

# Certificate - Software Development

## Introduction to C Language

# Some Features of C Language

- C is a **general-purpose** programming language initially developed by Dennis Ritchie between 1969 and 1973 at AT&T Bell Labs.
- Like most imperative (procedural) languages in the C has facilities for **structured programming**, allows variable scoping and recursion.
- Because of it's **closeness to machine** and **efficiency** it has been used to write many system software, including Unix operating system.
- C compilers are available for the majority of available computer architectures and operating systems (**portable**).
- Many later languages have borrowed directly or indirectly from C, including D, Go, Rust, Java, JavaScript, C#, Objective-C, Perl, PHP, Python.

# Small Keywords set of Standard C

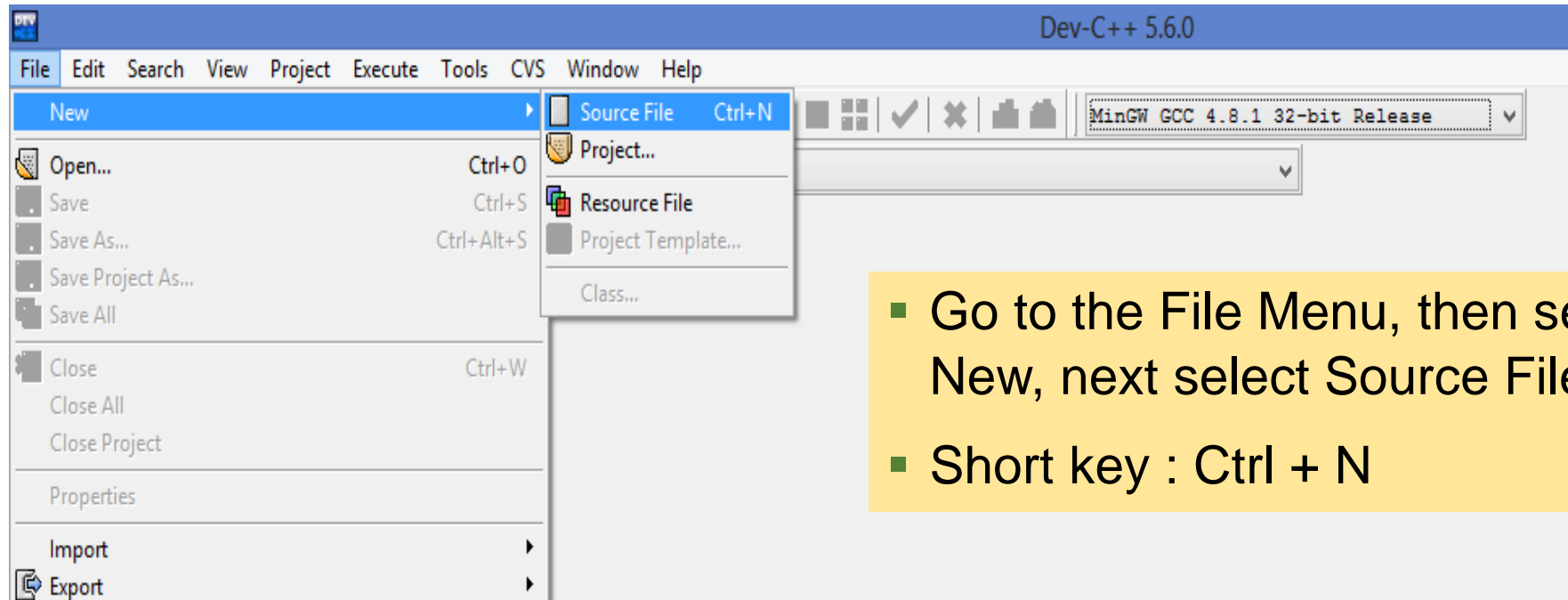
- Here is the list of keywords used in Standard C

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

- These keywords cannot be used as identifiers in the program.
- C is Case sensitive and keywords use lowercase letters.

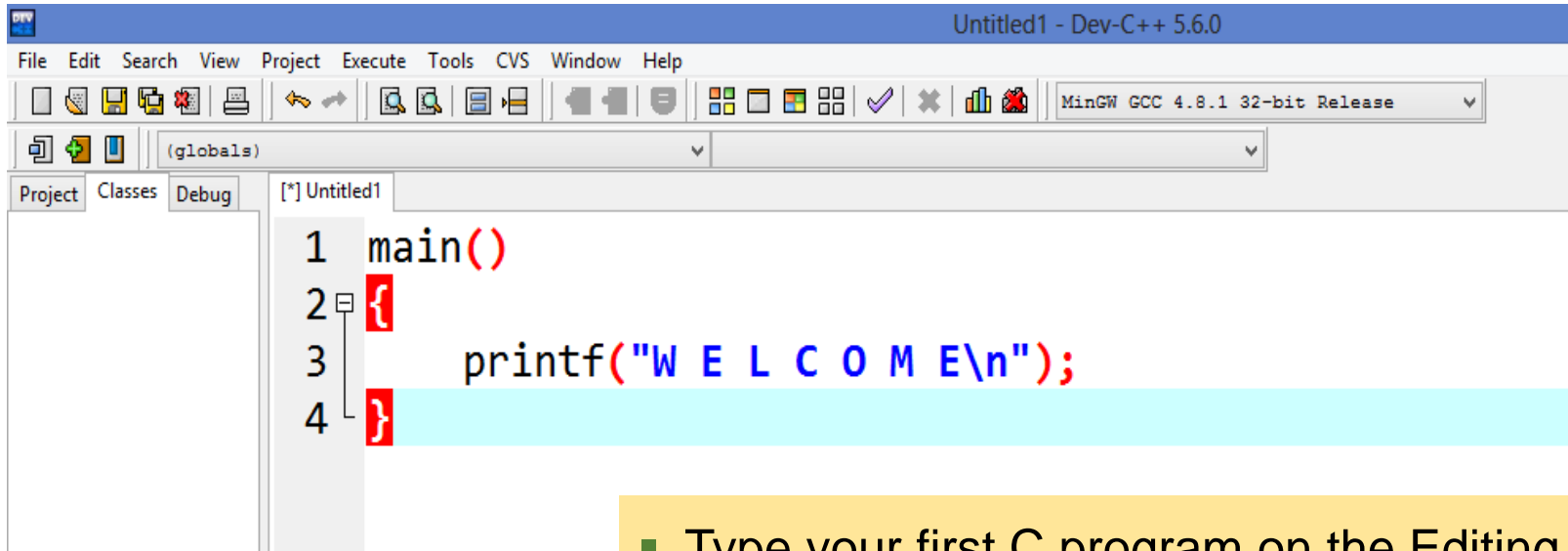
# Introducing Dev C++ IDE

- Bloodshed Dev-C++ is a full-featured Integrated Development Environment (IDE) for the C/C++ programming language.



- Go to the File Menu, then select New, next select Source File
- Short key : Ctrl + N

# Typing my first C Program

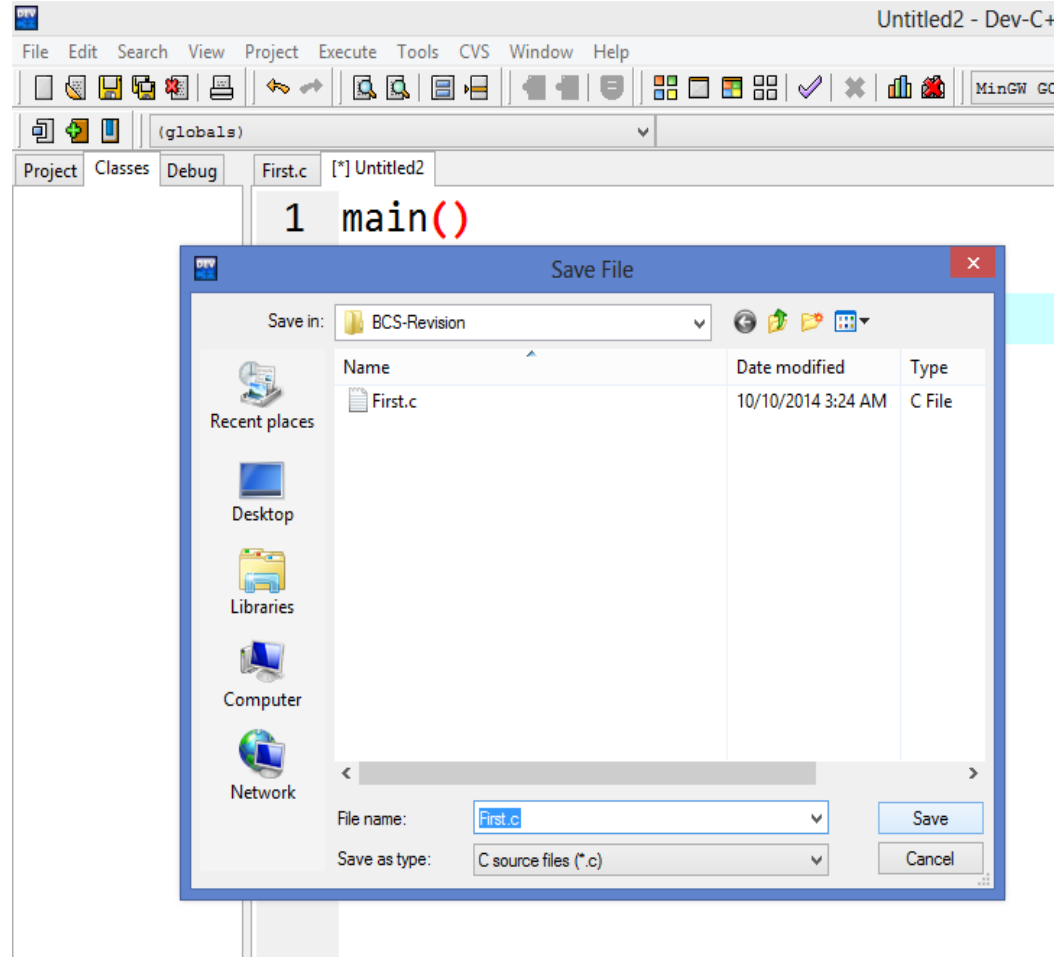


```

1 main()
2 {
3     printf("W E L C O M E\n");
4 }
  
```

- Type your first C program on the Editing window that appears on the IDE.
- You can see the Editor is context sensitive as it highlights different construct in different colours.

# Saving my Program




- Now save your C program. First go to File menu and then select Save.
- Short key : Ctrl + S
- From the Save File Window that appears
  - First change the file type to C source files ( Not C++ )
  - Then give the file name (here it is **first.c**) and
  - Then click on save

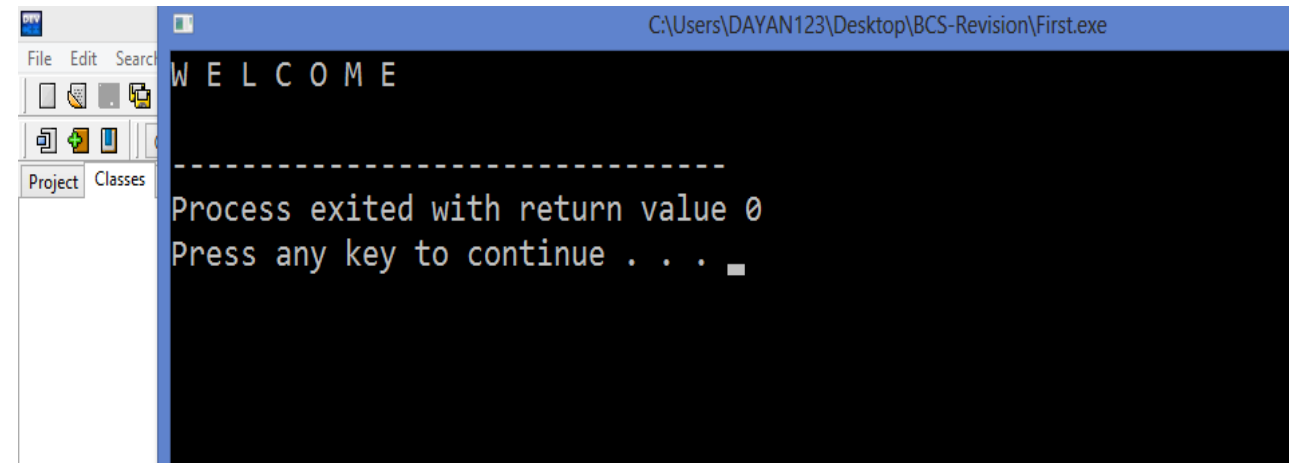


# Compiling & Running my Program

- Now it is time to compile and run your first C program. There are many ways you can do this on this IDE.

- Click on the following icon 
- Press Function key 11 (F11)
- Go to Execute menu and then select Compile + Run

- If your program doesn't have any syntax errors following output should come on the screen.
- When you press any key you can resume back to the IDE.

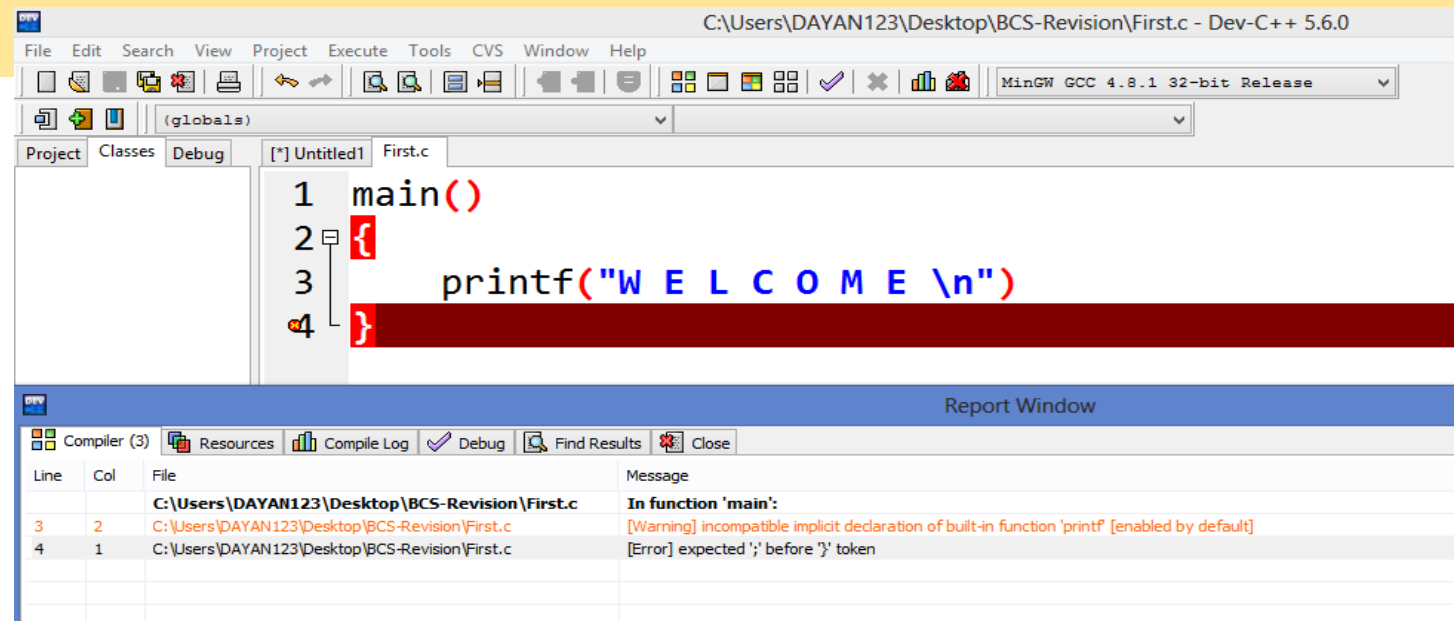


```

C:\Users\DAYAN123\Desktop\BCS-Revision\First.exe
WELCOME
-----
Process exited with return value 0
Press any key to continue . . .
  
```

# Debugging my Program

- If you remove the semicolon after printf() statement and then try to run the program following will appear due to the presence of syntax errors.
- Note that the highlighted line may not always contain the error and it can be in the line above or below that. See the report window for the description of the errors which will help in fixing the problem ( de-bugging).



The screenshot shows the Dev-C++ IDE with a C program in the editor. The program is as follows:

```

1 main()
2 {
3     printf("W E L C O M E \n")
4 }
  
```

The line containing the printf statement is highlighted in red. Below the editor, the 'Report Window' is open, displaying the following error messages:

Line	Col	File	Message
3	2	C:\Users\DAYAN123\Desktop\BCS-Revision\First.c	In function 'main': [Warning] incompatible implicit declaration of built-in function 'printf' [enabled by default]
4	1	C:\Users\DAYAN123\Desktop\BCS-Revision\First.c	[Error] expected ';' before '}' token



# Data types in C Language

- There are only a few basic data types in C. The first ones we'll be encountering and using are:

Data Type	Description	Range / Remarks
<b>char</b>	1-byte int type	Range -128 to +127 used to store characters
<b>int</b>	int type (2 or 4 bytes)	Whether it is 2 or 4 bytes depend on compiler
<b>short</b>	2-byte int type	Range -32,768 to +32,767
<b>long</b>	4-byte int type	For larger integers
<b>float</b>	4-byte floating point	Store fractional numbers
<b>double</b>	8-byte floating point	Store fractional numbers with more precision

# Declaration vs Initialization

- A declaration tells the compiler the name and the type of a variable you'll be using in your program. These are examples

```
char ch;
```

```
int num;
```

```
float avg;
```

- You can also declare several variables of the same type in one declaration, separating them with commas:

```
int num1, num2;
```

```
float sum, avg;
```

- At the time of declaring a variable you can also assign a value. That is called as initializing the variable

```
int num1=10, num2=98;
```

```
float pi=3.14, avg;
```

# Rules for Names (Identifiers)

- Variable names and function names are technically called as **user defined identifiers**. You must follow the rules given below for all such identifiers.
  - Can consist of letters, numbers, and underscores.
  - Names must begin with a letter or underscore.
  - Identifiers are case sensitive.
  - You cannot use keywords ( reserved words ).

# Constants in Programs

- Constant is just an immediate, absolute value found in an expression. The simplest constants are decimal integers, e.g. 0, 1, -12, 123. But C compilers can recognize a variety of constant types bases on how you give them.

Example1	Example2	Constant Type	Explanation
659	-28	integer decimal	Normal numbers are read as Decimal
<b>0</b> 56	- <b>0</b> 67	integer octal	With prefix <b>0</b> it is read as Octal
<b>0X</b> 9A	- <b>0X</b> E7	integer hexadecimal	With prefix <b>0X</b> it is read as Hexadecimal
'a'	'\n'	character constant	Enclosing within single quotes takes ASCII value
659 <b>L</b>	-90 <b>L</b>	long constant	Suffix <b>L</b> will force to store using 4 bytes
"Hello"	"123-45"	string constant	Strings are enclosed in double quotes
3.14	-0.0001	float constant	Fractional values
123 <b>e</b> 4	123.453 <b>e</b> 2	float constant	Scientific notation e=power of 10

# Using Identifiers for Constants

- If we embed constants values in the program body, such code becomes more difficult to change. Therefore it is good practice to define identifiers represent constants. There are 2 ways to do this.

## Using **# define** directive

```
# define pi 3.14
```

```
# define company "Esoft"
```

```
# define max 10
```

## Using constant declarations

```
const float pi = 3.14;
```

```
const char company[] = "Esoft";
```

```
const int max = 10;
```

- *# define is a pre-compiler director*
- *Before compilation these symbols are replaced by the values*

# Structure of a C program

- In C language all statements are defined inside a function. In every program there should be an important `main()` function. In addition to `main` there can be more functions in the program.

## Every program must have `main()` function

`main()` ← *function name*

`{` ← *function begin*

Statement 1;

Statement 2;

`}` ← *function end*





# Lesson Summary

- Features of C Language
- Keywords set of Standard C
- My first program on Dev C++ IDE
- Basic Data Types in C
- Declaration and Initializing
- Rules for Identifiers
- Constants in C
- Program Structure