Fundamentals of Structured Programming

Lecture 2

Prof. Zaki Taha Fayed

Dr. Sally Saad

SallySaad@gmail.com

DropBox folder link

https://www.dropbox.com/sh/85vnrgkfqgrzhwn/AABdwKLJZqZs26a7u-y0AFwia?dl=0

Credits to Dr.Salma Hamdy for Content Preparation

Remarks from Previous Lecture

- **#define** test "string" → OK
- **#define** test2 x

```
void main ()
{
   int x=10;
   cout<<test2; →OK
}
```

Remarks from Previous Lecture

- **C++ Created Files/Folders**

Solution Folder include:

Debug folder

- *Includes **.exe** file (Application itself)*
- *Includes other files related to the debug or the release of the project*

Name	Date modified	Type	Size
Debug	07-Feb-18 12:42 AM	File folder	
Lecture1	06-Feb-18 9:45 PM	File folder	
Lecture1	07-Feb-18 12:53 AM	SQL Server Comp...	6,912 KB
Lecture1	06-Feb-18 9:40 PM	Microsoft Visual S...	1 KB

***.sln** file

Opens the whole solution on the IDE

database file

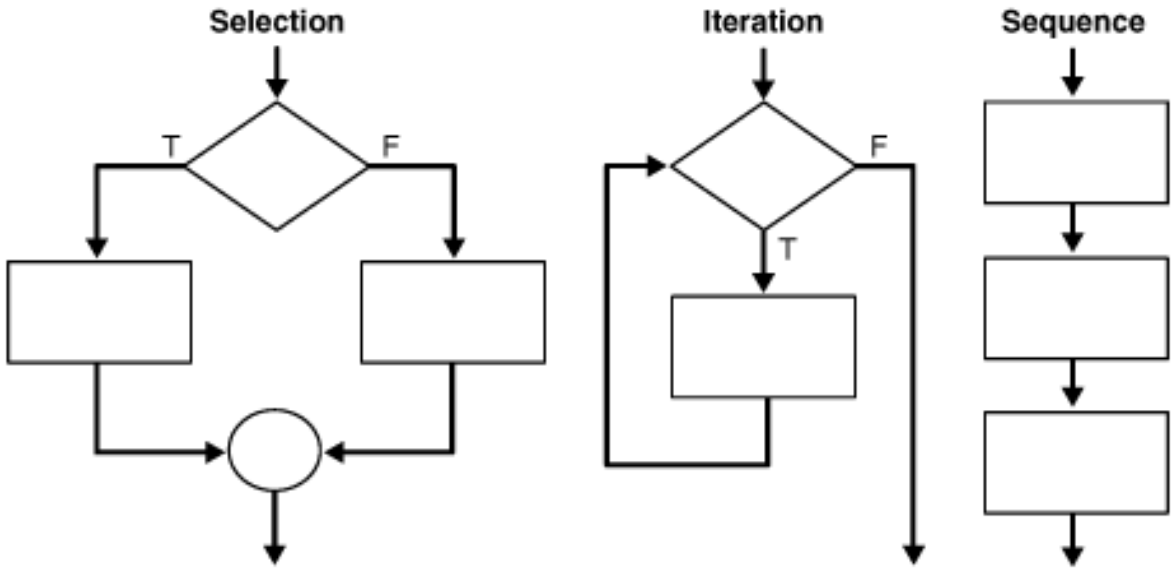
Beyond our scope ☺

Solution Folder

- *Includes different Project(s) within your solution*
- *Includes Source file(s)**.cpp file(s)***
- *Includes other projects files like **.h file(s)***
- *Includes other files like **log** files*

Contents

Control Structures



A. Branching: if Statement

- **Nested if-else statement**

```
if ( studentGrade >= 90 ) // 90 and above gets "A"
   cout << "A";
else
   if ( studentGrade >= 80 ) // 80-89 gets "B"
      cout << "B";
   else
      if ( studentGrade >= 70 ) // 70-79 gets "C"
         cout << "C";
      else
         if ( studentGrade >= 60 ) // 60-69 gets "D"
            cout << "D";
         else // less than 60 gets "F"
            cout << "F";
```

Both are identical

```
if ( studentGrade >= 90 ) // 90 and above gets "A"
   cout << "A";
else if ( studentGrade >= 80 ) // 80-89 gets "B"
   cout << "B";
else if ( studentGrade >= 70 ) // 70-79 gets "C"
   cout << "C";
else if ( studentGrade >= 60 ) // 60-69 gets "D"
   cout << "D";
else // less than 60 gets "F"
   cout << "F";
```

Most popular

A. Branching: if Statement (cont.)

- **Common Errors:**
  - Adding semi-colon at the end of the condition
  - Forgetting the braces for blocks of code.
  - Dangling else.

```
if ( x > 5 )
   if ( y > 5 )
      cout << "x and y are > 5";
else
   cout << "x is <= 5";
```

```
if ( x > 5 )
{
   if ( y > 5 )
      cout << "x and y are > 5";
}
else
   cout << "x is <= 5";
```

A. Branching: Switch Statement

Remark: Used Only for <u>equal</u> comparison for <u>integers</u> or <u>characters</u>

switch Statement

SYNTAX

```
switch (Controlling_Expression)
{
    case Constant_1:
        Statement_Sequence_1
        break;
    case Constant_2:
        Statement_Sequence_2
        break;
                    .
                    .
                    .
    case Constant_n:
        Statement_Sequence_n
        break;
    default:
        Default_Statement_Sequence
}
```

You need not place a **break** *statement in each case. If you omit a* **break**, *that case continues until a* **break** *(or the end of the* **switch** *statement) is reached.*

A. Branching: Switch Statement (cont.)

EXAMPLE

```
int vehicleClass;
double toll;
cout << "Enter vehicle class: ";
cin >> vehicleClass;

switch (vehicleClass)
{
    case 1:
        cout << "Passenger car.";
        toll = 0.50;
        break;
    case 2:
        cout << "Bus.";
        toll = 1.50;
        break;
    case 3:
        cout << "Truck.";
        toll = 2.00;
        break;
    default:
        cout << "Unknown vehicle class!";
  }
```

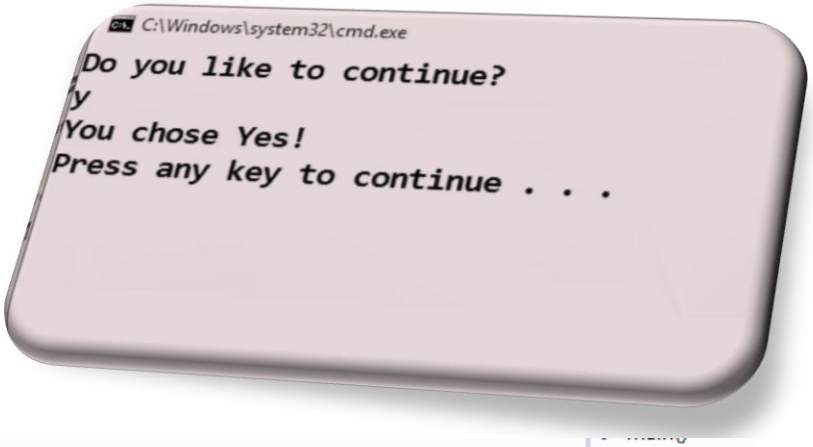
If you forget this **break**, *then passenger cars will pay $1.50.*

A. Branching: Switch Statement (cont.)

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

void main()
{
    cout<<"Do you like to continue?"<<endl;
    char choice;
    cin>>choice;
        if(choice=='Y' || choice=='y')
            cout<<"You chose Yes!\n";
        else
            cout<<"You chose No!\n";

}
```




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

void main()
{
    cout<<"Do you like to continue?"<<endl;
    char choice;
    cin>>choice;
        switch(choice)
        {
        case 'Y':
        case 'y':
            cout<<"You chose Yes!\n";
            break;
        case 'N':
        case 'n':
            cout<<"You chose No!\n";
            break;
        default: cout<<"Wrong choice!\n";
        }

}
```

B. Iteration: for Loop

```
for(start_count; bool_expression; action)
    statement;
```

Any of them can be empty statement

head: how many times to repeat body…track variable i values

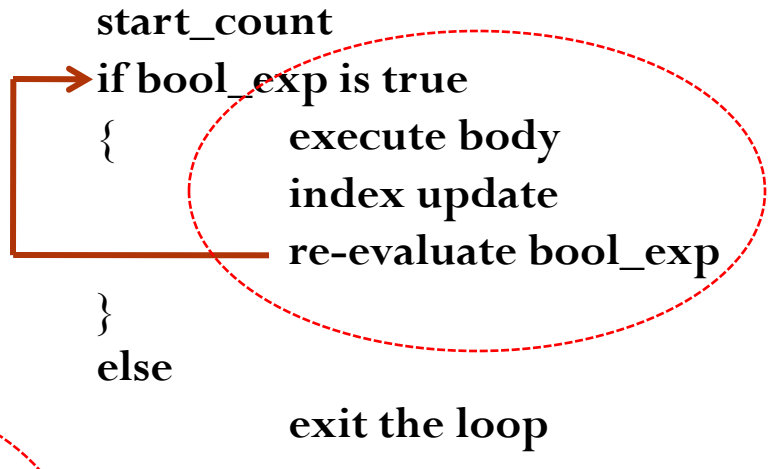
```
for(int i=0; i<3; i++)
    cout<<"Hello! I'm in a loop!\n";
    cout<<i;
```

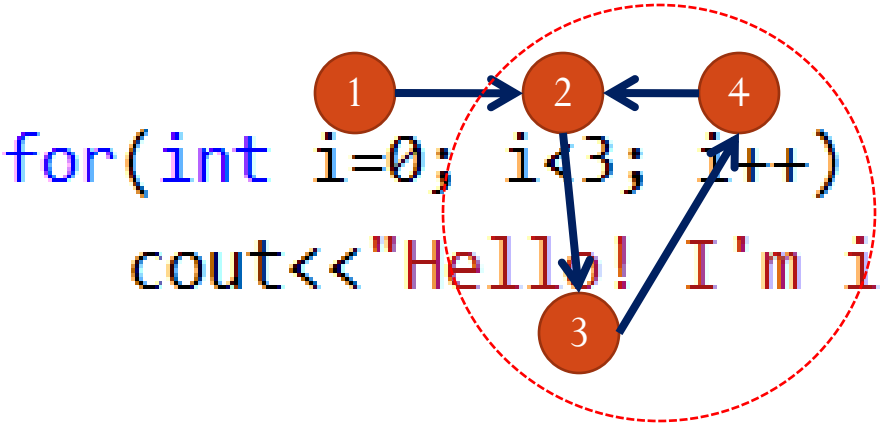
body: part of code that is repeated

B. Iteration: for Loop (cont.)

for(start_count, bool_expression, action)

statement;



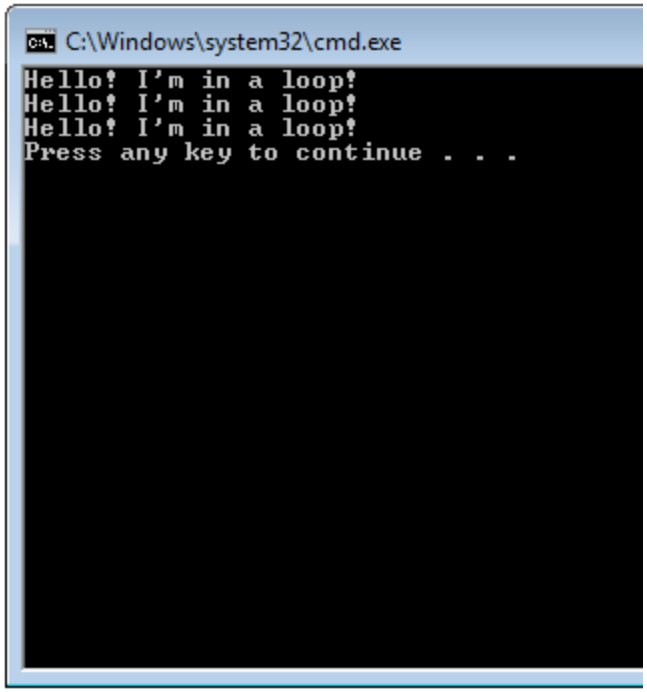



```
for(int i=0; i<3; i++)
    cout<<"Hello! I'm in a loop!\n";
```

B. Iteration: for Loop (cont.)

Example1

```
1  //
2  #include <iostream>
3  using namespace std;
4
5  void main ()
6  {
7      // variables
8
9      // body
10     for(int i=0; i<3; i++)
11         cout<<"Hello! I'm in a loop!\n";
12
13     // output
14
15 }
```



B. Iteration: for Loop (cont.)

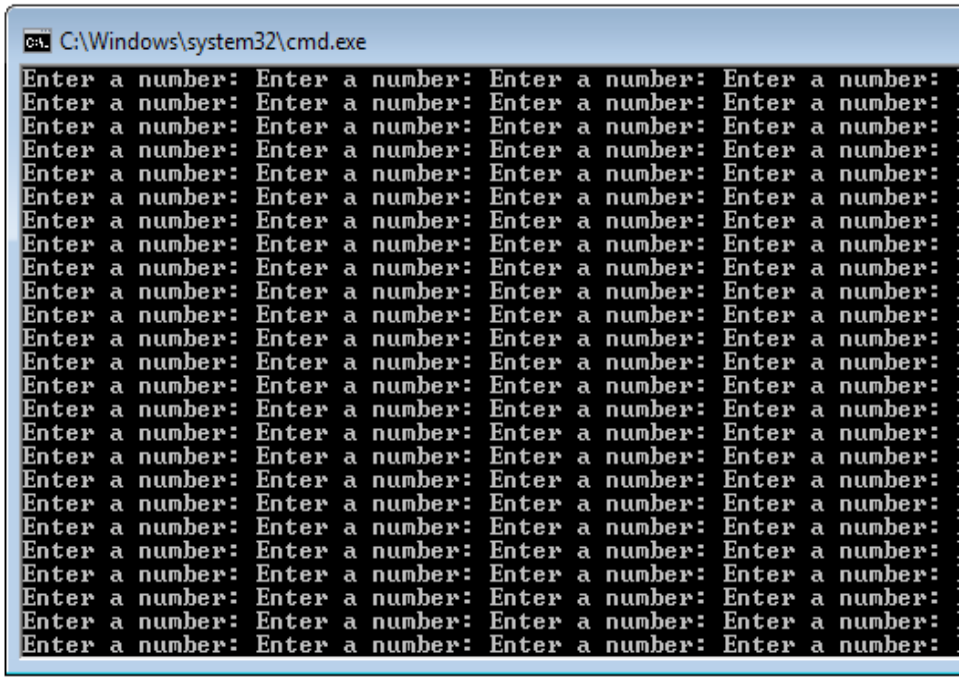
Remark

Index update (IN LOOP HEAD) MUST change the result of the bool_exp so we can exit the loop at some point of execution, otherwise you will go into an <u>infinite loop</u>… something very bad!

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

void main ()
{
    // variables
    int num = 0;
    // body
    for(int i=0; i<3; num++)
        cout<<"Enter a number: ";
        cin>>num;
        cout<<"square is: "<<num*num<<endl;

    // output
```



B. Iteration: for Loop (cont.)

Example2

```
1  //
2  #include <iostream>
3  using namespace std;
4
5  void main ()
6  {
7      // variables
8      int num;
9      // body
10     for(int i=0; i<3; i++)
11         cout<<"Enter a number: ";
12         cin>>num;
13         cout<<"square is: "<<num*num<<endl;
14
15     // output
16
17 }
18
```

Find the mistake!

C:\Windows\system32\cmd.exe

```
Enter a number: Enter a number: Enter a number: 1
square is: 1
Press any key to continue . . .
```

B. Iteration: for Loop (cont.)

Example3

```
1  //
2  #include <iostream>
3  using namespace std;
4
5  void main ()
6  {
7      // variables
8      int num;
9      // body
10     for(int i=0; i<3; i++)
11     {
12         cout<<"Enter a number: ";
13         cin>>num;
14         cout<<"square is: "<<num*num<<endl;
15     }
16
17     // output
18
```

This is a compound body.

C:\Windows\system32\cmd.exe

```
Enter a number: 1
square is: 1
Enter a number: 2
square is: 4
Enter a number: 3
square is: 9
Press any key to continue . . .
```

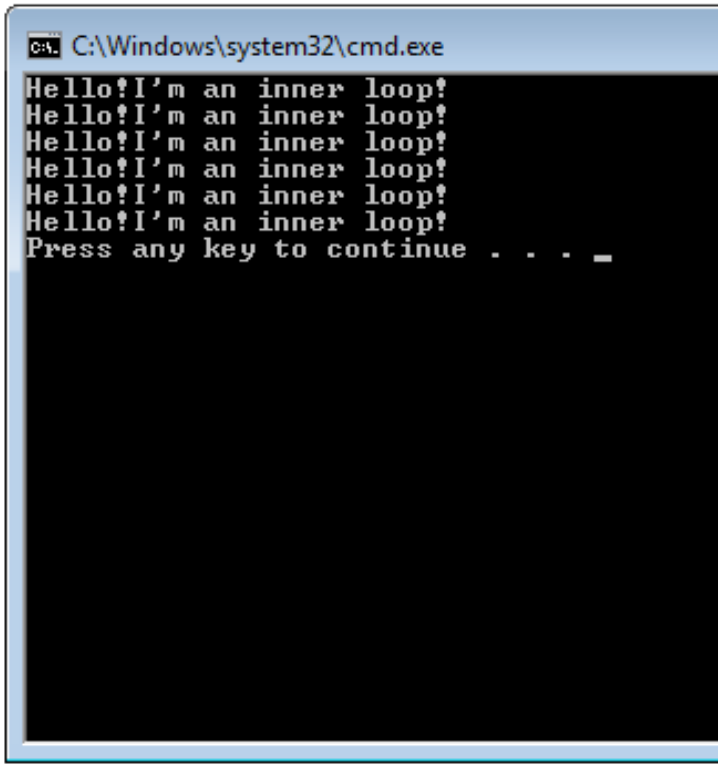
B. Iteration: for Loop (cont.)

Nested for Loop

```
for(start, bool_expression, action)
        for(start, bool_expression, action)
                statement;
```

Example4

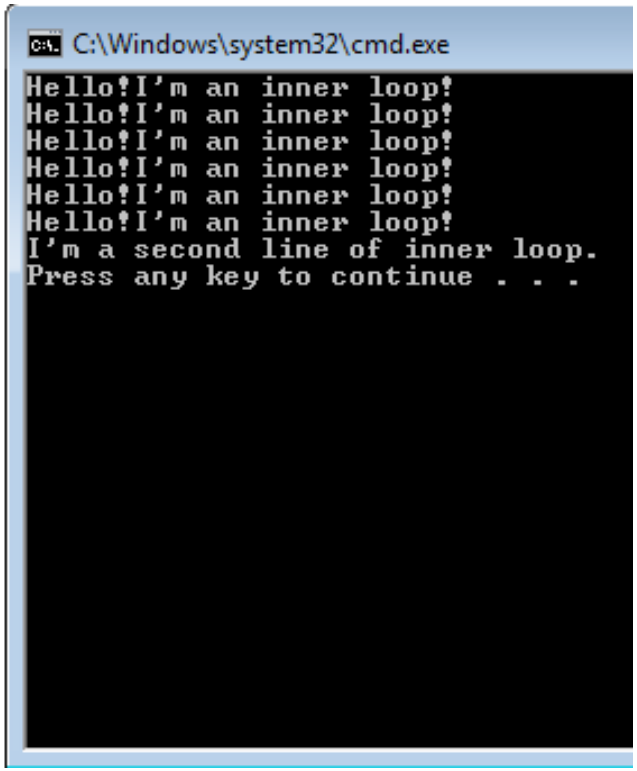
```
1  //
2  #include <iostream>
3  using namespace std;
4
5  void main ()
6  {
7      // variables
8      int num = 0;
9      // body
10     for(int i=0; i<3; i++)
11         for(int j=0; j<2; j++)
12             cout<<"Hello!I'm an inner loop!\n";
13
14     // output
15
16 }
```



B. Iteration: for Loop (cont.)

Example5

```
1  //
2  #include <iostream>
3  using namespace std;
4
5  void main ()
6  {
7      // variables
8      int num = 0;
9      // body
10     for(int i=0; i<3; i++)
11         for(int j=0; j<2; j++)
12             cout<<"Hello!I'm an inner loop!\n";
13             cout<<"I'm a second line of inner loop.\n";
14     // output
15
16 }
```



This statement is simply just a statement after the loops and will execute sequentially.

B. Iteration: for Loop (cont.)

Example6

```
1  //
2  #include <iostream>
3  using namespace std;
4
5  void main ()
6  {
7      // variables
8      int num = 0;
9      // body
10     for(int i=0; i<3; i++)
11         for(int j=0; j<2; j++)
12         {
13             cout<<"Hello!I'm an inner loop!\n";
14             cout<<"I'm a second line of inner loop.\n";
15         } // end of inner loop
16     // output
17
```

C:\Windows\system32\cmd.exe

```
Hello!I'm an inner loop!
I'm a second line of inner loop.
Hello!I'm an inner loop!
I'm a second line of inner loop.
Hello!I'm an inner loop!
I'm a second line of inner loop.
Hello!I'm an inner loop!
I'm a second line of inner loop.
Hello!I'm an inner loop!
I'm a second line of inner loop.
Hello!I'm an inner loop!
I'm a second line of inner loop.
Press any key to continue . . .
```

B. Iteration: for Loop (cont.)

Example7

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

void main ()
{
    // variables
    int num = 0;
    // body
    for(int i=0; i<3; i++)
    {
        for(int j=0; j<2; j++)
            cout<<"Hello!I'm an inner loop!\n";
            cout<<"I'm a second line of inner loop.\n";
    } // end of outer loop
    // output
```

C:\Windows\system32\cmd.exe

```
Hello!I'm an inner loop!
Hello!I'm an inner loop!
I'm a second line of inner loop.
Hello!I'm an inner loop!
Hello!I'm an inner loop!
I'm a second line of inner loop.
Hello!I'm an inner loop!
Hello!I'm an inner loop!
I'm a second line of inner loop.
Press any key to continue . . .
```

B. Iteration: for Loop (cont.)

Remarks

```
for(int i=2; i<num; i=i+1)

for(int i=2; i<num; i+=2)

for(int i=20; i>num; i--)
```

B. Iteration: While Loop

```
while(bool_expression)
{       statement1;
        statement2;
}
```

```
int i=0;
while(i<=2)
{
    cout<<"Hello!\n";
    i++;
}
```

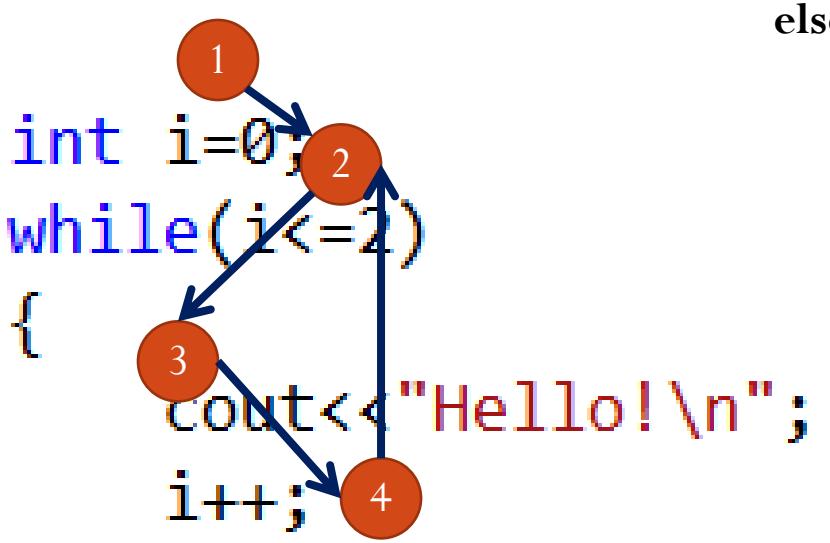
head: how many times to repeat body

body: part of code that is repeated

B. Iteration: While Loop (cont.)

```
while(bool_expression)
{       statement1;
        statement2;
}
```

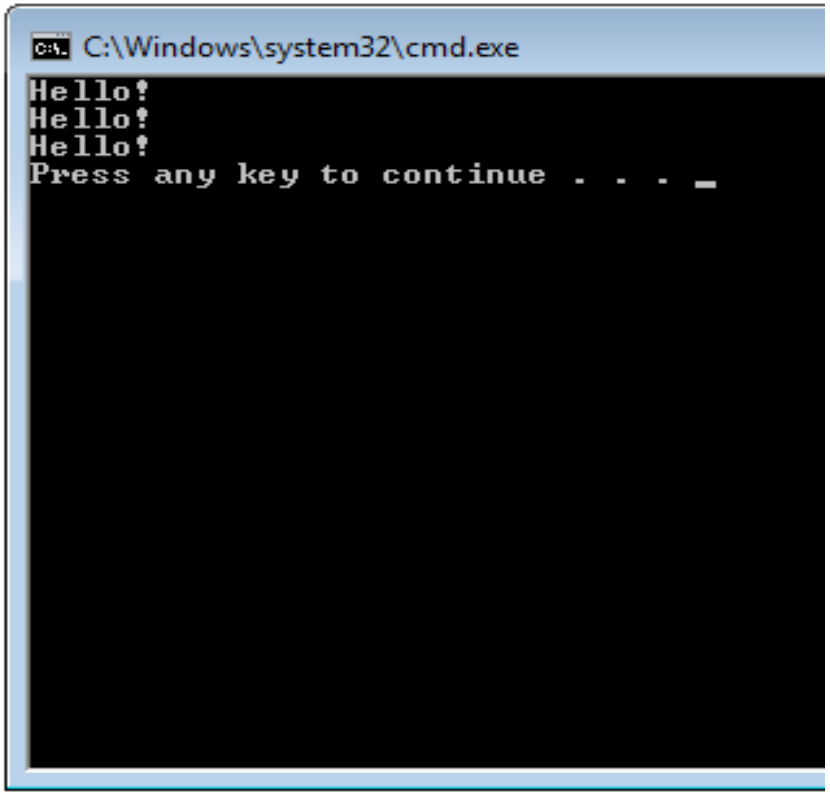
start count
if bool_exp is true
{ execute body
index update
re-evaluate bool_exp
}
else
exit the loop



B. Iteration: While Loop (cont.)

Example

```
6
7   #include <iostream>
8   using namespace std;
9
10
11  void main()
12  {
13      // variables
14      int i=0;
15      while(i<=2)
16      {
17          cout<<"Hello!\n";
18          i++;
19      }
20  } // end main
```



B. Iteration: While Loop (cont.)

Remark

Index update (INSIDE LOOP BODY) MUST change the result of the bool_exp so we can exit the loop at some point of execution, otherwise you will go into an <u>infinite loop</u>… something very bad!

```
7   #include <iostream>
8   using namespace std;
9
10
11  void main()
12  {
13      // variables
14      int i=0;
15      while(i<=2)
16      {
17          cout<<"Hello!\n";
18          i++;
19      }
    } // end main
```

B. Iteration: While Loop (cont.)

- The body of the while statement cannot be <u>simple.</u> It must be <u>compound</u>.

```
while(bool_expression)
{ statement1;
  statement2;
  statement3;
}
```

Block of code

WHY?
Because there must always be an extra statement to change the bool_exp

B. Iteration: Do..While Loop

```
do
{       statement1;
        statement2;
} while(bool_expression);
```


```
int i=0;
do
{
    cout<<"Hello!\n";
    i++;
} while(i<=2);
```

body: part of code that is repeated

head: how many times to repeat body

B. Iteration: Do..While Loop (cont.)

```
do
{       statement1;
        statement2;
} while(bool_expression);
```

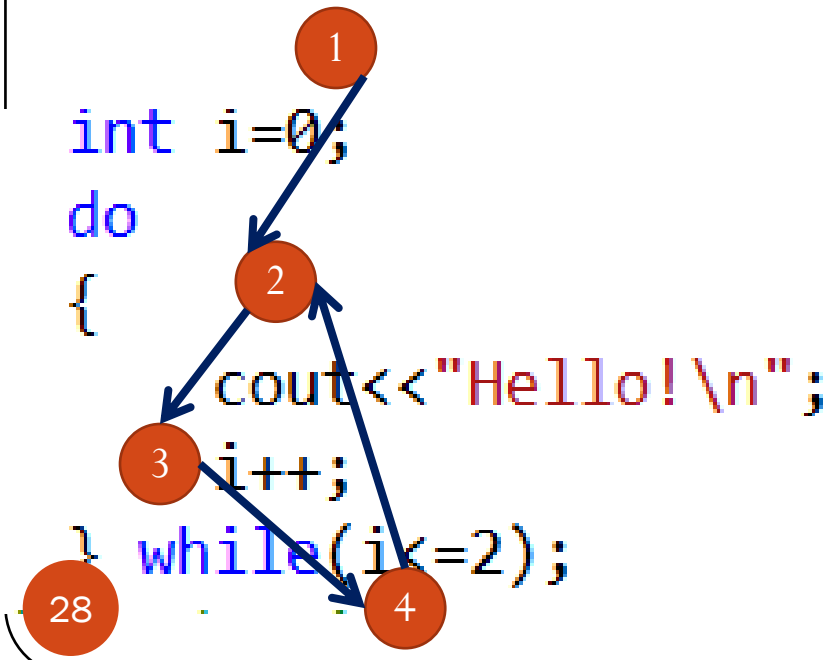
start count

{ execute body
index update
re-evaluate bool_exp

} if bool_exp is true

else

exit the loop



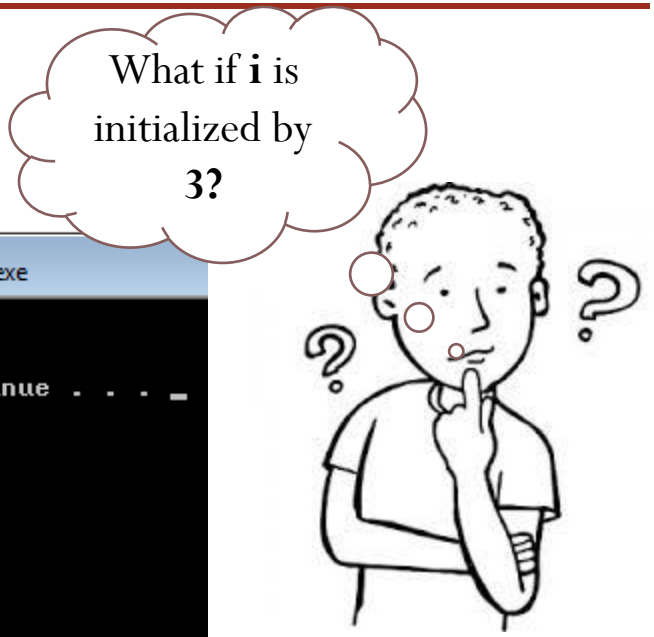
Compare to the while loop

```
int i=0;
while(i<=2)
{
    cout<<"Hello!\n";
    i++;
}
```

B. Iteration: Do..While Loop (cont.)

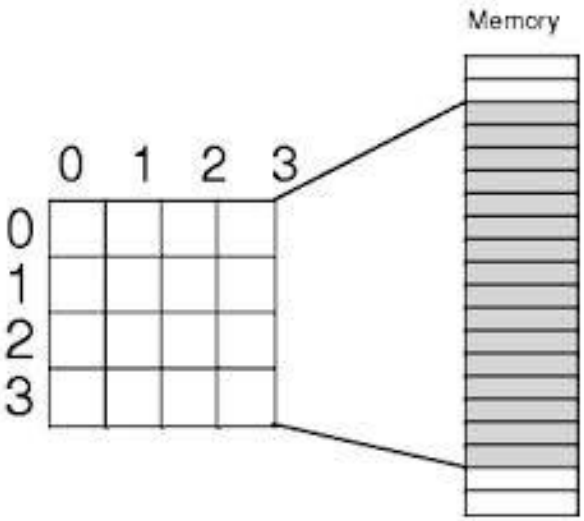
Example1:

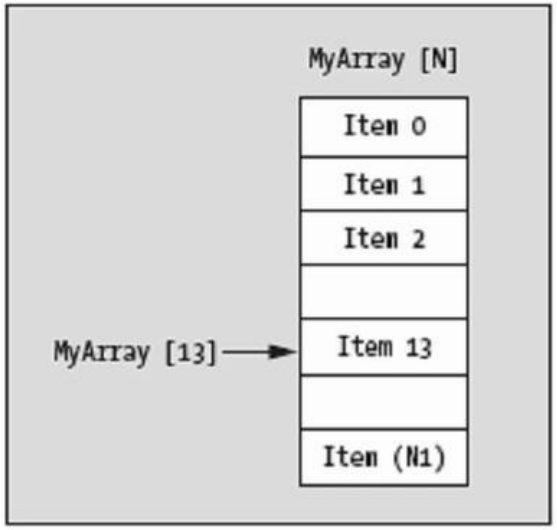
```
7   #include <iostream>
8   using namespace std;
9
10
11  void main()
12  {
13      // variables
14      int i=0;
15      do
16      {
17          cout<<"Hello!\n";
18          i++;
19      } while(i<=2);
20  } // end main
```



Break
(Refresh your minds in your place for <u>10</u> mints)

Arrays


Memory
0 1 2 3
0
1
2
3


MyArray [N]
Item 0
Item 1
Item 2
MyArray [13]
Item 13
Item (N1)

1. What is an Array? Why?

Arrays: collection of data of the *same type*.

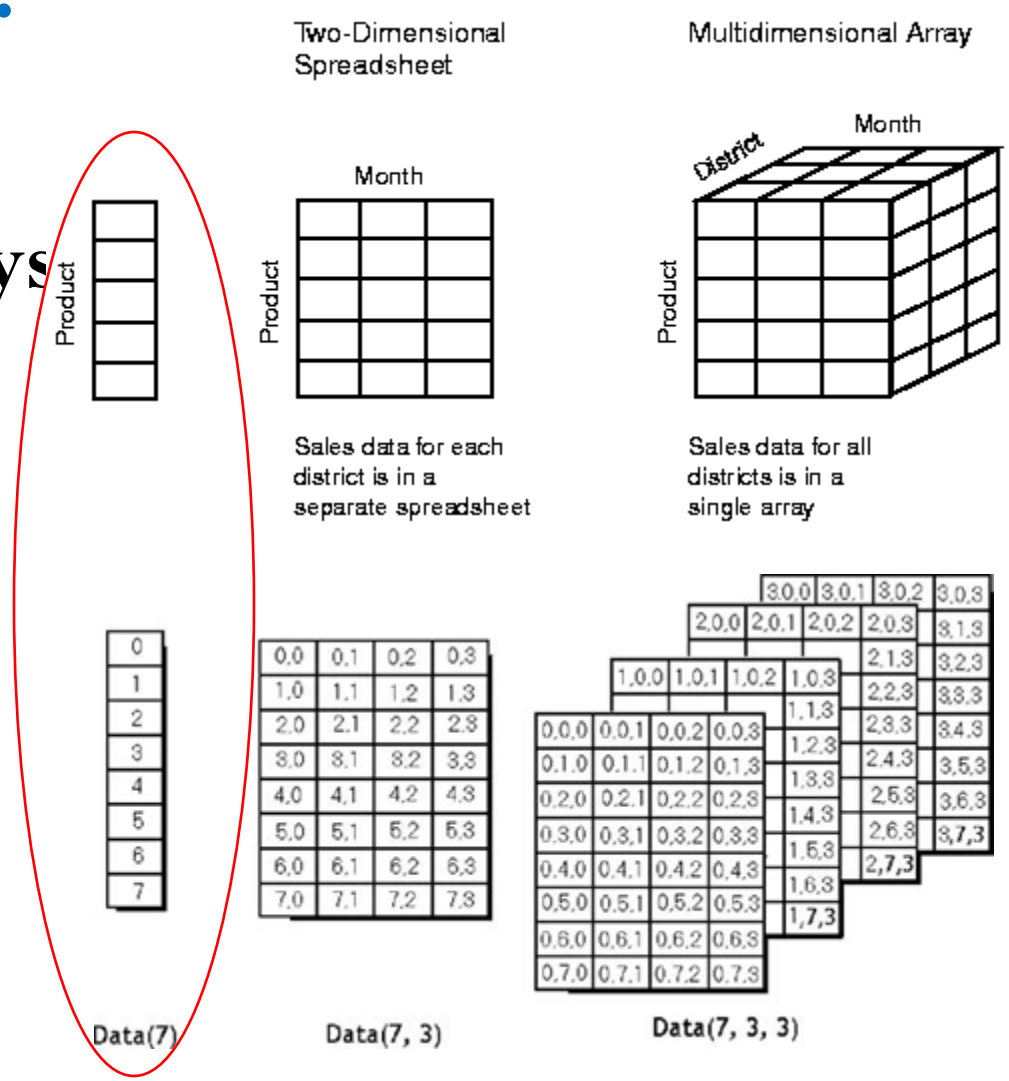
- **Called an "aggregate" data type.**
- **Enables you to manage several variables under one name.**
- **Adds organization and structure to your program.**
- **Examples**

{1, 2, 3} **{'a', 'b', 'p'}** **{3.5, 1.0, 5.6}**

1. What is an Array? Why? – (cont.)

What is a dimension?

one dimensional arrays



2. Declaration and Initialization

Declaration

type var_name[size]; → **allocates memory**

```
int score[5];
char choice[3];
float money[10];
```

Value or Constant or Expression → constant NOT variable

```
int size = 2;

int score[5+3];
char choice[3+size];
float money[size];
```

```
const int size = 2;

int score[5+3];
char choice[3+size];
float money[size];
```

2. Declaration and Initialization – (cont.)

Initialization

type var_name[size] = {value, value, …};

```
int score[5] = {1,2,3,4,5};

int score[5] = {1,2,3,4,5,6};
```

✖

```
int score[5] = {1,2};
```

Will zero the rest

- **What if undefined size? → Auto-initialization**

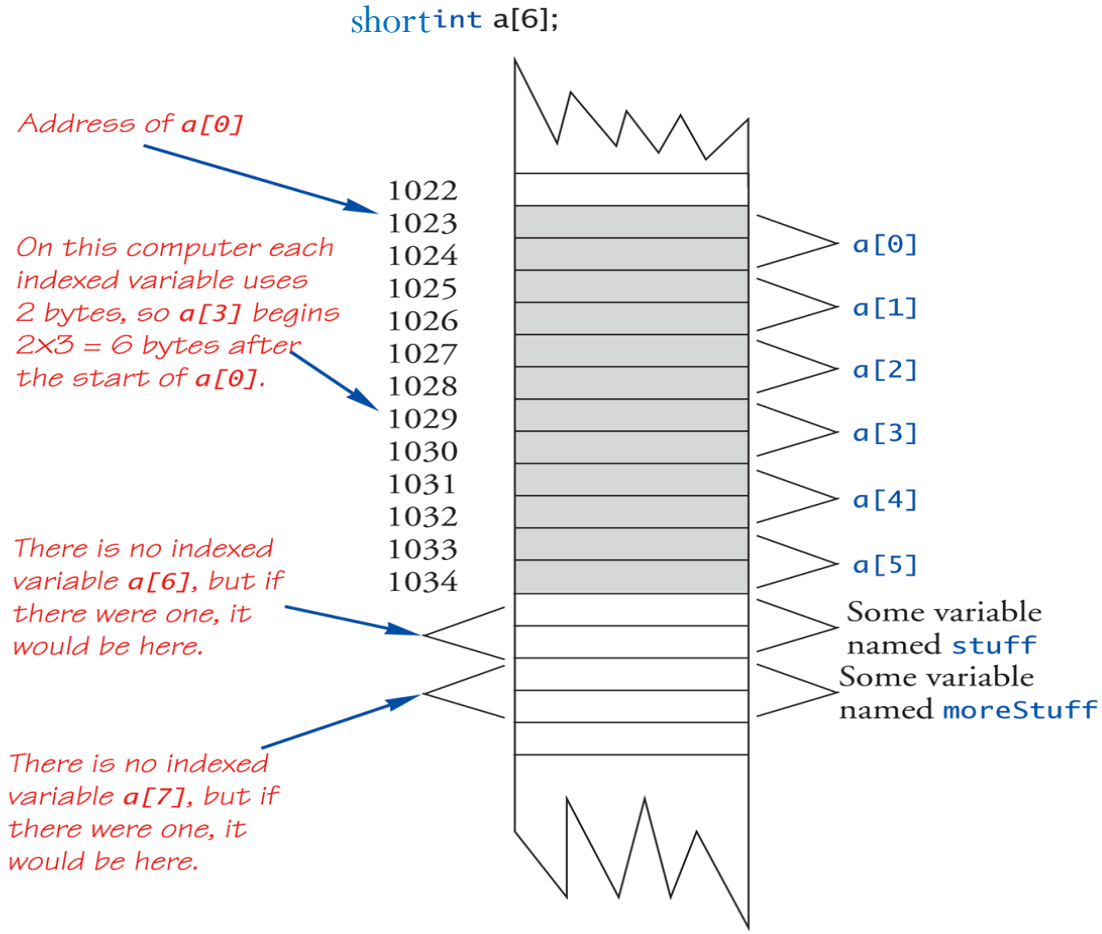
type var_name[] = {value, value, …};

```
int score[] = {1,2,3};
```

3. Arrays in Memory

- **Array elements are stored consecutively in memory.**

Display 5.2 An Array in Memory



4. Elements Referencing and Assignments

Accessing Array Elements

- *type name[size];*
- Element subscript
- **Index (subscript) starts at 0 to *size* -1.**
- **INDEX OUT OF RANGE ERROR.**

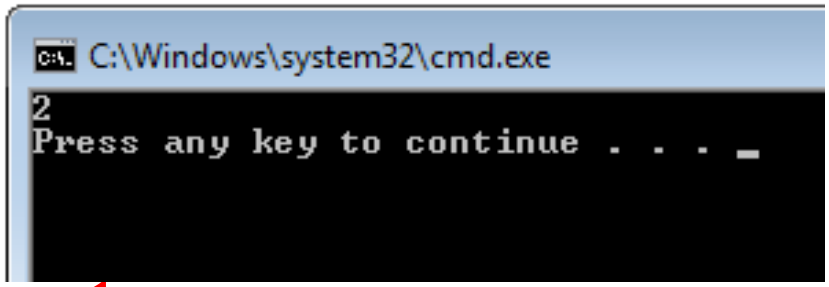
Any idea WHY ZERO-BASED?

4. Elements Referencing and Assignments

Elements Referencing (Accessing)

type var_name[size]; → Subscript or index

```
int score[] = {1,2,3};
cout<<score[1]<<endl;
```



- **Index starts at 0 to *size*-1.**
- **So in the above example we have**
  - Score[0] = 1
  - Score[1] = 2
  - Score[2] = 3

4. Elements Referencing and Assignments

Assignments

var_name[index] = expression;

Of the same base type of the array

```
int x;
int score[3];
score[0] = 5;
cin>>x;
score[1] = x;
score[2] = x-1;
```

- **Remember that you cannot go beyond range**

```
int score[2] = {0};

score[7] = 4;
```

5. Processing an Array

- **Let's define an array of 10 scores and fill its elements from the user, and display their average.**

```
int score[10] = {0}; // initialize all elements to zero

// input
cout<<"Please enter a score: "; cin>>score[0];
cout<<"Please enter a score: "; cin>>score[1];
cout<<"Please enter a score: "; cin>>score[2];
cout<<"Please enter a score: "; cin>>score[3];
cout<<"Please enter a score: "; cin>>score[4];
cout<<"Please enter a score: "; cin>>score[5];
cout<<"Please enter a score: "; cin>>score[6];
cout<<"Please enter a score: "; cin>>score[7];
cout<<"Please enter a score: "; cin>>score[8];
cout<<"Please enter a score: "; cin>>score[9];
// compute average
double avg = (score[0]+score[1]+score[2]+score[3]+score[4]+score[5]+score[6]
            +score[7]+score[8]+score[9])/10.0;
// output
cout<<"Average score: "<<avg<<endl;
```

Nonsense!

5. Processing an Array – (cont.)

- **To traverse or process elements of an array, a <u>loop</u> is required.**
- ***for* loops are preferred because we usually know the size of the array.**
- **Beware of INDEX OUT OF RANGE ERROR.**

5. Processing an Array – (cont.)

- **Let's define and array of 10 scores and fill its elements from the user, and display their average.**

```
int score[10] = {0}; // initialize all elements to zero

// input
cout<<"Please enter a score: "; cin>>score[0];
cout<<"Please enter a score: "; cin>>score[1];
cout<<"Please enter a score: "; cin>>score[2];
cout<<"Please enter a score: "; cin>>score[3];
cout<<"Please enter a score: "; cin>>score[4];
cout<<"Please enter a score: "; cin>>score[5];
cout<<"Please enter a score: "; cin>>score[6];
cout<<"Please enter a score: "; cin>>score[7];
cout<<"Please enter a score: "; cin>>score[8];
cout<<"Please enter a score: "; cin>>score[9];
// compute average
double avg = (score[0]+score[1]+score[2]+score[3]+score[4]+score[5]+score[6]
            +score[7]+score[8]+score[9])/10.0;
// output
cout<<"Average score: "<<avg<<endl;
```

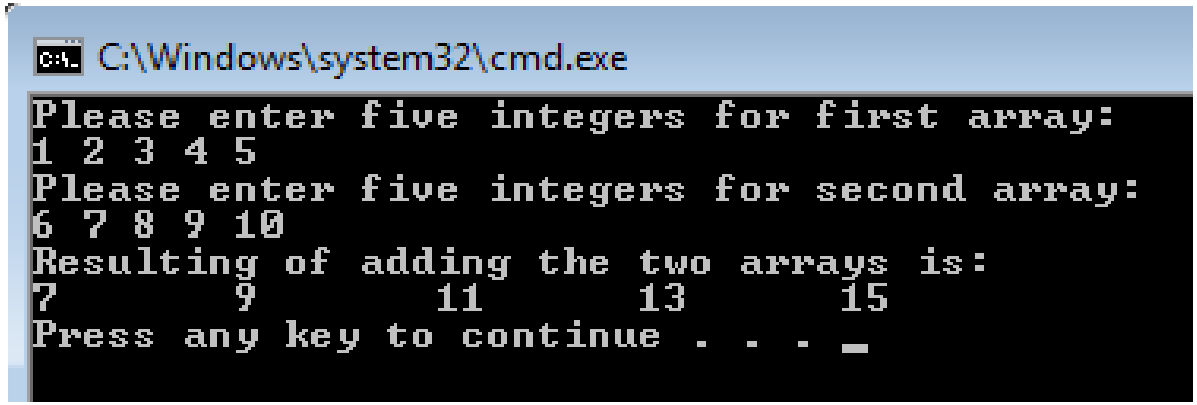
```
// input
for(int i=0; i<10; i++)
{
    cout<<"Please enter a score: ";
    cin>>score[i];
}
 // compute average
 int sum = 0;
 for(int i=0; i<10; i++)
     sum += score[i];
 double avg = sum/10.0;
```

6. Examples

Example 1

- **Write a program to accept two arrays, each of five integers, from the user and displays the sum of them.**

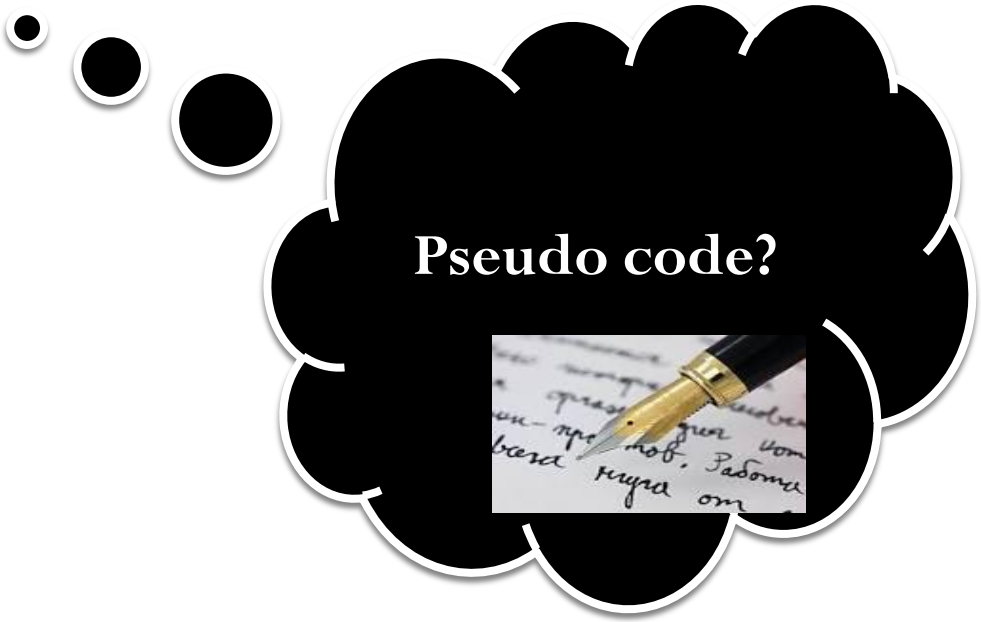
<u>Note</u> that the sum of two arrays is an array whose elements are the sum of corresponding input elements.



6. Examples – (cont.)

Example 2 (Linear Search)

- **Write a program to search a list of integers for a given number and display its location if found.**



6. Examples – (cont.)

Example 2 (flag vs. end of loop)

Linear Search

Pseudocode for linear search (version 1)

```
1.    for each element in the array
2.      if element = target value then
          display position
          set found flag to true.
3.      next element
4.    If found flag is not true, then display "not
      found" message
```

6. Examples – (cont.)

SMART ENOUGH..HA ? ☺

Example 3 (*Class BONUS*)

- **Write a program to accept an integer from the user and display each digit on a separate line.**

```
C:\Windows\system32\cmd.exe
Enter a number: 12345
Method 1 output:
5
4
3
2
1

Method 2 output:
1
2
3
4
5
```

6. Examples – (cont.)

Example 4

- **Write a program to take ten integers from the user and display the maximum and minimum of them. (initial value vs. first element)**

```
C:\Windows\system32\cmd.exe
Please enter ten numbers: 9 5 -9 4 2 4 10 45 -84 55
Maximum: 55
Minimum: -84
Press any key to continue . . .
```

6. Examples – (cont.)

Example 5 (*Take-HOME-Exercise*)

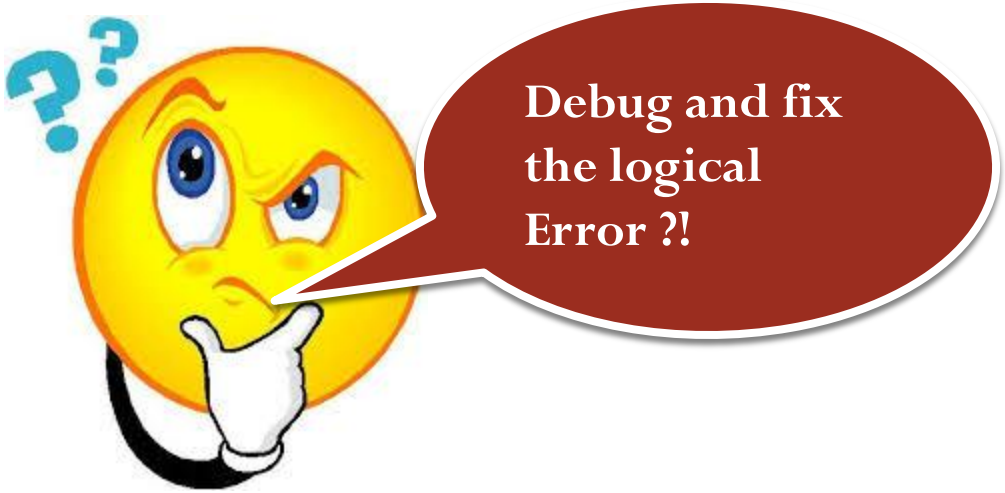
- **Write a program to find the second maximum of an array of 5 integers.**

(Sort Vs. Search)

Class Accumulative Project: Employees Salary for Companies

Class Accumulative Project: Employees Salary for Companies

- Refer to Bonus Example in Dropbox folder under folder Lecture1.
- View code



Class Accumulative Project: Employees Salary for Companies

Task1: Compute Net Salary for **AN** employee taking as an input the number of hours worked as well as the dependents. **(DONE☺)**

TASK 2 (NEW* BONUS):

- Update Code to Calculate Net Pay Salaries and taxes for 10 employees in your company.
- Submit your code as text in this form, **due Date Monday 19/2/2018**

https://docs.google.com/forms/d/1mVSLy3Czx3ZBXzwGH3ung2Updwpk0VIqZwgzDvR0Xds/edit?usp=sharing

View code

Over thinking ☺

(What if we take number of employees as an input from user?)

Thank You