found)    return index;
    else          return -1;
}
```

Sorting Arrays

- Selection Sort



Sorting Arrays

- Example: A program that sorts an array of up to 10 positive numbers

```
"F:\C++\May19\Debug\Sort.exe"
This program sorts numbers from lowest to highest.
Enter up to 10 nonnegative whole numbers.
Mark the end of the list with a negative number.
8 6 10 2
-1
In sorted order the numbers are:
2 6 8 10
Press any key to continue
```

Sorting Arrays

- Example:

```
#include<iostream>

void fill_array(int a[], int size, int& number_used);
void sort(int a[], int number_used);
void swap_values (int& v1, int& v2);
int index_of_smallest(const int a[], int start_index, int number_used);
int main()
{     using namespace std;
      cout<<"This program sorts numbers from lowest to highest.\n";

      int sample_array[10], number_used;
      fill_array(sample_array, 10, number_used);
      sort (sample_array, number_used);

      cout<<"In sorted order the numbers are:\n";
      for (int index=0; index<number_used; index++)
            cout<<sample_array[index] << " "; cout<<endl;
}
```

Sorting Arrays

- Example (continued):

```
void sort(int a[], int number_used)
{
    int index_of_next_smallest;
    for (int index = 0; index<number_used-1; index++)
    {
        index_of_next_smallest= index_of_smallest(a, index,
    number_used);
        swap_values(a[index], a[index_of_next_smallest]);
    }
}
```

Sorting Arrays

- Example (continued):

```
void swap_values (int& v1, int& v2)
{    int temp;
     temp = v1;
     v1=v2;
     v2=temp;}


int index_of_smallest (const int a[], int start_index, int number_used)
{    int min= a[start_index],
     index_of_min = start_index;
     for (int index=start_index+1; index<number_used; index++)
     if (a[index]<min)
     {       min = a[index];
             index_of_min = index;
     }
     return index_of_min;
}
```

Arrays and Classes

- Array of Structs or Classes

```
class Student
{public:
        int id;
        char Name[10];
        char Major[2];
};
```

- Student School[500];

Index	ID	Name	Major
0	999	Sara	CS
1	555	Nora	IS
2	222	May	IS

- Student
  - ID
  - Name
  - Major

Arrays and Classes

```
Student School[500];
…
For (int i=0; i<500; i++)
{ cout<<“Enter id: “;cin>>School[i].id;
   cout<<“Name: “;      cin>>School[i].Name;
   cout<<“Major: “;  cin>>School[i].Major;
}
```

Arrays and Classes

♦ Class with array members

```
class Student
{
 public:
   int ID;
   int Grades[10];
}
```

Arrays and Classes

- Try this:

Write a program that reads data into an array of 5 items where each item is a class such as the following:

- Item
  - Code
  - Price

Index	Code	Price
0	123	5.25
1	456	6.30
2	789	9.50

Make *item* an ADT:

- Use member functions to read/write data as well as constructors in your class definition.
- Code &Price should be private members.