C++ - APPENDICES

APPENDIX A : KEYWORDS 2

APPENDIX B : OPERATORS 3

APPENDIX C : OPERATOR PRECEDENCE 4

APPENDIX D : ESCAPE SEQUENCES 5

APPENDIX E : ASCII CHARACTER SET 6

APPENDIX F : USING THE GCC COMPILER FOR C++ 7

APPENDIX G : RELATIONAL OPERATORS 8

APPENDIX H : LOGICAL OPERATORS 8

APPENDIX I : USING THE DEV C++ IDE 9

APPENDIX J : USING OPENGL & GLUT 16

APPENDIX K : SETTING THE SEARCH PATH 18

APPENDIX A : KEYWORDS

asm	do	if	return	typedef
auto	double	inline	short	typeid
bool	dynamic_cast	int	signed	typename
break	else	long	sizeof	union
case	enum	mutable	static	unsigned
catch	explicit	namespace	static_cast	using
char	export	new	struct	virtual
class	extern	operator	switch	void
const	false	private	template	volatile
const_cast	float	protected	this	wchar_t
continue	for	public	throw	while
default	friend	register	true	
delete	goto	reinterpret_cast	try	

APPENDIX B : OPERATORS

Arithmetic operators:	* / %	multiplication/division/modulus
	+ −	addition/subtraction
	+ −	positive/negative sign (unary)
	++ − −	increment/decrement (unary)
Logical operators:	&&	AND
	||	OR
	!	NOT (unary)
Relational operators:	< <= > >=	less than, less than or equal, greater than, greater than or equal
	= = !=	equal to and not equal to
Bit operators:	<< >>	left and right bit shift
	&	bitwise AND
	|	bitwise OR
	^	bitwise exclusive or XOR
	~	bitwise NOT (unary)
Assignment operators:	= += −= *= /= %= &= ^= |= <<= >>=	
Address/Pointer operators:	&	address of (unary)
	*	dereference (unary)
Structure operators:	.	structure member acccess
	–>	member access thru a structure pointer
	. *	member dereference
	->*	indirect member dereference
Memory allocation operators:	new, new[ ]	allocation, allocation of array
	delete, delete[ ]	de-allocation, de-allocation of array
Other operators:	::	scope resolution
	( )	function call
	[ ]	array access
	(*type*)	type cast (unary)
	sizeof	data object size in bytes (unary)
	?:	conditional operator
	throw	throw exception
	,	comma operator

APPENDIX C : OPERATOR PRECEDENCE

Operators	Associativity
::	
( ) [ ] –> .	left to right {( ) function call}
! ~ ++ – – + – * & (*type*) sizeof new new[ ] delete delete[ ]	right to left {All Unary}
. * –>*	left to right
* / %	left to right
+ –	left to right
<< >>	left to right
< <= > >=	left to right
= = !=	left to right
&	left to right
^	left to right
|	left to right
&&	left to right
||	left to right
?:	right to left
= += –= *= /= %= &= ^= |= <<= >>=	right to left
throw ,	left to right

APPENDIX D : ESCAPE SEQUENCES

Escape	value
\n	newline
\t	tab
\f	formfeed
\a	alarm
\b	backspace
\r	carriage return
\v	vertical tab
\\	backslash
\’	Single quote
\”	double quote

APPENDIX E : ASCII CHARACTER SET

0	(nul)	16	(dle)	32	(sp)	48	0	64	@	80	P	96	`	112	p
1	(soh)	17	(dc1)	33	!	49	1	65	A	81	Q	97	a	113	q
2	(stx)	18	(dc2)	34	"	50	2	66	B	82	R	98	b	114	r
3	(etx)	19	(dc3)	35	#	51	3	67	C	83	S	99	c	115	s
4	(eot)	20	(dc4)	36	$	52	4	68	D	84	T	100	d	116	t
5	(enq)	21	(nak)	37	%	53	5	69	E	85	U	101	e	117	u
6	(ack)	22	(syn)	38	&	54	6	70	F	86	V	102	f	118	v
7	(bel)	23	(etb)	39	‘	55	7	71	G	87	W	103	g	119	w
8	(bs)	24	(can)	40	(	56	8	72	H	88	X	104	h	120	x
9	(tab)	25	(em)	41	)	57	9	73	I	89	Y	105	I	121	y
10	(lf)	26	(eof)	42	*	58	:	74	J	90	Z	106	j	122	z
11	(vt)	27	(esc)	43	+	59	;	75	K	91	[	107	k	123	{
12	(np)	28	(fs)	44	,	60	<	76	L	92	\	108	l	124	|
13	(cr)	29	(gs)	45	-	61	=	77	M	93	]	109	m	125	}
14	(so)	30	(rs)	46	.	62	>	78	N	94	^	110	n	126	~
15	(si)	31	(us)	47	/	63	?	79	O	95	_	111	o	127	

APPENDIX F : USING THE GCC COMPILER FOR C++

- Create a file using your favorite text editor or retrieve some file containing C++ source code from somewhere.{By convention your file should end in *.cpp* or *.cc* and the gcc compiler requires it.}
- To compile:

```
g++  hello.cpp
```

compiles and links the file *hello.cpp* and produces a file called *a.exe* that can be executed

```
g++  -o  hello  hello.cpp
```

compiles and links the file *hello.cpp* and produces an executable file called *hello.exe*

```
g++  -c hello.cpp
```

compiles only, the file *hello.cpp* is compiled into an object module *hello.o*

```
g++  -Wall hello.cpp
```

compiles and links and displays warnings on all things that are somewhat questionable

- There is documentation available for the gcc compiler.
- The gcc compiler can cross-compile for many different processors.

APPENDIX G : RELATIONAL OPERATORS

- Relational operators test a relationship and produce a true/false result.

operator	Function
= =	Equality
<	less than
>	greater than
<=	less than or equal
>=	greater than or equal
!=	not equal

APPENDIX H : LOGICAL OPERATORS

- Logical operators work on logical values and produce a logical result.

operator	Function
&&	AND
||	OR
!	NOT

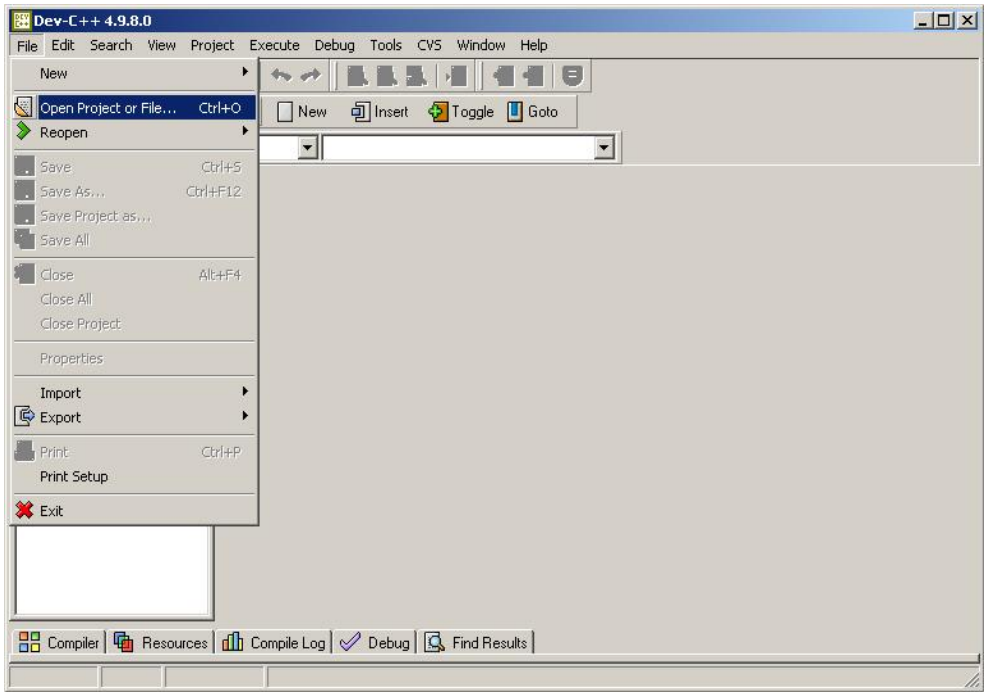
APPENDIX I : USING THE DEV C++ IDE

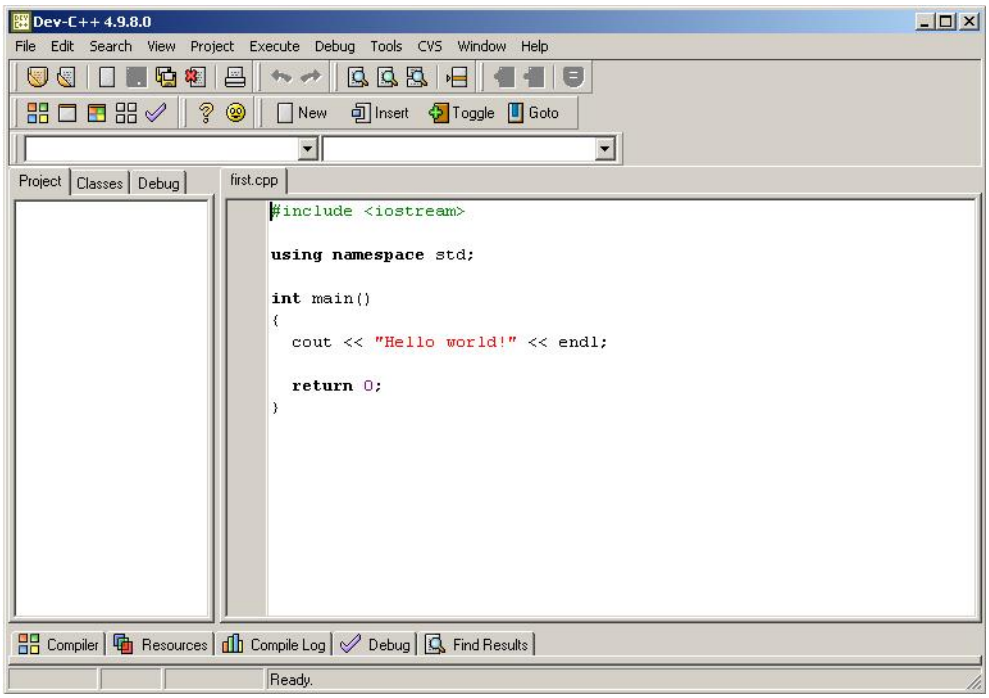
- The Dev-C++ Integrated Development Environment can be used to create, run and debug C programs on a PC.
- It has the following useful features:
    - It is available free of charge on the Web
    - It utilizes the gcc compiler

- Start Dev C++

cont…

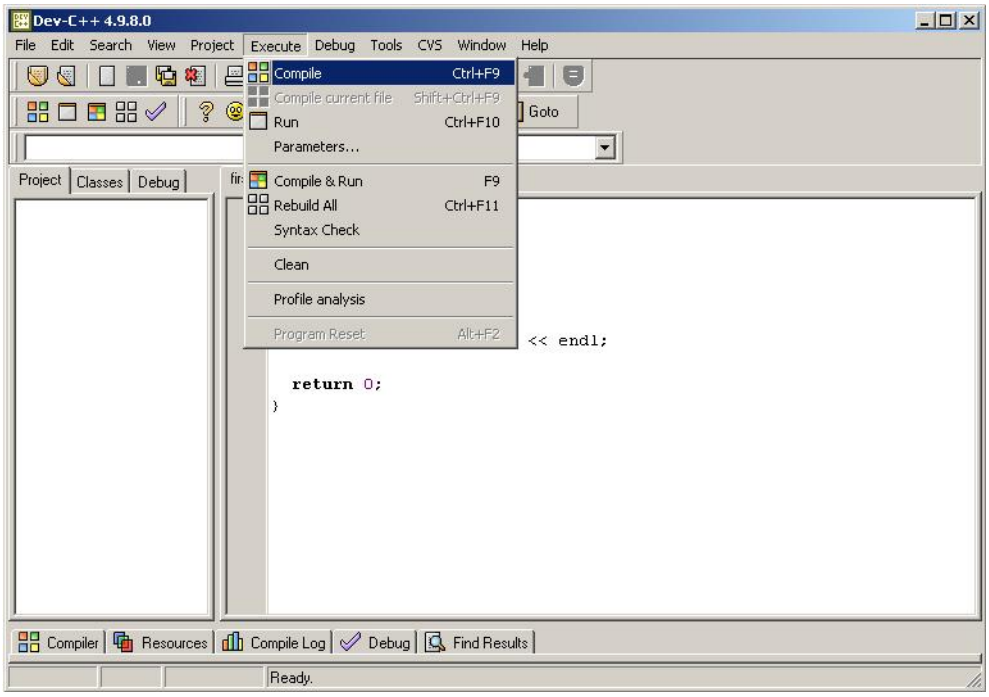
- Open a C++ file





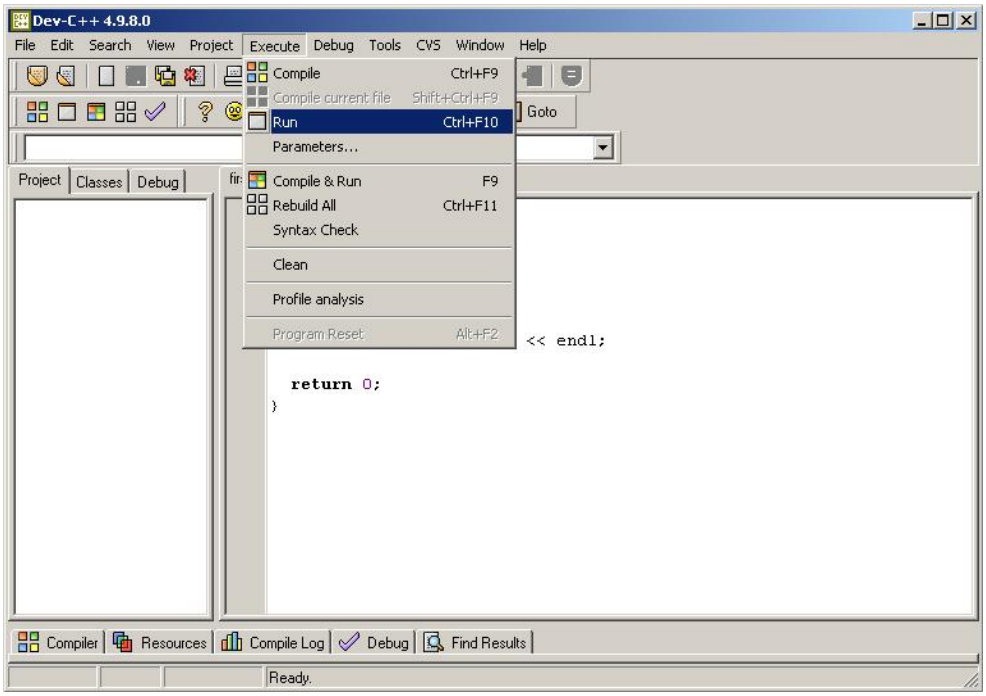
cont…

- Go to the Execute menu and select compile



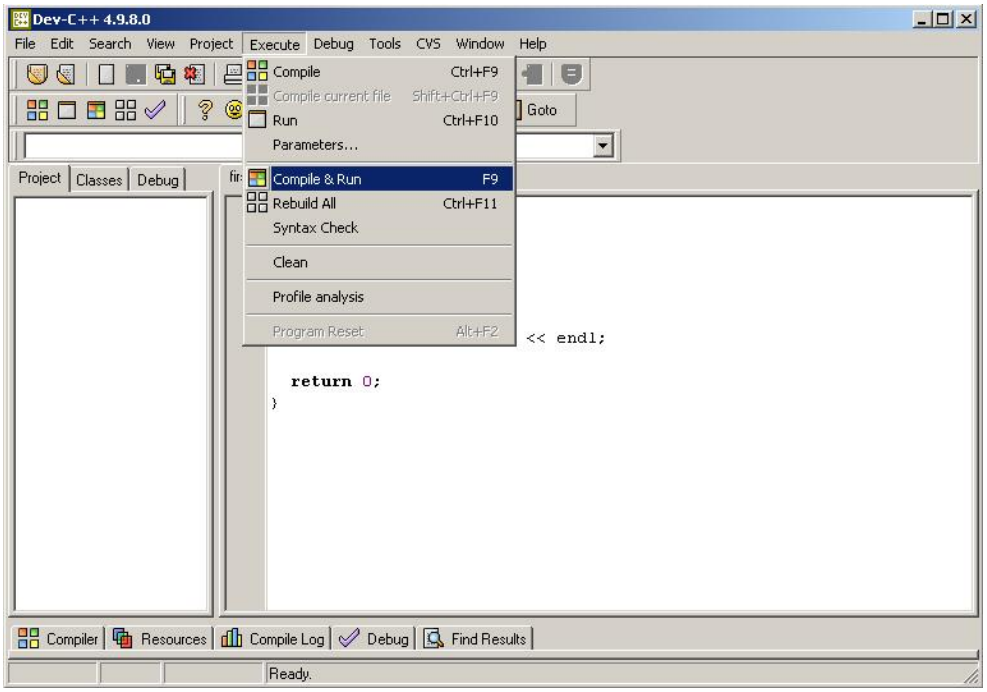
cont…

- Go to the Execute menu and select Run



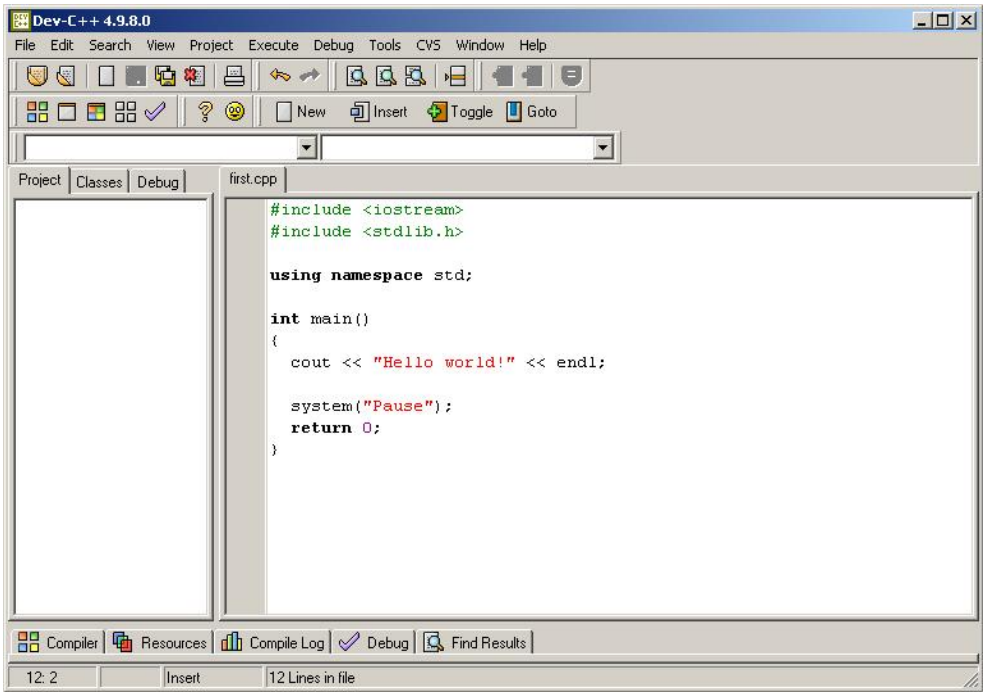
cont…

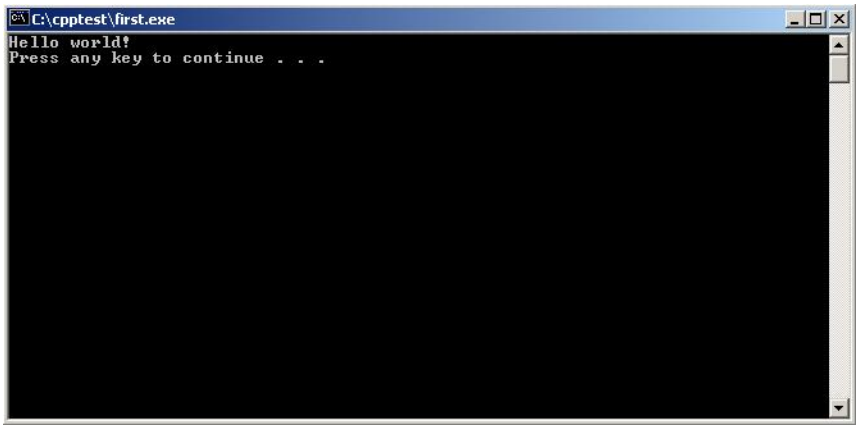
- You can also select Compile & Run initially:



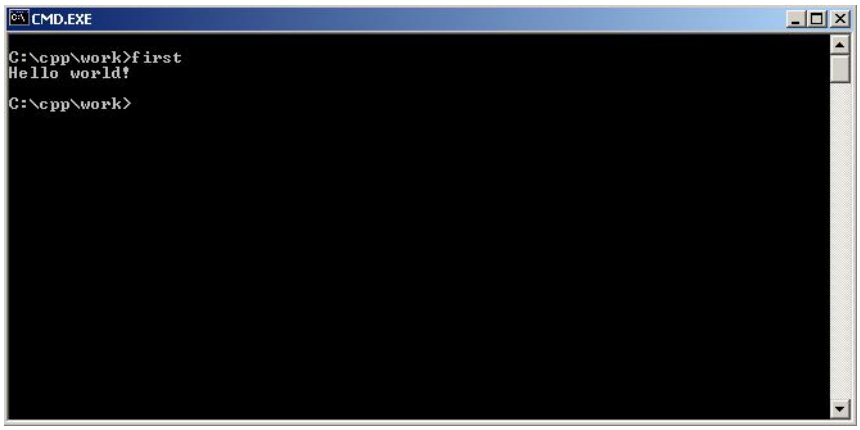
cont…

- To view the output you have two choices:
  - Alter the code as shown below
  - Or// open a separate command window in your *work* directory and execute the .exe file directly





cont…



APPENDIX J : USING OPENGL & GLUT

- OpenGL is an open source library of graphics routines available on many platforms.
- The Dev-C++ distribution comes with the OpenGL libraries. This means you can compile OpenGL code with Dev-C++.
- GLUT is the OpenGL Utility Toolkit. The library and include files for GLUT are not distributed with Dev-C++ but can be obtained on the Web.
    - Visit: *OpenGL.org* on the Web for a vast assortment of code, examples, and tutorials relating to OpenGL and GLUT.

- Three GLUT related files are included in the *examples* directory for the course, they are:

  glut.h – a header file of the GLUT functions

  libglut32.a – a library of the GLUT functions

  glut32.dll – a dynamic link library for executing programs linked with the above library

- The compile command should look as follows:

  ```
  g++ glutsource.cpp libglut32.a -lglu32 -lopengl32
  ```

  libglu32.a and *libopengl32.a* are OpenGL files included with the Dev-C++ distribution. There are make files for the OpenGL examples in the *examples* directory. These can be used to build the examples and as a reference for the compile command, the libraries to use and their order.

  Note: The order of the files & libraries matters.
- The dynamic link library file *glut32.dll* must be on the search path for the programs to run. If the current working directory is the examples directory, or the examples directory is in the search path, things will run fine.

- Recommended reading & information sources:

  OpenGL.org - on the web

  OpenGL Super Bible by Wright

  OpenGL Programming Guide by Woo

  OpenGL Reference Manual by Shreiner

APPENDIX K : SETTING THE SEARCH PATH

- The operating system only looks in a few places for programs
  - The current working directory
  - The directories listed on the *search path*, PATH
- To change the directories the operating system will search you can modify the PATH environment variable:
  - On the Windows OS:
    - Use the *set* command

      set PATH=%PATH%;C:\Dev C++\bin

      - This will assign the previous value of PATH (indicated by %PATH%) to the PATH variable along with the *C:\Dev C++\bin* directory
    - In this course the settings might be:

      H:\C++\examples> set PATH=%PATH%;C:\Program Files\ Dev C++\bin

      - For the GCC installed on the classroom machines
    - You can alter the PATH environment variable permanently for the entire system by going to:

      Start -> Control Panel -> System -> Advanced System Settings -> Environment Variables

      and editing the PATH variable value.

- If installed properly, and your PATH is set properly, you should be able to type *gcc* and get an error message back from the compiler. If you can't, verify that the *gcc.exe* file is in fact in the */bin* subdirectory you specified. If it is, try running it by typing the complete path to it.
- If the compiler is in fact on your machine, check your PATH variable setting. This can be done by typing *PATH* at the command prompt, to show your PATH variable setting, or by typing just *set*, to show all your environment variable settings. Your PATH variable must contain the complete path to the */bin* directory that contains the *gcc.exe* file.

Notes:

- The actual path to the */bin* subdirectory will likely be different on different machines, depending on the installation directory and name of the particular program that was installed.

- Generally, when setting the PATH value in a command window/shell; that setting is only good within that window/shell.
    - If you exit that window/shell the setting will be lost
- Check your */examples* directory for a provided batch file or script that will set the PATH variable for you.