4

Key concepts

- **C++ character set**

- **Tokens**
  - o Keywords
  - o Identifiers
  - o Literals
  - o Punctuators
  - o Operators

- **Integrated Development Environment (IDE)**
  - o Geany IDE

Getting Started with C++

C++ (pronounced "C plus plus") is a powerful, popular object oriented programming (OOP) language developed by Bjarne Stroustrup. The idea of C++ comes from the C increment operator ++, thereby suggesting that C++ is an added (incremented) version of C language.

The C++ language can be used to practice various programming concepts such as sequence, selection and iteration which we have already discussed in Chapter 3. In this chapter, we will have a brief overview of the fundamentals of C++. We will also familiarise different language processor packages that are used to write C++ programs.

Just like any other language, the learning of C++ language begins with the familiarisation of its basic symbols called characters. The lerning hierarchy proceeds through words, phrases (expressions), statements, etc. Let us begin with the learning of characters.

4.1 Character set

As we know, the study of any language, such as English, Malayalam or Hindi begins with the alphabet. Similarly, the C++ language also has its own alphabet. With regard to a programming language the alphabet is known as character set. It is a set of valid symbols, called characters

Dr. Bjarne Stroustrup developed C++ at AT&T Bell Laboratories in Murray Hill, New Jersey, USA. Now he is a visiting Professor at Columbia University and holder of the College of Engineering Chair in Computer Science at Texas A&M University. He has received numerous honors. Initial name of this language was 'C with classes'. Later it was renamed to C++, in 1983.

Bjarne Stroustrup

set. It is a set of valid symbols, called characters that a language can recognize. A character represents any letter, digit, or any other symbol. The set of valid characters in a language which is the fundamental units of that language, is collectively known as **character set.** The character set of C++ is categorized as follows:

(i) Letters : A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

a b c d e f g h i j k l m n o p q r s t u v w x y z

(ii) Digits : 0 1 2 3 4 5 6 7 8 9

(iii) Special characters : + - * / ^ \ ( ) [ ] { } = < > . ' " $, ; : % ! & ? _ (underscore) # @

(iv) White spaces : Space bar (Blank space), Horizontal Tab (→), Carriage Return (↵) , Newline, Form feed

(v) Other characters : C++ can process any of the 256 ASCII characters as data or as literals.

Spaces, tabs and newlines (line breaks) are called white spaces. White space is required to separate adjacent words and numbers.

4.2 Tokens

After learning the alphabet the second stage is learning words constituted by the alphabet (or characters). The term 'token' in the C++ language is similar to the term 'word' in natural languages. **Tokens** are the fundamental building blocks of the program. They are also known as lexical units. C++ has five types of tokens as listed below:

1. Keywords
2. Identifiers
3. Literals
4. Punctuators
5. Operators

4.2.1 Keywords

The words (tokens) that convey a specific meaning to the language compiler are called **keywords**. These are also known as reserved words as they are reserved by the language for special purposes and cannot be redefined for any other purposes. The set of 48 keywords in C++ are listed in Table 4.1. Their meaning will be explained in due course.

asm	continue	float	new	signed	try
auto	default	for	operator	sizeof	typedef
break	delete	friend	private	static	union
case	do	goto	protected	struct	unsigned
catch	double	if	public	switch	virtual
char	else	inline	register	template	void
class	enum	int	return	this	volatile
const	extern	long	short	throw	while

Table 4.1 : Keywords of C++

4.2.2 Identifiers

We usually assign names to places, people, objects, etc. in our day to day life, to identify them from one another. In C++ we use identifiers for this purpose. **Identifiers** are the user-defined words that are used to name different program elements such as memory locations, statements, functions, objects, classes etc. The identifiers of memory locations are called *variables*. The identifiers assigned to statements are called *labels*. The identifiers used to refer a set of statements are called *function names*.

While constructing identifiers certain rules are to be strictly followed for their validity in the program. The rules are as follows:

- Identifier is an arbitrary long sequence of letters, digits and underscores (_).

- The first character must be a letter or underscore (_).

- White space and special characters are not allowed.

- Keywords cannot be used as identifiers.

- Upper and lower case letters are treated differently, i.e. C++ is case sensitive.

Examples for some valid identifiers are Count, Sumof2numbers, _1stRank, Main, Average_Height, FOR

The following are some invalid identifiers due to the specified reasons:

Sum of Digits	→	Blank space is used
1styear	→	Digit is used as the first character
First.Jan	→	Special character (.) is used
for	→	It is a keyword

Identify invalid identifiers from the following list and give reasons:

`Data_rec,   _data,   Idata,   datal,   my.file,`
`asm,   switch,   goto,   break`

Let us do

4.2.3 Literals

Consider the case of the Single Window System for the admission of Plus One students. You may have given your date of birth in the application form. As an applicant, your date of birth remains the same throughout your life. Once they are assigned their initial values, they never change their value. In mathematics, we know that the value of π is a constant and the value of gravitational constant 'g' never changes, i.e. it remains 9.8m/s^2. Like that, in C++, we use the type of tokens called **literals** to represent data items that never change their value during the program run. They are often referred to as constants. Literals can be divided into four types as follows:

1. Integer literals
2. Floating point literals
3. Character literals
4. String literals

Integer literals

Consider the numbers 1776, 707, -273. They are integer constants that identify integer decimal values. The tokens constituted only by digits are called **integer literals** and they are whole numbers without fractional part. The following are the characteristics of integer literals:

- An integer constant must have at least one digit and must not contain any decimal point.

- It may contain either + or – sign as the first character, which indicates whether the number is positive or negative.

- A number with no sign is treated as positive.

- No other characters are allowed.

Classify the following into valid and invalid integer constants and give your reasons:

Let us do

| 77,000 | 70 | 314. | -5432 | +15346 |
| +23267 | -.7563 | -02281+0 | 1234E56 | -9999 |

In addition to decimal numbers (base 10), C++ allows the use of octal numbers (base 8) and hexadecimal numbers (base 16) as literals (constants). To express an octal number we have to precede it with a 0 (zero character) and in order to express a hexadecimal number we have to precede it with the characters 0x (zero, x). For example, the integer constants 75, 0113 and 0x4B are all equivalent to each other. All of these represent the same number 75 (seventy-five), expressed as a base-10 numeral, octal numeral and hexadecimal numeral, respectively.

Floating point literals

You may have come across numbers like 3.14159, 3.0×10^8, 1.6×10^{-19} and 3.0 during your course of study. These are four valid numbers. The first number is π (Pi), the second one is the speed of light in meter/sec, the third is the electric charge of an electron (an extremely small number) – all of them are approximated, and the last one is the number three expressed as a floating-point numeric literal.

Floating point literals, also known as real constants are numbers having fractional parts. These can be written in one of the two forms called fractional form or exponential form.

A real constant in fractional form consists of signed or unsigned digits including a decimal point between digits. The rules for writing a real constant in fractional form are given below:

- A real constant in fractional form must have at least one digit and a decimal point.

- It may also have either + (plus) or – (minus) sign preceding it.

- A real constant with no sign is assumed to be positive.

A real constant in exponential form consists of two parts: *mantissa* and *exponent*. For instance, 5.8 can be written as $0.58 \times 10^1 = 0.58E1$ where mantissa part is 0.58 (the part appearing before **E**) and exponential part is 1 (the part appearing after **E**). The number E1 represents 10^1. The rules for writing a real constant in exponential form are given below:

- A real constant in exponent form has two parts: a mantissa and an exponent.

- The mantissa must be either an integer or a valid fractional form.

- The mantissa is followed by a letter **E** or **e** and the exponent.

- The exponent must be an integer.

The following are valid real constants.

52.0	107.5	-713.8	-.00925
453.E-5	1.25E08	.212E04	562.0E09
152E+8	1520E04	-0.573E-7	-.097

Some invalid real constants are given along with the reason:

58,250.262 (Comma is used), 5.8E (No exponent part), 0.58E2.3 (Fractional number is used as exponent).

Classify the following into valid and invalid real constants and justify your answer:

77, 00,000	7.0	3.14	-5.0E5.4	+53.45E-6
+532.67.	.756E-3	-0.528E10	1234.56789	34,56.24
4353	+34/2	5.6E	4356	0

Let us do

Character literals

When we want to store the letter code for gender usually we use 'f' or 'F' for *Female* and 'm' or 'M' for *Male*. Similarly, we may use the letter 'y' or 'Y' to indicate *Yes* and the letter 'n' or 'N' to indicate *No*. These are single characters. When we refer a single character enclosed in single quotes that never changes its value during the program run, we call it a **character literal** or **character constant**.

Note that x without single quote is an identifier whereas 'x' is a character constant. The value of a single character constant is the ASCII value of the character. For instance, the value of 'c' will be 99 which is the ASCII value of 'c' and the value of 'A' will be the ASCII value 65.

C++ language has certain non-graphic character constants, which cannot be typed directly from the keyboard. For example, there is no way to express the Carriage Return or Enter key, Tab key and Backspace key. These non-graphic symbols can be represented by using **escape sequences**, which consists of a backslash (\) followed by one or more characters. It should be noted that even though escape sequences contain more than one character enclosed in single quotes, it uses only one corresponding ASCII code to represent it. That is why they are treated as character constants. Table 4.2 lists escape sequences and corresponding characters.

Escape sequence	Corresponding non-graphic character
\a	Audible bell (alert)
\b	Back Space
\f	Form feed
\n	New line or Line feed
\r	Carriage Return
\t	Horizontal Tab
\v	Vertical Tab
\\	Back slash
\'	Single quote
\"	Double quote
\?	Question mark
\0	Null character

Table 4.2 : Escape Sequences

In Table 4.2, you can also see sequences representing \', \" and \?. These characters can be typed from the keyboard but when used without escape sequence, they carry a special meaning and have a special purpose. However, if these are to be displayed or printed as it is, then escape sequences should be used. Examples of some valid character constants are: 's', 'S', '$', '\n', '+', '9'

C++ represents Octal Number and Hexadecimal Number with the help of escape sequences. The \0n and \xHn represent a number in the Octal Number System and the Hexadecimal Number System respectively.

Some invalid character constants are also given with the reason for invalidity:

A (No single quotes), '82' (More than one character), "K" (Double quotes instead of single quotes), '\g' (Invalid escape sequence or Multiple characters).

String literals

Nandana is a student and she lives in Bapuji Nagar. Here, "Nandana" is the name of a girl and "Bapuji Nagar" is the name of a place. These kinds of data may need to be processed with the help of programs. Such data are considered as string constants and they are enclosed within double quotes. A sequence of one or more characters enclosed

within a pair of double quotes is called **string constant**. For instance, `"Hello friends"`, `"123"`, `"C++"`, `"Baby\'s Day Out"`, etc. are valid string constants.

Classify the following into different categories of literals.

```
     'a'      "rita"     -124     12.5     -12e-1

"raju\'s pen"  0     -11.999   '\\'     32760
```

Let us do

4.2.4 Punctuators

In languages like English, Malayalam, etc. punctuation marks are used for grammatical perfection of sentences. Consider the statement: *Who developed C++?* Here '?' is the punctuation mark that tells that the statement is a question. Similarly at the end of each sentence we put a full stop (.). In the same way C++ also has some special symbols that have syntactic or semantic meaning to the compiler. These are called **punctuators**. Examples are: #   ; ' "   ( ) [ ] { }

The purpose of each punctuator will be discussed in due course.

4.2.5 Operators

When we have to add 5 and 3, we express it as 5 + 3. Here + is an operator that represents the addition operation. Similarly, C++ has a rich collection of operators. An **operator** is a symbol that tells the compiler about a specific operation. They are the tokens that trigger some kind of operation. The operator is applied on a set of data called **operands**. C++ provides different types of operators like arithmetic, relational, logical, assignment, conditional, etc. We will discuss more about operators in the next chapter.

Classify the following into different categories of tokens.

```
    /       -124      +    -12e-1    '    "KL01"

  Sum "raju\'s pen"if    rita   '\\' break  }
```

Let us do

4.3 Integrated Development Environment (IDE)

Now we have learned the basic elements of a C++ program. Before we start writing C++ programs, we must know where we will type this program. Like other programming languages, a text editor is used to create a C++ program. The compilers such as GCC (GNU Compiler Collection), Turbo C++, Borland C++, and many other similar

compilers provide an Integrated Development Environment (IDE) for developing C++ programs. Many of these IDEs provide facilities for typing, editing, searching, compiling, linking and executing a C++ program. We use GCC with Geany IDE (IT@School Ubuntu Linux 12.04) for the purpose of illustrating the procedure for coding, compiling and executing C++ programs.

GCC with Geany IDE

There are many compilers available like GCC, Borland C++, Turbo C++, Microsoft C++ (Visual C++), Unix AT & T C++, etc. Out of these, GCC is a free software available with Linux operating system. GCC stands for GNU Compiler Collection and is one of the popular C++ compilers which works with ISO C++ standard. GCC versions are available for Windows platforms also. Geany is a cross-platform IDE that works with GCC for writing, compiling and executing C++ programs.

a. Opening the edit window

The edit window of Geany IDE can be opend from the **Applications** menu of Ubuntu Linux by proceeding as follows:

<div align="center">

Applications → Programming → Geany

</div>

Geany IDE opens its windows as shown in Figure 4.1. It has a title bar, menu bar, tool bar and a code edit area. We can see a tab named **untitled**, which indicates the file
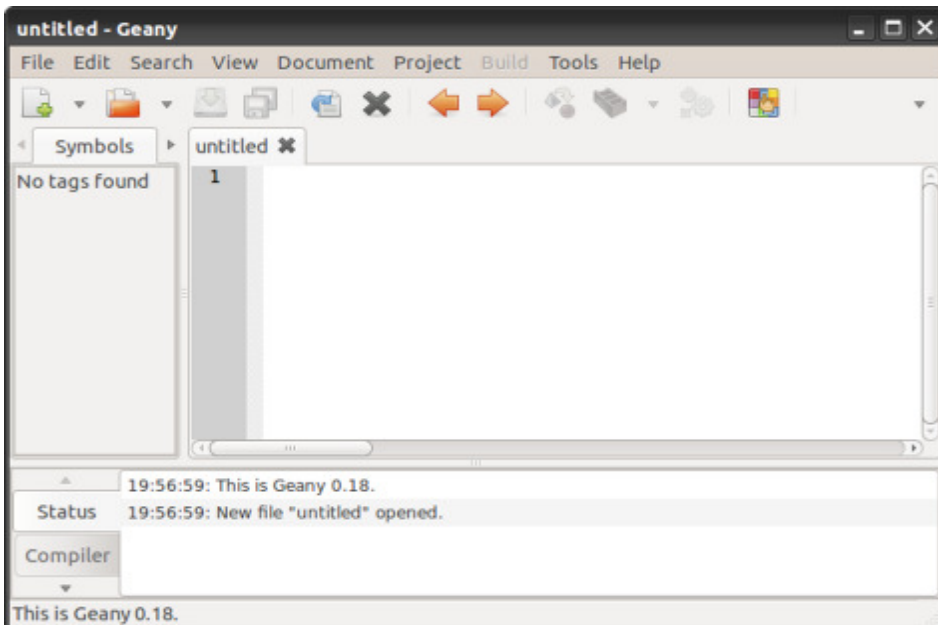

Fig. 4.1: Edit window of Geany IDE for Ubuntu Linux 12.04

name for the opened edit area. If we use Geany 1.24 on Windows OS, the opening window will be as shown in Figure 4.2. We can see that both of these are quite the same.
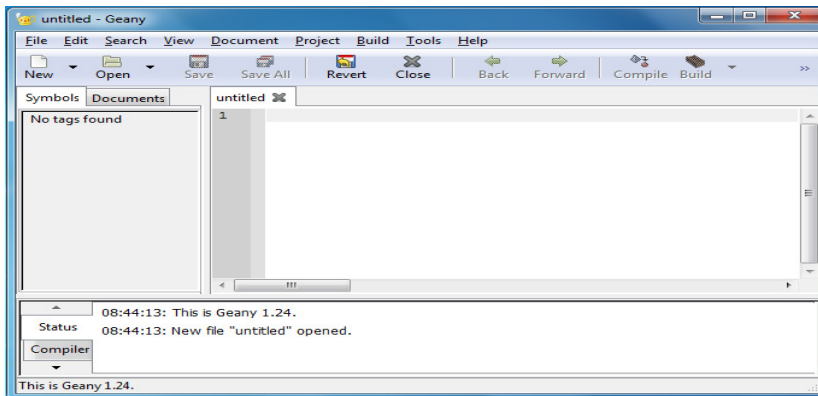

Fig. 4.2: Edit window of Geany IDE 1.24

In this window, we type the program in a file with the default name untitled. To open a new file, choose **File** menu, then select **New** option or click New button on the toolbar.

b. Saving the program

Once a file is opened, enter the C++ program and save it with a suitable file name with extension **.cpp**. GCC being a collection of compilers, the extension decides which compiler is to be selected for the compilation of the code. Therefore we have to specify the extension without fail. If we give the file name before typing the program, GCC provides different colours automatically to distinguish the types of tokens used in the program. It also uses indentation to identify the level of statements in the source code. We will disucss the concept of indentation later.

Let us write a simple program given as Program 4.1 and save with the name **welcome.cpp**.

Program 4.1: A program to familiarise the IDE

```cpp
// my first C++ program
#include<iostream>
using namespace std;
int main()
{
    cout << "Welcome to the world of C++";
    return 0;
}   //end of program
```

The IDE window after entering Program 4.1 is shown in Figure 4.3. Observe the diffrence in colours used for the tokens.
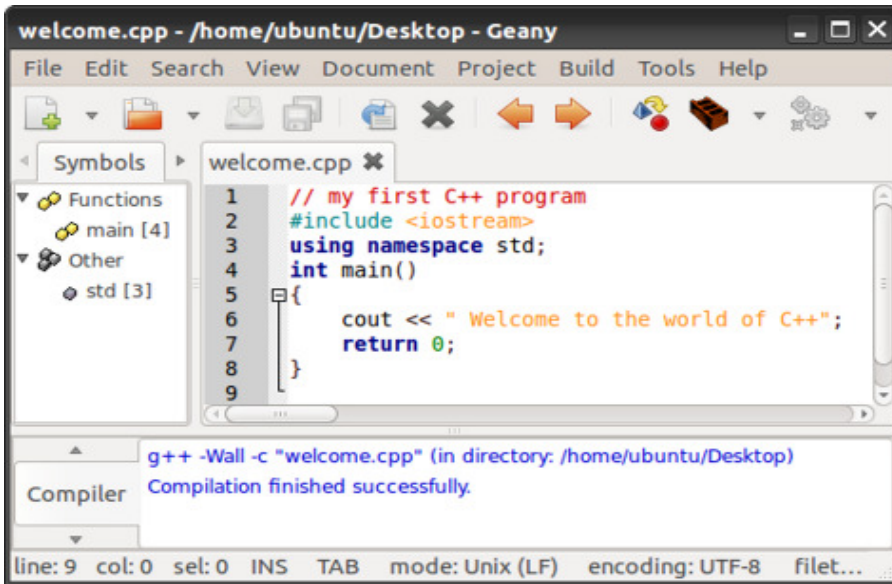
Fig. 4.3 : Program saved with a name in Geany

To save the program, choose **File** menu and select **Save** option or use the keyboard shortcut **Ctrl+S**. Alternatively the file can be saved by clicking the Save button [image] in the toolbar.

It is a good practice to save the program every now and then, just by pressing **Ctrl+S**. This helps to avoid the loss of data due to power failures or due to unexpected system errors. Once the program typing is completed, it is better to save the file before compiling or modifying. Copying the files from the temporary volatile primary memory to permanent non volatile secondary memory for storage is known as **saving** the program.

> C++ program files should have a proper extension depending upon the implementation of C++. Different extensions are followed by different compilers. Examples are .cpp, .cxx, .cc, .c++
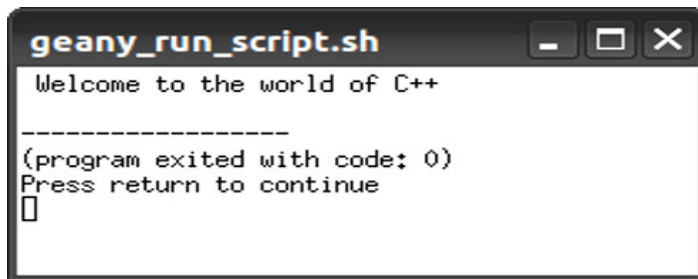
c. Compiling and linking the program

The next step is to compile the program and modify it, if errors are detected. For this, choose **Build** menu and select **Compile** option. Alternatively, we can use the Compile button [image]. If there are some errors, those errors will be displayed in the compiler status window at the bottom, otherwise the message Compilation finished successfully will be displayed. (refer Figure 4.3).

After successful compilation, choose **Build** menu and select **Build** option for linking or click the Build button [image] in the toolbar. Now the program is now ready for execution.

d. Running/Executing the program

Running the program is the process by which a computer carries out the instructions of a computer program. To run the program, choose **Build** menu and select **Execute** option. The program can also be executed by pressing the function key **F5**. The output will be displayed in a new window as shown in Figure 4.5.

Fig. 4.5 : Output window of Geany

e. Closing the IDE

Once we have executed the program and desired output is obtained, it can be closed by selecting **Close** option from **File** menu or by clicking the **X** symbol in the active tab or in the title bar of the IDE. For writing another program, a new file can be opened by the **New** option from the **File** menu or by clicking the **New** button in the tool bar. The key combination **Ctrl+N** can also be used for the same.

After developing program, we can come out of Geany IDE by choosing **File** menu and selecting **Quit** option. The same can be achieved by clicking the Close button of the IDE window or by using the key combination **Ctrl+Q**.

Let us do

1. Write a program to print the message "SMOKING IS INJU-RIOUS TO HEALTH" on screen.

2. Write a program to display the message "TOBACCO CAUSES CANCER" on monitor.

Let us sum up

C++ was developed by Bjarne Stroustrup in early 1980s. C++ has its own character set. Tokens are the smallest unit of a program and are constituted by one or more characters in C++. There are five types of tokens namely keywords, identifiers, literals, punctuators and operators. Programs are written in computer with the help of an editor. Software like GCC and Geany IDE provide facilities to enter the source code in the computer, compile it and execute the object code.

Learning outcomes

After the completion of this chapter the learner will be able to

- list the C++ character set.
- categorise various tokens.
- identify keywords.
- write valid identifiers.
- classify various literals.
- identify the main components of Geany IDE.
- write, compile and run a simple program.

Sample questions

Very short answer type

1. What are the different types of characters in C++ character set?

2. What is meant by escape sequences?

3. Who developed C++?

4. What is meant by tokens? Name the tokens available in C++.

5. What is a character constant in C++ ?

6. How are non-graphic characters represented in C++? Give an example.

7. Why are the characters \ (slash), ' (single quote), " (double quote) and ? (question mark) typed using escape sequences?

8. Which escape sequences represent newline character and null character?

9. An escape sequence represents _____ characters.

10. Which of the following are valid character/string constants in C++?

    ```
    'c'       'anu'       "anu"       mine       'main's'       " "
    'char              '\ \'
    ```

11. What is a floating point constant? What are the different ways to represent a floating point constant?

12. What are string-literals in C++? What is the difference between character constants and string literals?

13. What is the extension of C++ program file used for running?

14. Find out the invalid identifiers among the following. Give reason for their invalidity

    a) `Principal amount`   b) `Continue`   c) `Area`

    d) `Date-of-join`   e) `9B`

15. A label in C++ is _____.

    a) Keyword   b) Identifier   c) Operator   d) Function

16. The following tokens are taken from a C++ program. Fill up the given table by placing them at the proper places

    (`int`, `cin`, `%`, `do`, `=`, `"break"`, `25.7`, `digit`)

Keywords	Identifiers	Literals	Operators

Short answer type

1. Write down the rules governing identifiers.
2. What are tokens in C++ ? List different types of tokens allowed in C++.
3. Distinguish between identifiers and keywords.
4. How are integer constants represented in C++? Explain with examples.
5. What are character constants in C++? How are they implemented?

Long answer type

1. Briefly describe different types of tokens.
2. Explain different types of literals with examples.
3. Briefly describe the Geany IDE and its important features.