C Programming concepts

A C Program

- Every C program consists of one or more modules called **functions**. One of the functions must be called **main**. The program will always begin by executing the main function, which may call other functions. Any other function must be defined separately either ahead or after the *main* function.

Each Function Contains:

- A function heading – consists of the **function name**, followed by an optional list of **arguments**.
- **Argument declarations –** Argument types (if the function has arguments)
- **Body** – What the function should do. Enclosed in a pair of braces({ }). – The body contains expression detailing what the function should do.

Example:

- **A C program to output a greeting**

```
#include <stdio.h>
void main()
{
printf(“hello world”); //printf is an output function
}//end
```

Comments

- These are statements within the program that are ignored by the compiler. The programmer uses comments to explain what the purpose of a statement or group of statements in the program does.
- A comment line is started with two forward slashes (//). Everything that comes after the comment indicators in the line will be ignored.

Comments

- Some times the comment can span several lines. In this case, they should be enclosed as shown below

```
/* comments
..comments */
```

Use of Comments

- The following is a summary of the use of comments
- documentation of variables and their usage
- explaining difficult sections of code
- describes the program, author, date, modification changes, revisions etc
- copyrighting

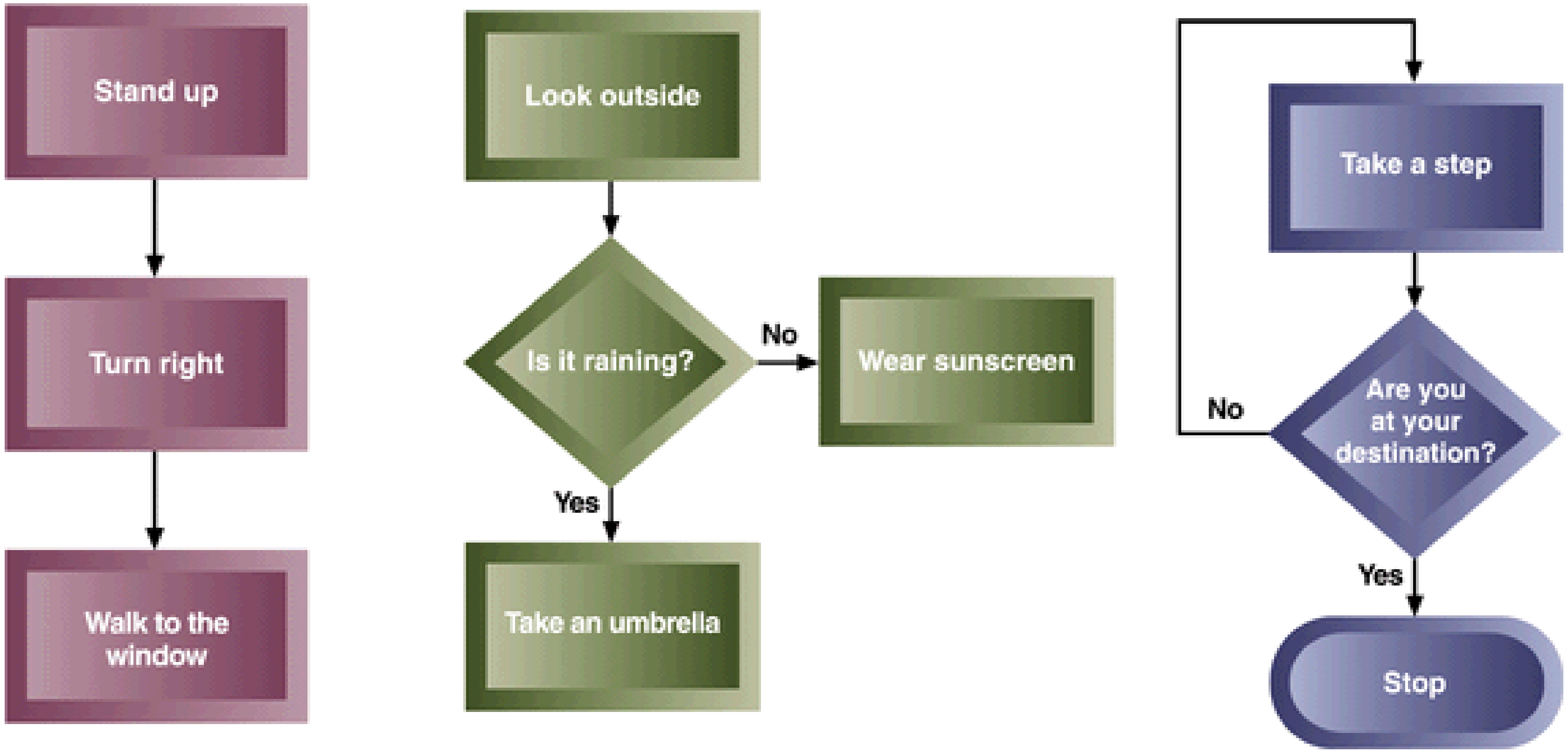
Testing and debugging

- Testing a computer programming identifies errors/bugs.
- A mistake in a program is called a **bug or an error**. They cause a program to “behave in an unexpected manner” i.e yield incorrect results, not to execute at all etc.
- The process of eliminating mistakes in your program is called **debugging.**

Types of errors

- There are three commonly recognized kinds of bugs or errors: syntax errors, run-time errors, and logic errors.
  - A **syntax error** is a grammatical mistake in a program.
  - An error that is detected when a program is run is called a **run-time error.**
  - When a program compiles and executes without any error but gives incorrect output, then it is said to have a **logical error**.

Three Programming Constructs



Sequence

Decision

Repetition

C Program Character set

- A computer program consists of instructions formed using certain symbols and words according to a rigid rules called syntax rules (Grammar) of the programming language used. Each program instruction must conform precisely to the syntax rules of the language. Like all programming languages, C has its own set of vocabulary and grammar

C Program Character set

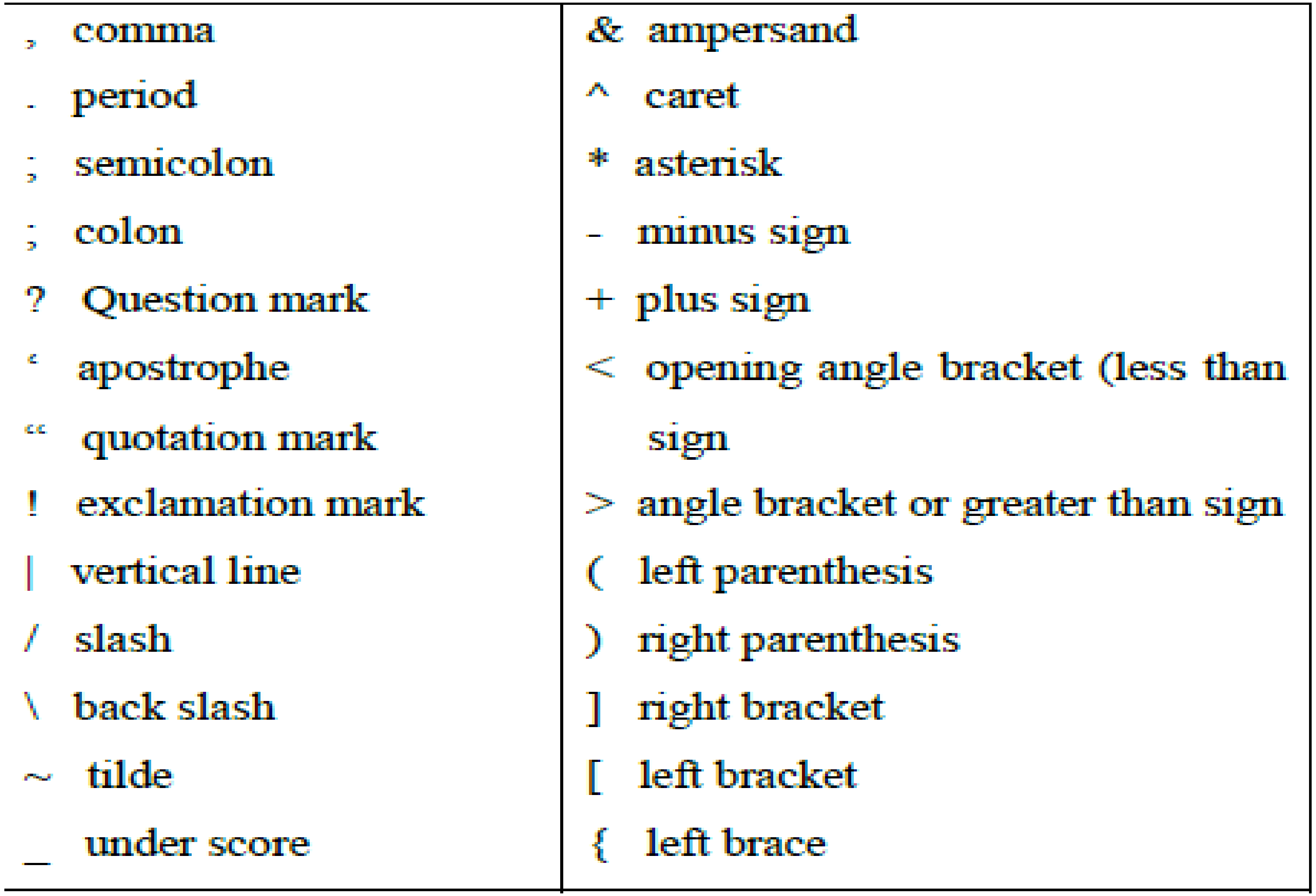
The characters that can be used to form words, numbers and expressions depend upon the computer on which the program is run. The characters in C are grouped into the following categories;

1. Letters : Upper case A ……. Z and Lower casa a ……..z.

2. Digits: 0……..9

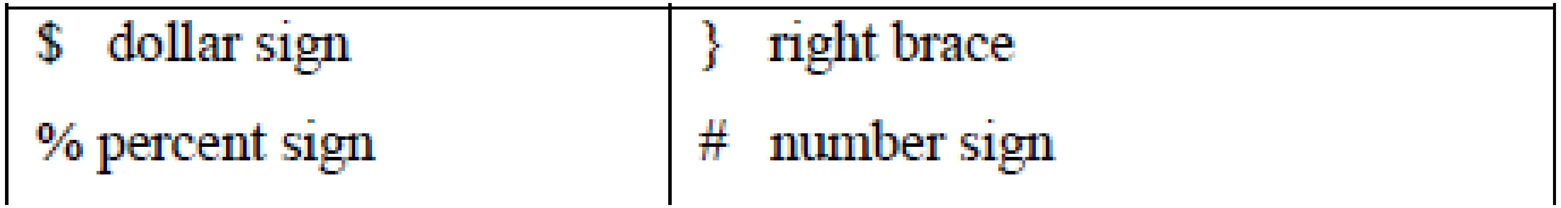
3. Special characters

Special characters



,	comma	&	ampersand
.	period	^	caret
;	semicolon	*	asterisk
;	colon	-	minus sign
?	Question mark	+	plus sign
‘	apostrophe	<	opening angle bracket (less than sign
“	quotation mark		
!	exclamation mark	>	angle bracket or greater than sign
|	vertical line	(	left parenthesis
/	slash	)	right parenthesis
\	back slash	]	right bracket
~	tilde	[	left bracket
_	under score	{	left brace

Special characters



$ dollar sign	} right brace
% percent sign	# number sign

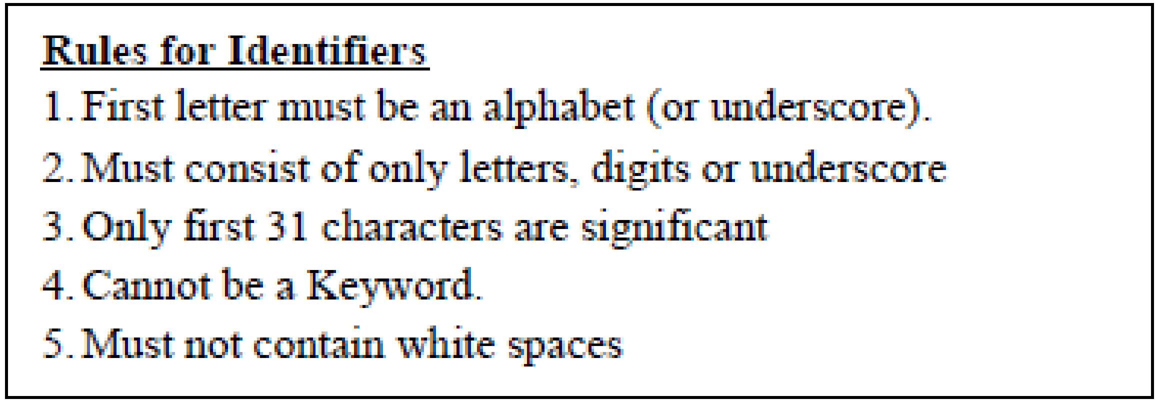
- C compiler ignores white spaces (Blank, Horizontal tab, Carriage return, New line Form feed) unless they are a part of a string constant. White spaces may be used to separate words but are prohibited between the characters of keywords and identifiers.

Reserved words / Key words and Identifiers

- Reserved words (occasionally called keywords) are one type of grammatical construct in programming languages. These words have special meaning within the language and are predefined in the language‟s formal specifications e.g case, char..

Identifiers

- Identifiers refer to the names of variables, functions and arrays. These are user defined names and consists of a sequence of letters and digits, with a letter as a first character.



Rules for Identifiers

1. First letter must be an alphabet (or underscore).
2. Must consist of only letters, digits or underscore
3. Only first 31 characters are significant
4. Cannot be a Keyword.
5. Must not contain white spaces

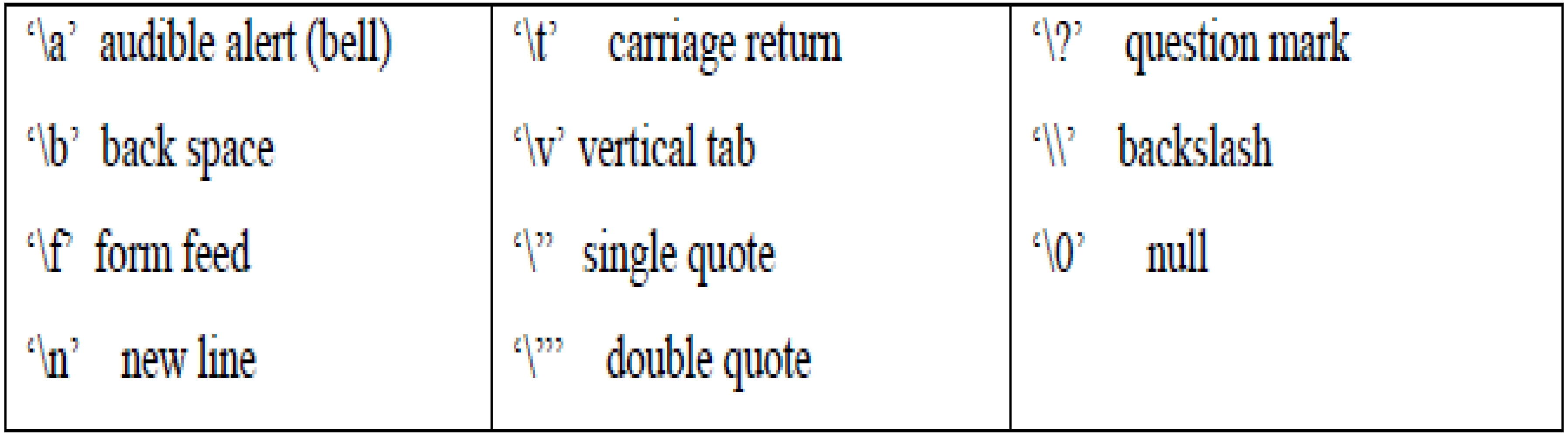
Constants

- Constants in C Program refers to fixed values that do not change during the execution of the program.
- Examples: integer constants, real constants, character constants and string constants.

Backslash Character constants

- C supports some special backslash constants are used in output functions Example „\n‟ stands for new line. Though having two characters they represent one character, these combinations are known as ***escape sequences.***

Backslash Character constants



‘\a’ audible alert (bell)	‘\t’ carriage return	‘\?’ question mark
‘\b’ back space	‘\v’ vertical tab	‘\\’ backslash
‘\f’ form feed	‘\’’ single quote	‘\0’ null
‘\n’ new line	‘\”’ double quote	

Variables

- A variable is a data name that may be used to store a data value. Unlike constants that remain unchanged during the execution of a program, a variable may take different values at different times during execution.
- A variable name should be chosen in a meaningful way so as to reflect its function or nature in the program. Example price, rate, total, amount etc. Naming follows the same rules as stated in the identifiers section.

Variables

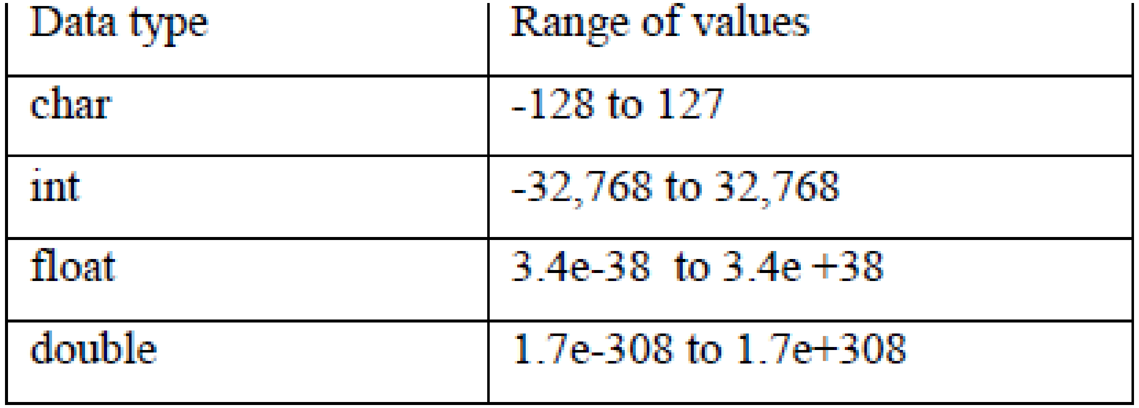
- The data item must be assigned to the variable at some point in the program. The data item can then be accessed later in the program simply by referring to the variable name. A given variable can be assigned different data items at various places within the program. Thus the information represented by a variable can change at various places during the execution of the program but the data type represented by the variable cannot change.

Data types

All C compilers support five (5) fundamental data types namely

- Integer (int),
- Character (Char),
- Floating point (float),
- Double-precision floating point (double) and
- Void.

Data types



Data type	Range of values
char	-128 to 127
int	-32,768 to 32,768
float	3.4e-38 to 3.4e +38
double	1.7e-308 to 1.7e+308

Data types

Integer data types

Integers are whole numbers with a range of values supported by a particular machine. Integer data types are defined in C as **int. C supports three classes of integer storage, short int, int and long int in both signed and unsigned forms.**

Data types

Floating point types

Floating point (or real) numbers are stored in 32 bits (in all 16-bit and 32-bit machines) with 6 digits of precision. Floating point data type is defined in C as **float. When the accuracy provided by float is not sufficient, the type double can be used to define the number.**

Data types

Void types

- Void data type has no values, and usually used to specify the **void type of function which does not return any value to the calling function example main(void)**

Character type

- A single character can be defined as character (char) type of data. Characters are usually stored in 8-bit (one byte) of internal storage.

Declaring variables

- Variable to be used in a C program need to be declared to the C compiler. Declaration of variables does the following two things; it tells the compiler what the variable name is and it specifies what type of data the variable will hold. A variable must be declared before it is used in a C program. A variable can be used to store a value of any data type.
- Syntax for declaring a variable ***data_type v1, v2, vn;***

Declaring variables

- v1, v2 and vn are names for variables and the variables are separated with commas. A declaration statement must end with a semi-colon. examples int count; int count, price double ratio;

Assigning Values to Variables

- Variables are created for use in the program statements. Values can be assigned to variables using the operator „=‟ in a C program or through reading the values from the keyboard using the scanf() function.

C Program format

1. Documentation Section

2. Linker Section

3. Definition Section

4. Global Declaration Section

5. Main() Function Section

{

Declaration section

Executable Section

} 6. Sub-program Section Function 1 Function 2 Function3 etc is User defined functions

Basic Input/output

- To read data from the keyboard and print (display) information on the screen one uses the *printf* and *scanf* functions in C.
- **Printf** and **scanf** are defined in **stdio.h.** They are used for basic input and output. printf is used like in the program above. It can take several parameters (arguments).

Basic Input/output: printf

- The Syntax of the printf function is as shown below:
- printf(Control string, arg1,arg2,…,argn);
- The **control string** refers to a string that contains the formatting information (how the output should be arranged and the type of data in the output) and *arg1…argn* represent the individual data items to be output.

Basic Input/output: scanf

- used to read data entered at the keyboard.
- Its syntax is similar to that of printf
- scanf(Control string, arg1,arg2,…,argn);
- The **control string** contains required formatting information (the format of the data that should be entered).
- arg1…argn are the variables where the entered information will be stored. Each of the variable identifier is usually preceded by an ampersand e.g. &agr1. This gives the address in memory of the variable. The data entered is stored at that memory location.

Formatting output

- %d = formatting command that prints the output as a decimal integer
- %5.2f” = formatting command that prints the output as a floating point integer with five places in all and two places to the right of the decimal point.

Example

Example: *The following example is used to illustrate the use of the input output functions. We add Two numbers by allowing the user to enter them on the keyboard, we then calculate the result and output it to the user.*

Example

```
#include <stdio.h>
void main()
{
/*declare variables to hold the two numbers to be added and the answer (integers)*/
int num1,num2,sum;
// Ask the user to input the two number – use printf
printf(“Enter the First Number:”);
scanf(“%d”,&num1);
printf(“\nEnter the second number:”);
scanf(“%d”,&num2);
//add the two numbers
sum=num1+num2 ;
//output the sum.
printf(“\n%d+%d=%d”,num1,num2,sum);
}//end
```

```
/* program for addition */
#include<stdio.h>
Int main()
{
int number;
float amount;
 number = 100;
amount = 30.75 + 75.35;
printf("%d\n", number);
printf("%5.2f", amount);
}
```

8. C program compilation process

Computer Program Compilation Process

A computer program compilation process involves the following five stages;

1. Lexical analysis: the source program is transformed into tokens. During the transformation all whitespaces and comments are discarded. All character sequences that do not form valid tokens are discarded and an error message is generated.

2. Syntactical analysis: analysis to ensure the program syntax is appropriate to the language. Failure results in an error message being generated.

Computer Program Compilation Process

3. Type / Semantic checking: responsible for ensuring that the compile-time semantic rules of the language are enforced. An error message is generated if the semantic rules are violated

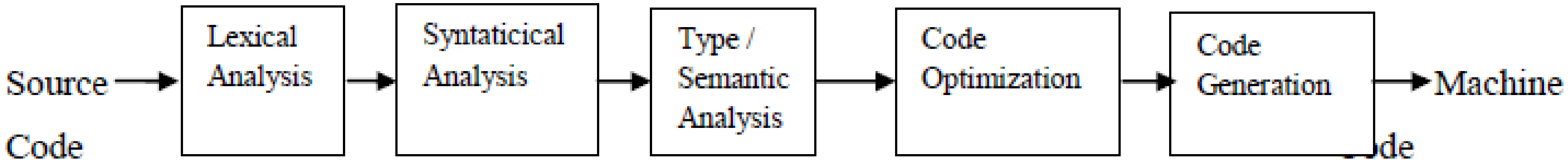
4. Code optimization: improves the intermediate code based on the target machine architecture

5. Code generation: target machine code is generated.

Computer Program Compilation Process

The first three stages are concerned with finding and reporting errors to the programmer, while the last two are concerned with generating efficient machine code to run on the targeted computer.

Computer Program Compilation Process



Source Code

Lexical Analysis

Syntaticical Analysis

Type / Semantic Analysis

Code Optimization

Code Generation

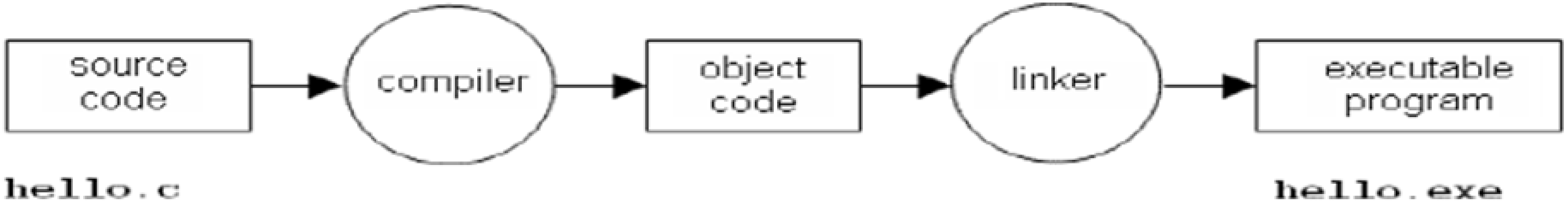
Machine Code

Steps involved in executing a C program

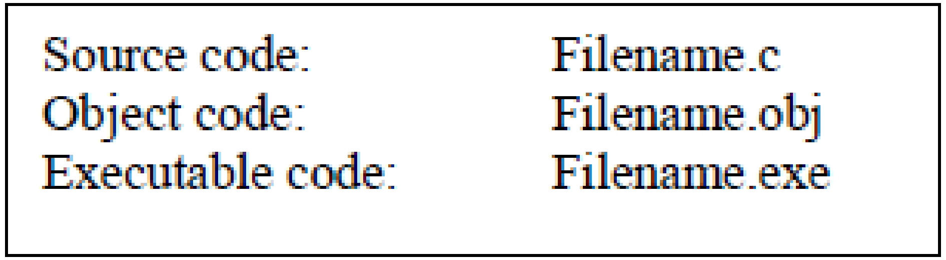
The steps involved in executing a C program includes;

- Creating the program
- Compiling the program
- Linking the program with functions that are needed from the C library
- Executing the program

Steps involved in executing a C program



C Program File naming convention



Source code:	Filename.c
Object code:	Filename.obj
Executable code:	Filename.exe

C Libraries

- In C, a library is a set of functions contained within a single "archive" file.
- Special functionality is provided in the form of libraries of ready-made functions.
- Libraries are files of ready-compiled code which we can merge with a C program at compilation time.
- Libraries provide frequently used functionality and, in practice, at least one library must be included in every program: the so-called C library, of standard functions.

Including Library files in C Program

- The most commonly used header file is the standard input/output library which is called stdio.h. This belongs to a subset of the standard C library which deals with file handling and provides standard facilities for input to and output from a program.

Including Library files in C Program

Example library files

- Stdio.h, (printf() function)
- math.h (for mathematical functions) etc.
- conio.h (for handling screen out puts such as pausing program execution getch() function)

The format for including the header file is **#include header.h**

Example: #include<math.h>, #include<stdio.h>