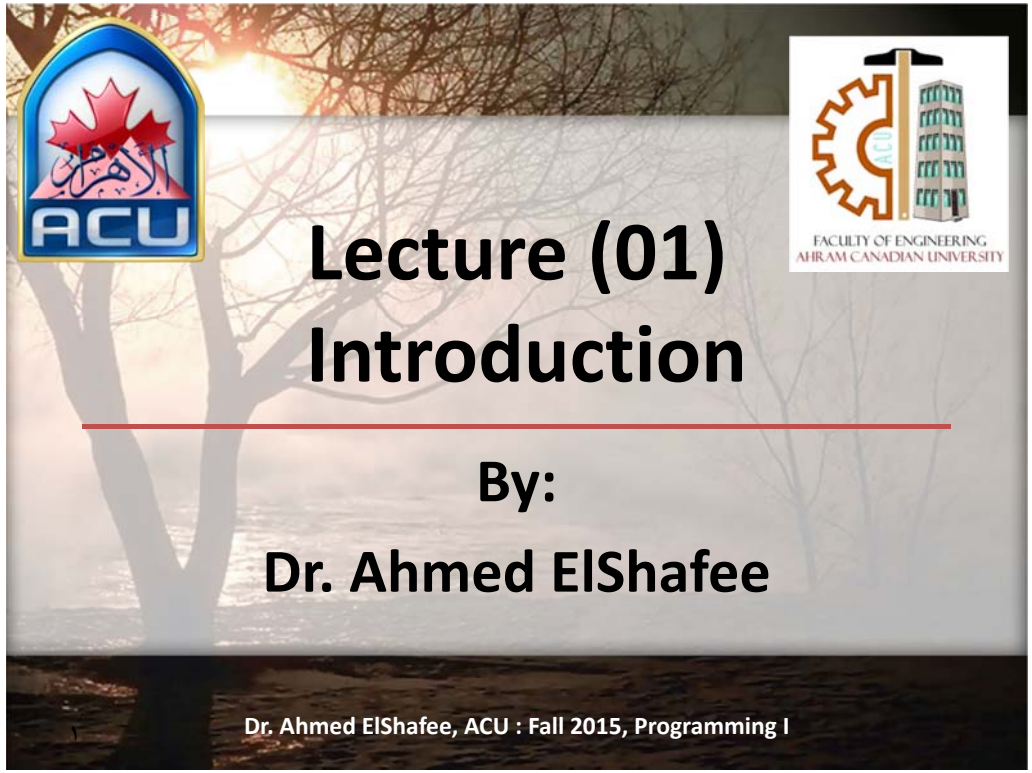


C language

- C language was developed by “Dennis Richie” on DEC-PDP-11
- The OS od such computer was UNIX os
- Following table show language history

Language	Developer	Year	Language type
CPL	London collage	1960	Low level
BCPL	Martin Richards	1967	Low level
B	Thomson	-	High level
C	Dinnis Richie	1970	High level
C++	Bjarne Stroustrup	1979	High level

- so C is a high level language based on low level language, so it combines advantages of both low level and high level language
- C is the most popular programming language, that used for applications and OSs development.

- C used built for UNIX OS that used PDP-11 PC.
- So the early version of C wasn’t compatible with other OS
- So it was expected that different version of C will appears (like Basic) each release developed for certain processor and certain OS
- In 1983 ANSI organization developed a standard version of C language called “ANSI C” to achieve operability between different OSs and different machines
- While each machine and OS have their own compiler. So an ANSI C program that developed on Linux OS can run on windows OS if a suitable compiler is used.
- In 1989 the first ANSI C compiler appeared
- In 1990 the first ANSI C/C++ compiler appeared

C language advantages

- Structed language
- Bit maipulation language
- Fast
- Standard “ANSI C”
- UNIX OS default programming language
- Operable
- Portable
- GUI enabled

C++ language

- Coder can not distinguish between code written in C and code written in C ++ because of the integration between the two languages
- Simply C++ is additive features and instruction to original ANSI C to enhance program functionality and code reuse

- Difference between C and C++

1. IO stream
2. Variables overloading
3. Objects and classes

- The main reason for the development of C ++ is the new trend of object-oriented programming and the need for it in building windows OS
- desktop windows application, is mainly consists of different objects like buttons, menu, check box, etc

General output function; my first program

Function name: printf()

Liberary: stdio.h

- To print string

```
Printf(“Hello world”);
```

- Using escape parameters:

symbol	meaning
\n	New line
\t	Tab
\a	Beeb
\b	Back space
\x	Hexadecimal
\\	\
\?	?
\’	‘
\”	“

Example 01

```
(Global Scope)
// Example01.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"

#include <stdio.h>

int _tmain(int argc, _TCHAR* argv[])
{
    printf("\n Hello \tworld");
    printf("\n Hello again");
    printf("\n");
    printf("\n\aline is empty\n");
    printf("\npress enter to exit");
    getchar();
    return 0;
}
```


Semicolons & Blocks in C++:

- In C++, the semicolon is a statement terminator. That is, each individual statement must be ended with a semicolon. It indicates the end of one logical entity.
- For example, following are three different statements:

```
printf("hello world\n");
return 0;
```

- A block is a set of logically connected statements that are surrounded by opening and closing braces. For example:

```
int main(int argc, char** argv) {
        printf("hello world\n");
        return 0;
}
```


- C++ does not recognize the end of the line as a terminator. For this reason, it does not matter where on a line you put a statement. For example

```
printf("hello world\n");
return 0;
```

- is the same as

```
printf("hello world\n"); return 0;
```


Whitespace in C++:

- Whitespace separates one part of a statement from another and enables the compiler to identify where one element in a statement, such as int, ends and the next element begins. Therefore, in the statement,

```
return 0;
```

```
#include <stdio.h>
```

comments

- helps anyone reading its source code.
- C++ supports single-line and multi-line comments. All characters available inside any comment are ignored by C++ compiler.
- C++ comments start with /* and end with */. For example:

```
/* This is a comment */
/* C++ comments can also
* span multiple lines
 /*
```

- A comment can also start with //, extending to the end of the line. For example:

```
printf("hello world\n");// prints Hello World
```


Data Types

- While doing programming in any programming language, you need to use various variables to store various information.
- Variables are nothing but reserved memory locations to store values.
- This means that when you create a variable you reserve some space in memory .
- You may like to store information of various data types like character, wide character, integer, floating point, double floating point, boolean, etc.
- Based on the data type of a variable, the operating system allocates memory and decides what can be stored in the reserved memory.

- Primitive Built-in Types:
- C++ offer the programmer a rich assortment of built-in as well as user defined data types.
- Following table lists down seven basic C++ data types

Keyword	Type
bool	Boolean
char	Character
int	Integer
float	Floating point
double	Double floating point
void	Valueless
wchar_t	Wide character

- Several of the basic types can be modified using one or more of these type modifiers
  - signed
  - unsigned
  - short
  - long

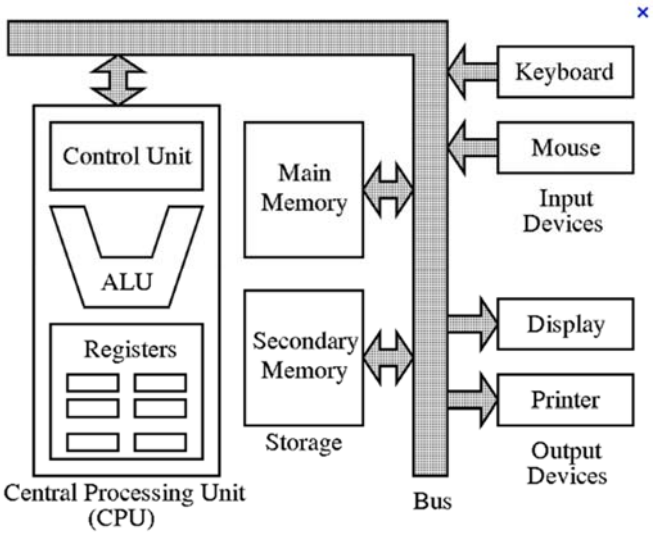
- The following table shows the variable type, how much memory it takes to store the value in memory, and what is maximum and minimum value, which can be stored in such type of variables.

Type	Typical Bit Width	Typical Range
char	1byte	-127 to 127 or 0 to 255
unsigned char	1byte	0 to 255
signed char	1byte	-127 to 127
int	4bytes	-2147483648 to 2147483647
unsigned int	4bytes	0 to 4294967295
signed int	4bytes	-2147483648 to 2147483647
short int	2bytes	-32768 to 32767
unsigned short int	2bytes	0 to 65,535
signed short int	2bytes	-32768 to 32767
long int	4bytes	-2,147,483,647 to 2,147,483,647
signed long int	4bytes	same as long int
unsigned long int	4bytes	0 to 4,294,967,295
float	4bytes	+/- 3.4e +/- 38 (~7 digits)
double	8bytes	+/- 1.7e +/- 308 (~15 digits)
long double	8bytes	+/- 1.7e +/- 308 (~15 digits)
wchar_t	2 or 4 bytes	1 wide character

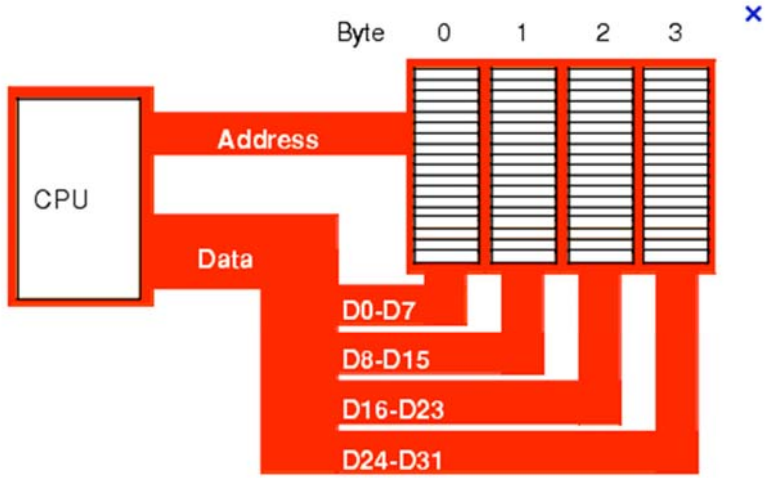
variables

- Programs work by manipulating data placed in memory.
- The data can be numbers, text, objects, pointers to other memory areas, and more besides.
- The data is given a name, so that it can be re-called whenever it is need.
- The name, and its value, is known as a Variable.

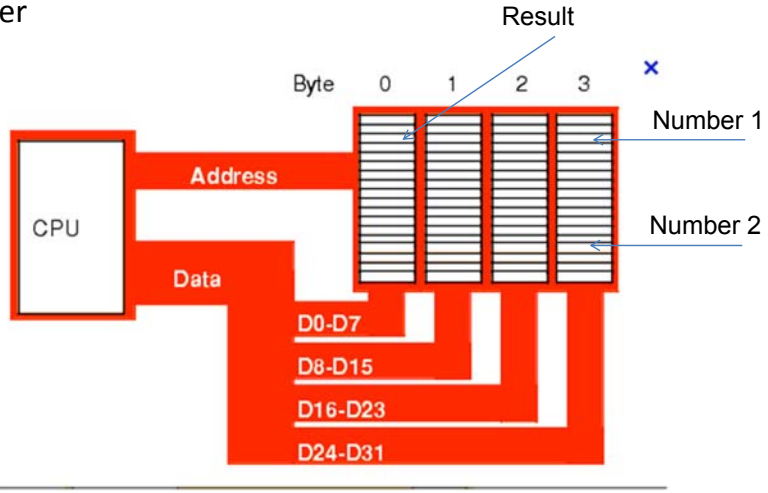
- Lets get back to computer architecture



- Now focus on the memory it self



- The idea of using variables in to use any available space in memory, and refer to it by unique name assigned the program developer



- So to tell C/C++ that you want to store a whole number, you first type the word **int,** followed by a space (32 bits).
- You then need to come up with a name for your integer variable.
- So to set up a whole number (integer),

int first_number;

C/C++ Identifiers:

- A C++ identifier is a name used to identify a variable, function, class, module, or any other user-defined item. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores, and digits (0 to 9).
- C++ does not allow punctuation characters such as @, $, and % within identifiers. C++ is a case-sensitive programming language. Thus, **Manpower** and **manpower** are two different identifiers in C++.
- Here are some examples of acceptable identifiers:

mohd	zara	abc	move_name	a_123	
myname50		_temp	j	a23b9	retVal

C++ Keywords:

- The following list shows the reserved words in C++. These reserved words may not be used as constant or variable or any other identifier names

this	new	else	Asm
throw	operator	enum	Auto
true	private	explicit	Bool
try	protected	export	Break
typedef	public	extern	Case
typeid	register	false	Catch
typename	reinterpret_cast	float	Char
union	return	for	Class
unsigned	short	friend	Const
using	signed	goto	const_cast
virtual	sizeof	If	continue
void	Static	inline	Default
volatile	static_cast	int	Delete
wchar_t	Struct	long	Do
while	Switch	mutable	Double
Template	namespace	dynamic_cast	

- To store something in the variable called first_number, you type an equals sign and then the value you want to store

int first_number;

first_number = 10;

- If you prefer, you can do all this on one line

int first_number = 10;

- Let's try some simple addition.
- Add two more int variables to your code, one to store a second number, and one to store an answer:

int first_number, second_number, answer;

- Notice how we have three variable names on the same line.
- You can do this in Java, if the variables are of the same type (the **int type, for us).**
- Each variable name is then separated by a comma.
- We can then store something in the new variables:

first_number = 10;

second_number = 20;

answer = first_number + second_number;

- You can also use numbers directly.
- Change the **answer line to this:**

answer = first_number + second_number + 12;

- Create FirstProject04 example, to find the answer.
- You can store quite large numbers in the **int variable type. The maximum value is** 2147483647 (32 bits).
  (0111 1111 1111 1111 1111 1111 1111 1111)
- If you want a minus number the lowest value you can have is -2147483648
  (1000 0000 0000 0000 0000 0000 0000 0000)

- C/C++ deals with string as an array of characters.

```
char name[] = =“ahmed elshafee”;
```

```
char name[20] = =“ahmed elshafee”;
```


Printing variables

- To print a value stored in a variable, printf should know it type, using one of the following syntax

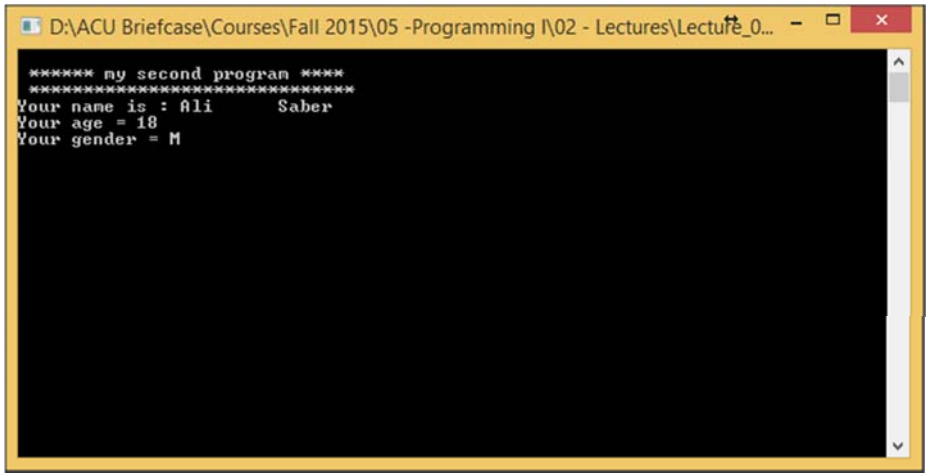
%d	Integer values
%c	Characters
%s	String values
%f	Float values
%x	Hexadecimal
%o	octal

```
int x;
x = 10;
Printf(“x = %d”,x);
```

```
Char name[] = “Ali Saber”;
Printf(“name is:  %s”,name);
```


Example 02

```
#include<stdio.h>

int _tmain(int argc, _TCHAR* argv[])
{
    char fname[10]="Ali";
    char lname[10]="Saber";
    int age=18;
    char gender='M';
    printf("\n ***** my second program ****");
    printf("\n ****************************");
    printf("\nYour name is : %s\t%s", fname,lname);
    printf("\nYour age = %d",age);
    printf("\nYour gender = %c",gender);
    getchar();
    return 0;
}
```

Formatting output

- To control the format of printed decimal numbers use the following examples

```
Printf("%3d",count);
```

- Prints 3 digits decimal number, adds padding blank space to the left

```
Printf("%03d",count);
```

- Prints 3 digits decimal number, adds padding zeros to the left

```
Printf("%5s",str);
```

- Prints string of five characters, , adds padding zeros to the left

```
Printf("%5s",str);
```

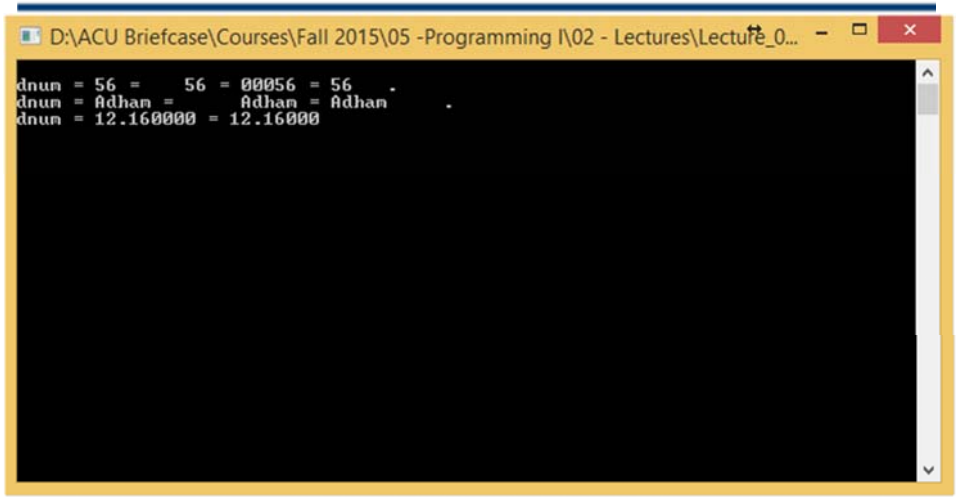
- Prints string of five characters, , adds padding zeros to the right


```
Printf("%.3f",fnum);
```

- prints 3 digits for float part (after float point) padding blank spaces to the right

Example03

```
// Example03.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include <stdio.h>

int _tmain(int argc, _TCHAR* argv[])
{
    int dnum=56;
    printf("\ndnum = %d = %5d = %05d = %-5d.",dnum,dnum,dnum,dnum);
    char name[]="Adham";
    printf("\ndnum = %s = %10s = %-10s.",name,name,name);
    float fnum=12.16;
    printf("\ndnum = %f = %.05f",fnum,fnum);
    getchar();
    return 0;
}
```

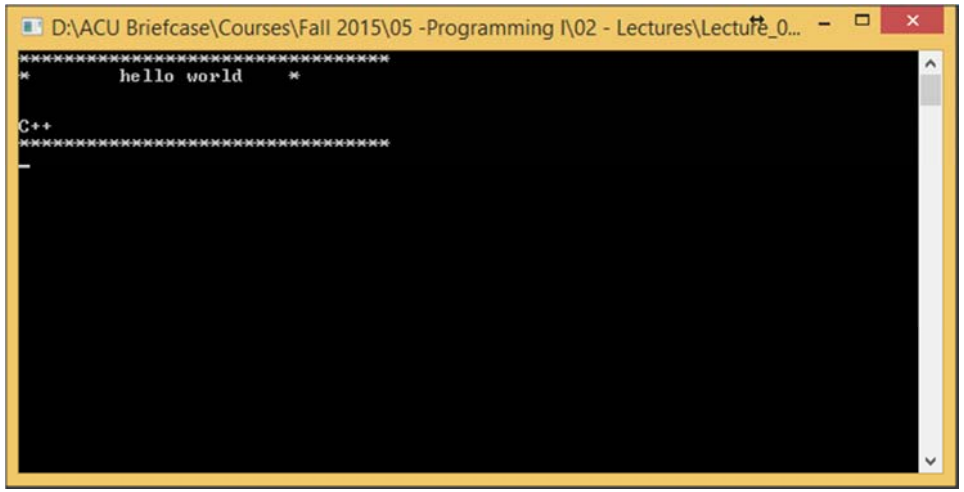
Other output functions

Function : puts(“string to print”);
Library: stdio.h

Function: putchar(‘a’);
Library: stdio.h

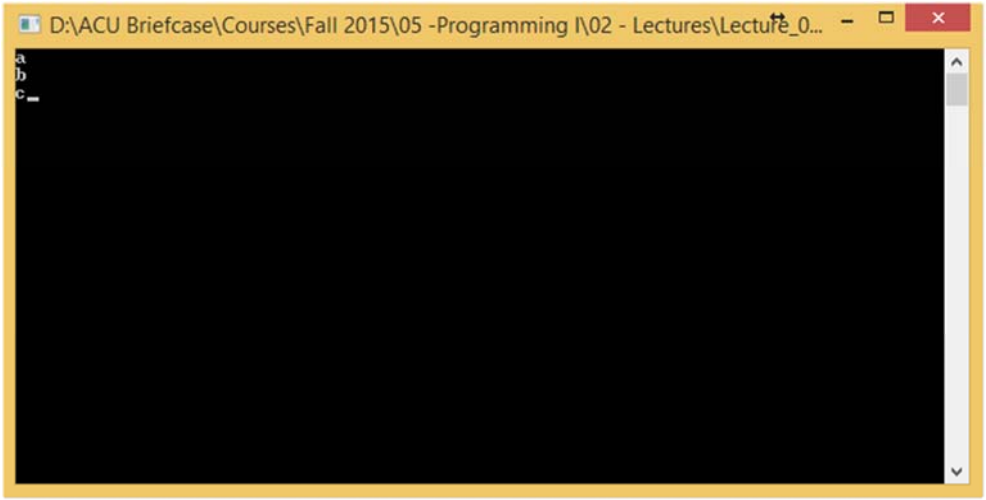
Example 04

```
// Example04.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include <stdio.h>

int _tmain(int argc, _TCHAR* argv[])
{
    char lang[]="C++";
    puts("**********************************");
    puts("*\t hello world\t*");
    puts("\n");puts(lang);
    puts("**********************************");
    getchar();
    return 0;
}
```

Example 05

```
// Example05.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include <stdio.h>

int _tmain(int argc, _TCHAR* argv[])
{
    char ch;
    ch='a';
    putchar(ch);
    putchar('\n');
    putchar('b');
    putchar('\n');
    putchar(99);
    getchar();
    return 0;
}
```



General input function

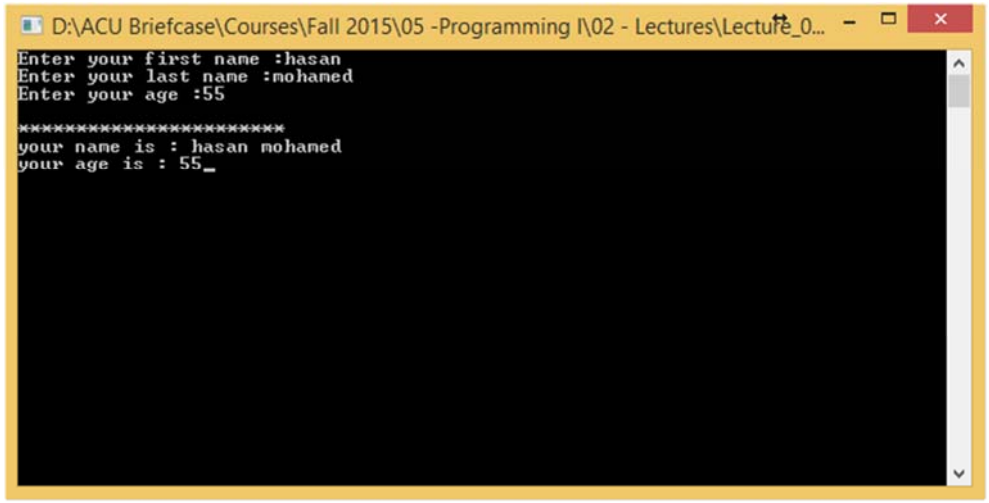
- Function: scanf();
- Library: stdio.h

```
Int num;
Scanf("%d",&num);
```

- Using & before variable name to refer for variable location.

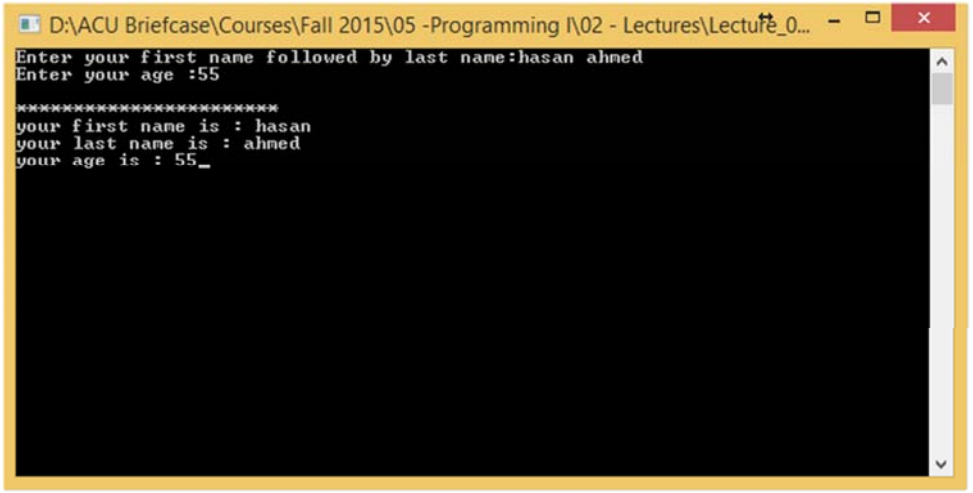
Example 06

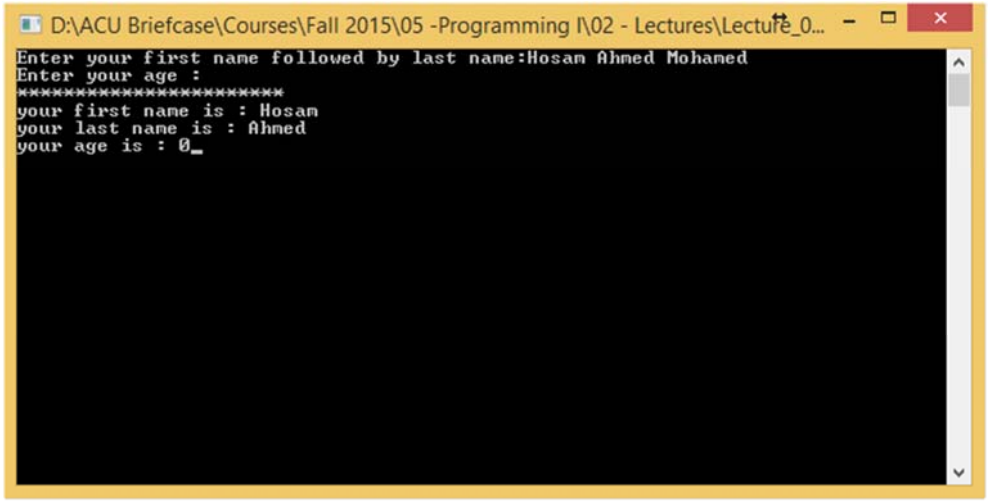
```
#include "stdafx.h"
#include "stdio.h"

int _tmain(int argc, _TCHAR* argv[])
{
    char fname[10],lname[10];
    int age=0;
    printf("Enter your first name :");
    scanf("%s",&fname);
    printf("Enter your last name :");
    scanf("%s",&lname);
    printf("Enter your age :");
    scanf("%d",&age);
    printf("\n**********************");
    printf("\nyour name is : %s %s",fname,lname);
    printf("\nyour age is : %d",age);
    fflush(stdin);
    getchar();
    return 0;
}
```



Example 07

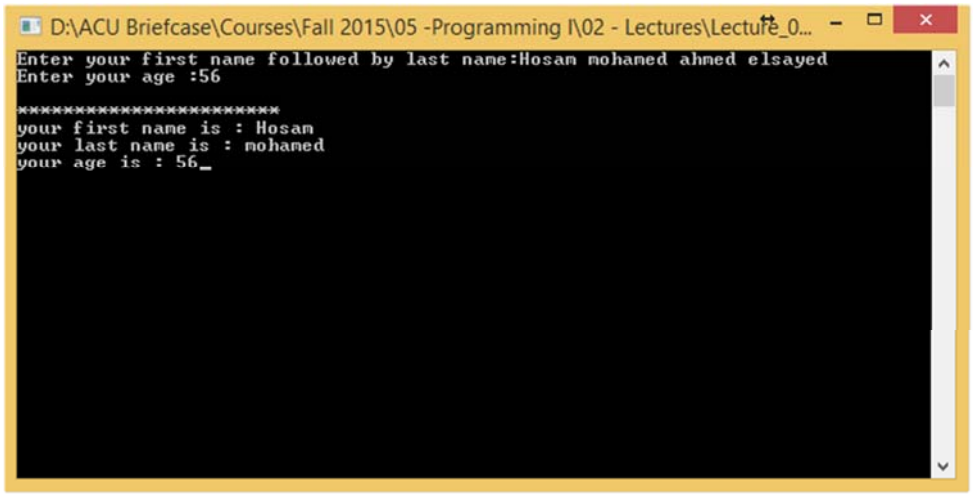
```c
#include "stdafx.h"
#include "stdio.h"

int _tmain(int argc, _TCHAR* argv[])
{
    char fname[10],lname[10];
    int age=0;
    printf("Enter your first name followed by last name:");
    scanf("%s %s",&fname,&lname);
    printf("Enter your age :");
    scanf("%d",&age);
    printf("\n**********************");
    printf("\nyour first name is : %s",fname);
    printf("\nyour last name is : %s",lname);
    printf("\nyour age is : %d",age);
    fflush(stdin);
    getchar();
    return 0;
}
```





Example 08

```c
#include "stdafx.h"
#include "stdio.h"

int _tmain(int argc, _TCHAR* argv[])
{
    char fname[10],lname[10];
    int age=0;
    printf("Enter your first name followed by last name:");
    scanf("%s %s",&fname,&lname);
    fflush(stdin);
    printf("Enter your age :");
    scanf("%d",&age);
    printf("\n**********************");
    printf("\nyour first name is : %s",fname);
    printf("\nyour last name is : %s",lname);
    printf("\nyour age is : %d",age);
    fflush(stdin);
    getchar();
    return 0;
}
```



Function : gets(string variable)

Library: stdio.h

```
char string[20];
gets(string);
```

Function : char variable=getchar()

char variable=getche()

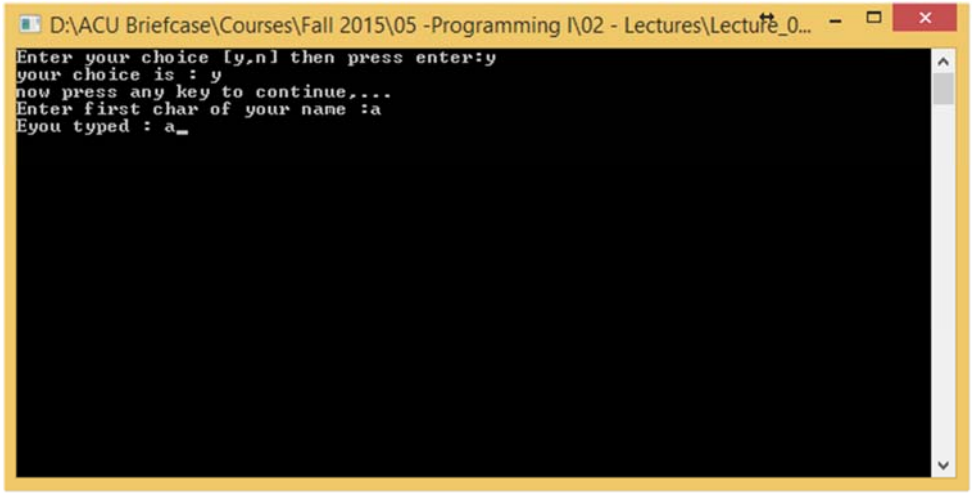
char variable=getch()

Library: stdio.h, conio.h, conio.h

```
char ch1=getchar();     // press enter to exceute next line
Char ch2=getche();      // show char no need to press enter
Char ch3=getch();       // doesn't show char no need to press enter
```


Example 09

```
// example09.cpp : Defines the entry point for the console appl
//

#include "stdafx.h"
#include <stdio.h>
#include <conio.h>
int _tmain(int argc, _TCHAR* argv[])
{
    char ch;
    printf("Enter your choice [y,n] then press enter:");
    ch=getchar();
    printf("your choice is : %c",ch);
    printf("\nnow press any key to continue,...");
    ch=getch();
    printf("\nEnter first char of your name :");
    ch=getche();
    printf("\nEyou typed : %c",ch);
    getche();
    return 0;
}
```

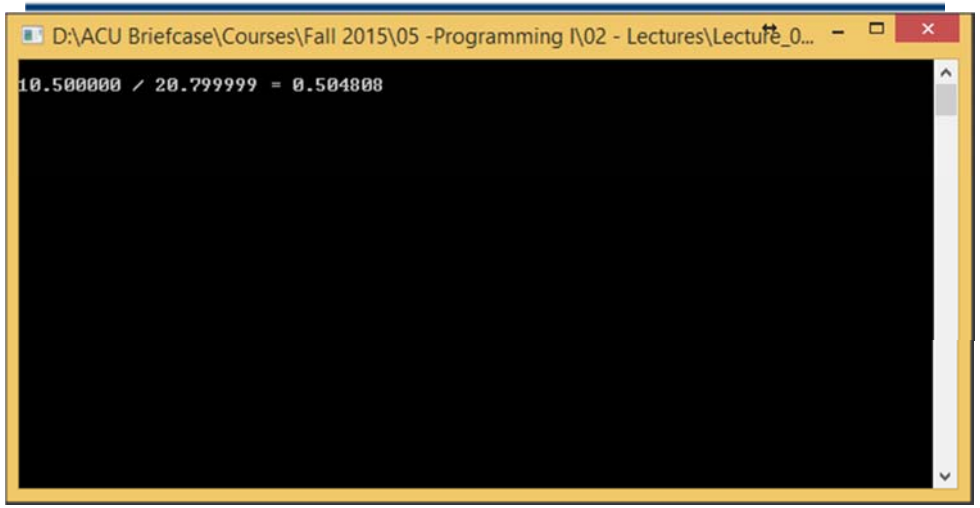



Simple arithmetic

- With the variables you've been using, you can use the following symbols to do
- calculations:
- + (the plus sign)
- - (the minus sign)
- * (the multiplication sign is the asterisk)
- / (the divide sign is the forward slash)

Example10

```
// Example10.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include <stdio.h>
#include <conio.h>

int _tmain(int argc, _TCHAR* argv[])
{
    float first_number, second_number, answer;
    first_number = 10.5f;
    second_number = 20.8f;
    answer = first_number / second_number;
    printf("\n%f / %f = %f",first_number,second_number,answer);
    getch();
    return 0;
}
```

Operator Precedence

- You can, of course, calculate using more than two numbers. But you need to take care of what exactly is being calculated. Take the following as an example:

first_number = 100;

second_number = 75;

third_number = 25;

answer = first_number – second_number + third_number;

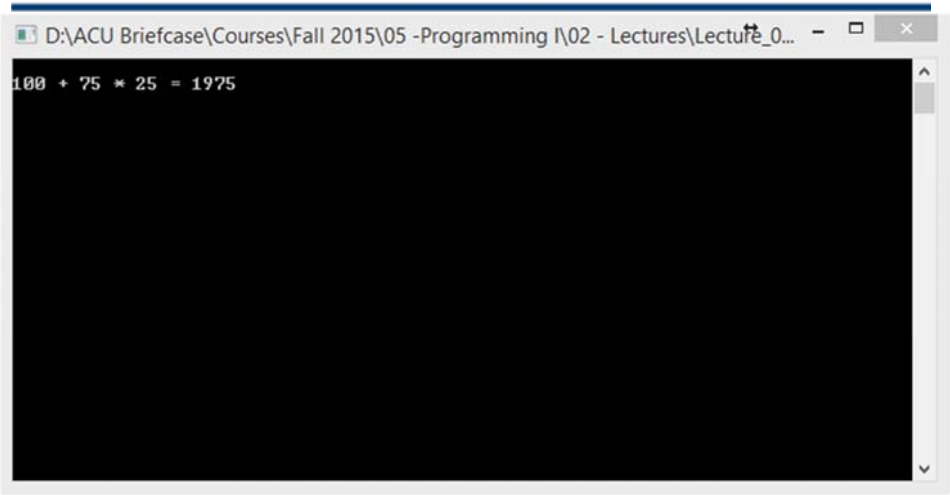
- If you did the calculation left to right it would be 100 – 75, which is 25. Then add the third number, which is 25. The total would be 50. However, what if you didn't mean that?

- What if you wanted to add the second and third numbers together, and then deduct the total from the first number? So 75 + 25, which is 100. Then deduct that from the first number, which is 100. The total would now be 0.
- To ensure that Java is doing what you want, you can use round brackets. So the first calculation would be:

answer = (first_number – second_number) + third_number;

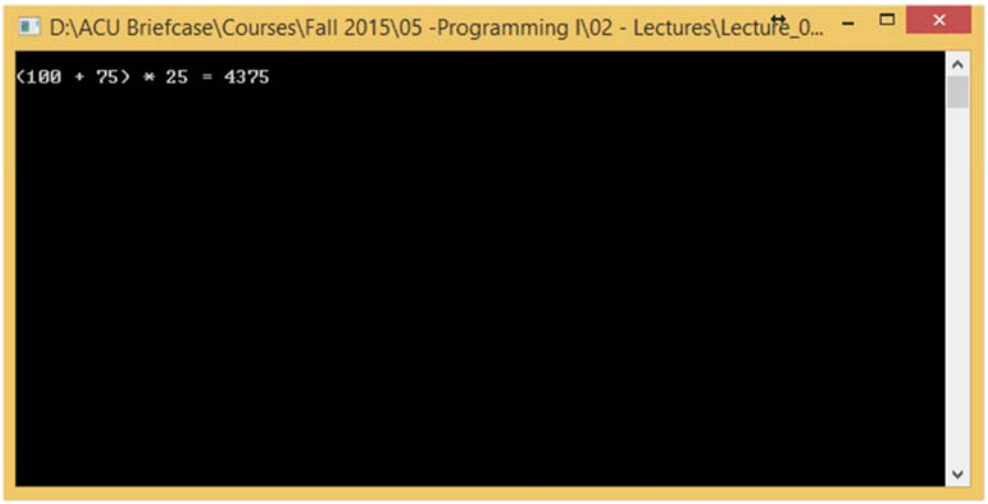
Example11

```
// Example11.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include <stdio.h>
#include <conio.h>

int _tmain(int argc, _TCHAR* argv[])
{
    int first_number, second_number, third_number,answer;
    first_number = 100;
    second_number = 75;
    third_number = 25;
    answer = first_number + second_number*third_number;
    printf("\n%d + %d * %d = %d",first_number,second_number,third_number,answer);
    getch();
    return 0;
}
```

Example12

```
// Example11.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include <stdio.h>
#include <conio.h>

int _tmain(int argc, _TCHAR* argv[])
{
    int first_number, second_number, third_number,answer;
    first_number = 100;
    second_number = 75;
    third_number = 25;
    answer = (first_number + second_number)*third_number;
    printf("\n(%d + %d) * %d = %d",first_number,second_number,third_number,answer);
    getch();
    return 0;
}
```


D:\ACU Briefcase\Courses\Fall 2015\05 -Programming I\02 - Lectures\Lecture_0...
(100 + 75) * 25 = 4375

Assignment


Thanks,..
See you next week (ISA),…
٦٣
Dr. Ahmed ElShafee, ACU : Fall 2015, Programming I