

Chapter

1

WELCOME TO COMPUTER AND PROGRAMMING WORLD!

1.1 Why Computers?

Imagine that you are at Mumbai V.T. station at 2 o'clock in the midnight. Sleeper class tickets for Delhi express are fully booked and only first class tickets are available. You need Rs. 2,000 more to purchase a first class ticket. Where can you get these 2000 rupees at midnight? Of course! ATM. Thanks to a computer technology which has made human life easier and luxurious!

Computers have become an integral part of the human life now-a-days. **Automated Railway Reservation System** is a well known practical example of applications of a computer. You can book railway ticket from internet at anytime from anywhere. Computers are used in **hospitals** to monitor and track patients, doctors, other staffs etc. You can play various **games** on a computer. Application software like **Microsoft word** can be used to create various types of documents. In a nut shell, computer technology is a boon to human life and hence study of computers is important for everyone.

Let's add two numbers, say, 5 and 12. Your brain can do this calculation very easily. Now, tell me what is the addition of 13579 and 2468? It's difficult for a normal brain to process it quickly. However, computer can perform this addition in a fraction of second. Further, suppose you have to add at least such 1000 numbers, this repetitive calculation job will be tedious and there can be mistakes. One can definitely rely on a computer to do this job in fraction of seconds and that too without any mistakes.

2 C for Beginners

The great scientist, Blaise Pascal's father was working in a tax department. He had to spend nights in doing calculations. He faced similar tedious situation of repetitively adding tax values and hence invented the calculator in 1643. It was used for calculation purpose till 1799 in Europe. Then, Charles Babbage, famously known as a **Father of a computer**, developed analytical Engine performing multiplication and division.

1.1.1 What is a Computer?

Computer = compute + er

Computer is a general purpose machine (electronic device) that does computations million or even billion times faster than normal human being. The calculation can be of arithmetic or logical¹ type.

OR

A computer is a general purpose device that can accept data (input), manipulate it (process) according to specific instructions, generate results (output) as a result of the processing and also (optional) stores the results.

Atanasoff-Berry invented First Electronic Digital Computing Device in 1937 that was capable of solving upto 29 simultaneous linear equations.

We have just read that computer does computations. It can add any 2 numbers like 13579 and 2468 in a fraction of a second. *Being a beginner, let's try to utilise a computer to add 2 simple numbers and gradually gain the knowledge to become a best programmer.* We need to give set of instructions in a logical order to perform this task of addition.

• **Set of instructions arranged in a logical sequence to perform a certain task is a program.**

OR

• **Program is a set of instructions arranged in a logical sequence to perform a certain task.**

Before using a computer, let us quickly understand the manual process^{*} of the same which everyone knows.

Input Data³ : 2 Numbers

Output Data⁴ : Number which is the addition of these 2 numbers

If you want to ask the addition of 2 numbers to your friend, following are the steps involved.

- You will tell 2 numbers to your friend which he/she will hear it by an ear.
- Friend's brain will store and add these 2 numbers.
- He/she will communicate the addition of these 2 numbers by speaking to you.

¹ Logical calculations will be discussed in the next chapter.

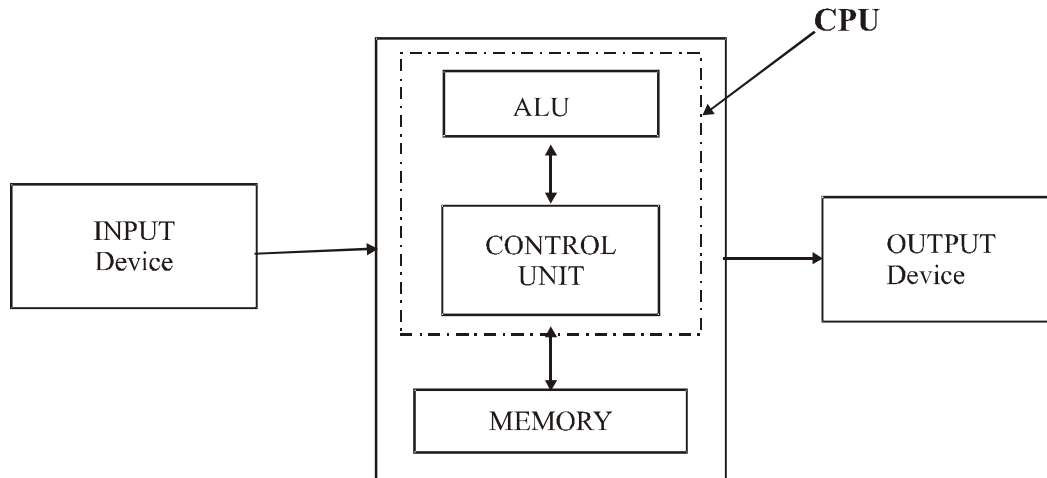
² Manual process is the process executed by a human being.

³ Data received by a computer from user is referred as an input data.

⁴ Data communicated (displayed) by a computer on a monitor, printer etc. is referred as an output data.

Ear, mouth and brain acts as input, output and processing unit of a human being respectively. As we want to get this execution done from a computer, let's understand input, output and other units of a computer.

1.1.2 Block Diagram of a Computer



Computer can be logically divided into following four parts.

1.1.2.1. Input unit

We need to input two numbers to a computer to get the sum of these two numbers. What are the parts of a computer that can receive this input data? Computer receives information through input devices like

- Keyboard
- Mouse
- Speaking to a computer
- Joystick
- Track Ball
- Scanner
- Network (One can access a file on the network located on the other computer.)

For the time being, beginner may consider keyboard, mouse as i/p devices.

1.1.2.2. Memory unit

Memory is a unit of a computer where data is stored.

In case of the above addition of two numbers program⁵, user enters two numbers as input, let us say through keyboard. When this data is successfully received by a computer, it is saved in a memory.

⁵ Program is a set of instructions arranged in a logical sequence to perform certain task. e.g. We need to write set of instructions in a logical sequence to add the numbers 13579 and 2468.

4 C for Beginners

Computer memories can be categorised as

- I. Primary Memory
- II. Secondary Memory

I) Primary Memory

Primary memory is directly accessed by a CPU. It is volatile in nature means data stored in it is lost if the power is switched off (Exception: ROM). Primary memories can be further classified as

i. RAM (Random Access Memory)

It provides random access to data stored in it. Data stored on RAM can be changed.

The main feature of RAM is that the time required to retrieve data from any memory location is constant irrespective of the memory location address. That's why it is known as RAM.

ii. ROM (Read Only Memory)

It is a read only memory. Once information is stored on ROM, it cannot be easily modified or erased.

II) Secondary Memory

It is not directly accessed by a CPU. It is a permanent storage of the data that is non volatile in nature. It is also cheaper and lesser in speed than primary memory. Secondary memory devices are magnetic tape, diskette, hard disk, CD ROM etc.

When you ask mobile number to someone, can you note it or just store it in your brain forever? It is not possible for a normal brain to store all the numbers. Hence, we note the data in a diary. **Here, human brain acts as a primary memory, while, diary plays a role of secondary memory.** We can retrieve the data from the brain very quickly as compared to the data written in the diary. Diary, notebook etc. acts as secondary memory for the human being.

When someone calls you, you hear it by ear and your brain takes certain action based on the message it receives. There must be certain programs existing in human memory by birth which contains instructions to recognize voice, to produce certain voice by mouth etc. These programs are always alive for normal human being and must be residing in read only memory of human being. Similarly, ROM stores the data that cannot be erased. ROM contains the programs that are required for the functioning of a computer. e.g. BIOS program starts a computer.

FAQ⁶. It is true that we cannot forget the image of our best friend even if we try our best to forget it. Can't it be an example of read only memory of human being?

When we revise certain study material, we won't forget it. However, if we do not revise it, we forget it slowly. Similarly, you will slowly forget the image of your friend if you do not remember or memorise him/her. Practically, it is unavoidable to remember the close friend and his/her image gets revised in our memory. Hence, we cannot forget it. Therefore, it is not an example of ROM of human being.

1.1.2.3. Central Processing Unit (CPU)

It is the overall administrator (co-coordinator) of a computer.

CPU has two main parts.

1. Control Unit
2. ALU

I. Control Unit

The control unit (CU) reads instructions from memory, decodes⁷ and executes.

When the instruction requires to store data into a memory, it receives the data from input device like a keyboard and stores it into a memory. When the instruction requires calculation, it passes the required information to the ALU.

It is also referred as a brain within brain because based on the result of the execution of the instruction by control unit, operations of other parts of a computer takes place.

CPU speed is a feature that tells how quick CPU can execute an instruction.

The speed of the CPU is 2.0 GHz. It means the CPU can execute 2×10^9 instructions per second. Many today's computers have multiple CPU's.

II. Arithmetic and logic unit (ALU)

It can perform arithmetic and logical calculations on data. For example, it can add two numbers together (in binary).

1.1.2.4. Output unit

Once this calculation is done by a computer, the calculated sum (output) can be sent to the following output devices.

⁶ **FAQ** stands for general/frequently asked question/questions. This questions generally pops up in reader's mind while reading or understanding a program. **This approach is used to effectively deliver the knowledge.**

⁷ Convert the instruction into binary language.

6 C for Beginners

- Monitor/VDU
- Printer
- To control other devices
- Network like internet
- Soundcard
- Film
- Video
- Robot arms etc.

For the time being, beginner may consider monitor, printer as output devices. Let's compare Primary and secondary memory, RAM and ROM.

Primary Memory	Secondary Memory
CPU can access it directly.	CPU cannot access it directly.
It is volatile (Exception: ROM).	It is permanent.
The speed is high.	The speed of accessing the data stored in secondary memory is slow.
The cost is more.	The cost is less.
RAM,ROM is an examples of primary memory.	Secondary memory devices are diskette, hard disk, CD ROM etc.

RAM	ROM
The data stored on RAM can be modified.	The data stored on ROM cannot be modified easily.
It stores data temporarily. When the machine is switched off, the data stored on RAM vanishes.	It stores data permanently.

1.2 Machine Language

Hope you have understood the basic concepts explained in previous sections.

Let's go one step ahead and try to use computer practically to add two numbers. It is necessary to give right instructions to a computer to add two numbers. Oh, but, can a computer directly understand the instruction "Add two numbers and show the result?" No. We must have to give this instruction in the language which computer can understand. Then, what is the language of a computer?

Computer can only understand binary language known as a machine language. Binary language has only 2 digits i.e. 0 and 1.

Hence, the following instructions need to be written in a machine language to add two numbers.

Do not try to understand the following instructions; just see that how complicated and messy instructions (code) are written in a machine language to add two numbers.

Binary Instruction	Meaning
010 0 001101	Store the value of the Accumulator in memory location 13.
001 1 000101	Load the value 5 into the Accumulator.
010 0 001110	Store the value of the Accumulator in memory location 14.
001 0 001101	Load the value of memory location 13 into the Accumulator.
011 0 001110	Add the value of memory location 14 to the Accumulator.
010 0 001111	Store the value of the Accumulator in memory location 15.
111 0 000000	Stop execution.

Machine language is the basic language of a computer which is directly understood by it. It consists of binary numbers that tells computer to execute basic operations like printing, adding 2 numbers etc.

Advantages of Machine Language:

1. Though it is tough to write programs in machine language, the execution is fast.
2. It is directly understood by a computer. Other languages are not directly understood by a computer.

Disadvantages of Machine Language:

1. As one has to write the instructions in the number format, chances are always there of doing mistakes in the program.
2. Machine languages are specific to machine. It means the program executed on one machine cannot be executed on the other machine because the binary representation for various instructions are different for different machines.

1.3 Assembly Language

Human being is inventive and innovative by nature. Machine language instructions are in the form of the numbers (that too only 0 and 1) and difficult to read. People started to use English like abbreviations (Mnemonics) to give instructions to a computer to perform a particular task.

Binary instruction or set of instructions are mapped to English abbreviations known as mnemonics.

You can just have a glance on below equivalent assembly language instructions for the above machine language instructions.

```

mov      a1,0Ah
mov      b1,14h
mov      eax,a1
add      eax,b1
mov      c1,eax

```

Assembly language is a machine-orientated language in which mnemonics (symbolic names) are used to represent each machine-language instruction. It is a programming language that is one step above machine language. Each CPU has its own specific assembly language. Hence, assembly language program executed on one machine cannot get executed on the other machine of different processor (CPU).

Advantages:

1. Assembly language is easy to write and also chances of making mistakes in writing instructions are less as compared to machine language programs.

Disadvantages:

1. Assembly language instructions (code) is not directly understood by a computer. It needs to be converted into machine language instructions by another program called as an assembler.
2. Though easier than machine language, writing programs in assembly language is complicated and messy.

Machine and assembly languages are low level languages.

FAQ. Does computer directly understand assembly language instructions?

No. Computer can only understand binary language. Hence, the assembly language instruction needs to be converted into a machine language instructions.

Assembler is a program that converts assembly language instructions into machine language instructions.

As seen above, one has to write so many instructions even to write a simple program. Further, middle and high level languages are developed to speed up the programming

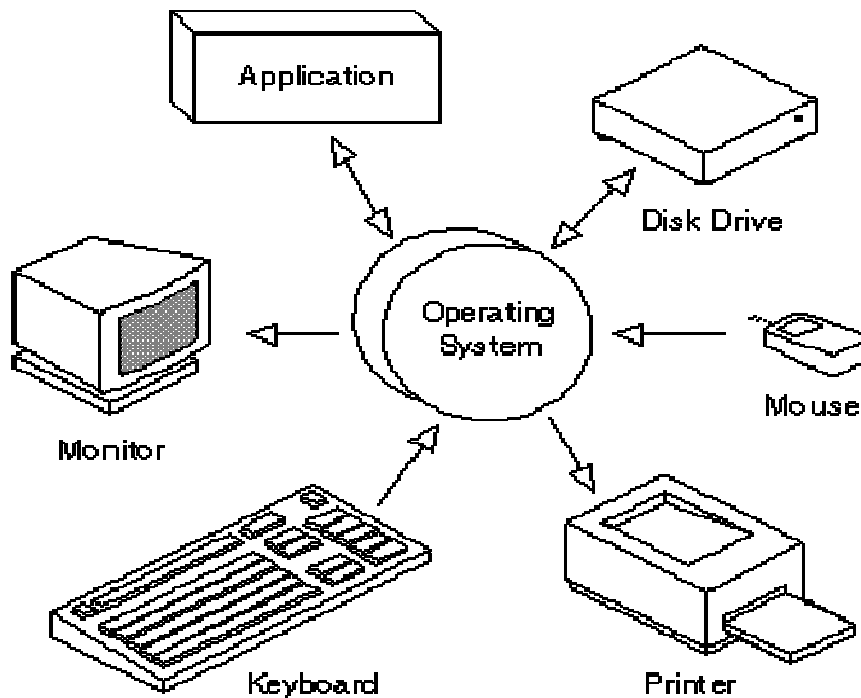
1.4 Operating System

An operating system (OS) is a set of computer programs that manages the hardware and software resources of a computer. It is heart of the computer.

Operating systems perform basic tasks, such as recognizing input from the keyboard, sending an output to the display screen, keeping track of files and directories on the disk, controlling peripheral devices such as disk drives and printers etc.

Windows, Unix ,Linux ,Macintosh are the operating systems.

Unix is a multiuser system. Many users can work on the same machine by connecting through terminals. It means all the users share the CPU and memory of one machine at the same time.



1.5 C Language

1.5.1 History

Operating system UNIX was initially written in assembly language. It was supporting multiple users to use the same system through many terminals. From different terminals, programmers at AT&T Bell labs were playing **Space Travel computer game** on mainframe machine where Unix was installed. As mainframe machine was not fully supporting this game, it was decided to port the game on PDP-7⁸ machine and then PDP-11. PDP machines lacked operating system. Hence, original program of Unix system was needed to be converted into the compatible PDP assembly language which was a tough task. In order to solve this issue of the portability, **Dennis MacAlistair Ritchie** wrote a new language known as C language. C has inherited most of the features from its antecede B language.

C was developed at AT&T Bell Laboratories in 1972 where Dennis Ritchie was working. Currently, Dennis Ritchie is heading System Software Research Development Department @ Lucent Technologies, USA.



1.5.2 Why one should use C language for programming?

If you do not understand below features, don't get upset! You will understand these features in the later part of this book.

- i. **Portable:** Portability of the program was a major issue. Programs written on one machine were not working on the other machines. e.g. Assembly language is not portable because programs written for specific machine cannot be used for other machines. C language is portable. Program written in C language can be compiled and executed from every processor with minor changes if any. Hence, C solves issues of portability.
- ii. C programs can be written systematically as it incorporates features of **structured programming**. It improves the readability and subsequently helps at the time of enhancement, maintenance, debugging etc. This will be discussed in chapter 2,3 and 4.
- iii. It is a middle level language and has **best features of low level and high level language**.
- iv. C is fairly **concise**. e.g. C has prefix and postfix⁹ operators that expresses computations concisely.
 - v. Currently, C is the most commonly-used language for embedded systems.
- vi. **Pointer** feature of C allows programmer to directly access the data stored in a memory. Programmer can manage data in a better manner. The speed of execution is faster because programmer can access the data from memory locations directly.
- vii. C also allows the programmer to produce programs that are impossible to understand. This feature is used by **product companies** so that instructions written for the product should be difficult to understand to others.

⁸ PDP-7 and PDP-11 are the machine names.

⁹ It will be discussed in the chapter 3.

1.5.3 Embarking C Programming: First C Program

I know you are eager to write first C program, aren't you? Welcome to the world of C programming!

Let's write a C program that performs addition of 2 numbers (i.e. 13579 and 2468) and display it. Though it is a simple program, it will give you exposure to the various aspects of C programming.

Before writing any program, it's better to understand how the same task can be done manually.

You will do this calculation manually as below.

1. You will remember value 13579 as number1 in your memory
2. You will remember value 2468 as number2 in your memory
3. You will add number1 and number2.
4. You will say "The sum is = 16047"

In a programmer's language, the above mentioned steps can be written as below.

1. Reserve a memory to store a value 13579.
2. Store a value 13579 in the above reserved memory.
3. Reserve a memory to store a value 2468.
4. Store a value 2468 in the above reserved memory.
5. Reserve memory to store the addition of 13579 and 2468.
6. Add these 2 numbers and store the sum in the above reserved memory.
7. Display the addition of these 2 numbers (sum).

Don't get scared with the word "Algorithm". The above 7 steps are collectively known as an "Algorithm" for a program that adds 13579, 2468 and displays the result. History of algorithms is given in the General Information section at the end of this chapter.

Algorithm is a set of steps arranged in a logical sequence to perform a certain task.

Let's convert every step in the algorithm into an equivalent c language instruction. Hold on! It's important to know following 2 thumb rules before writing any C program.

Rule 1:

Remember forever, every C program starts with the main function block as below.

```
int main()
{
    return 0;
}
```

It's not possible to write a C program without main function block.

Rule 2:

Every variable must be declared at the beginning of a block¹⁰. i.e. at the opening curly brace of a block. C program can be easily developed by just translating steps in the algorithm into equivalent C instructions.

¹⁰ Block begins with { and ends with } as shown for main function.

12 C for Beginners

Step1: Reserve a memory to store a value 13579.

As 13579 is an integer type of data, we need to reserve a memory that can hold (store) an integer value. Following instruction reserves a memory to hold an integer value.

```
int Number1;
```

int is a keyword. Every keyword has a special meaning. int keyword specifies to allocate a memory to hold an integer value.

Number1 is referred as a integer variable. Integer variable can store 1 integer value at a time. In English, every statement is terminated by a full stop. Similarly, in C, every instruction is terminated by a semicolon (;).

Step2: Store a value 13579 in the above reserved memory.

```
Number1=13579;
```

Number1

13579

0x0042103c

Here, the value 13579 is stored in a memory location 0x0042103c¹¹ (reserved memory in step 1). Number1 is referred as a variable name.

Step 3: Reserve memory to store a value 2468.

Similarly, Number2 is declared as below.

```
int Number2;
```

Step 4: Store a value 2468 in the above reserved memory.

```
Similarly,  
Number2=2468;
```

Number2

2468

0x00422031

Step 5: Reserve memory to store the addition of 13579 and 2468.

Similar to step 1 and step 3, variable sum is declared as below.

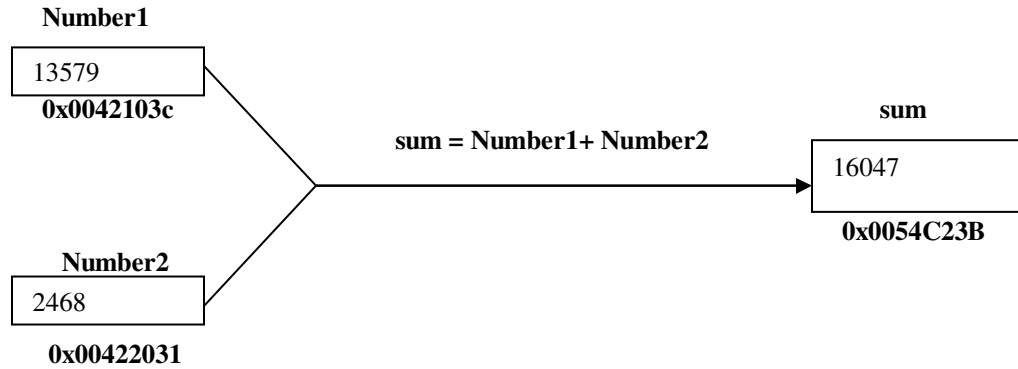
```
int sum;
```

As per thumb rule 2 discussed earlier, variable must be declared at the beginning of a block.

Step 6: Add these 2 numbers and store the sum in the above reserved memory.

```
sum = Number1 + Number2;
```

¹¹ This number is presented in the hexadecimal system format.



Step 7: Display addition of these 2 numbers (sum).

`printf("%d",variablename)` is a standard command (function or instruction) that displays the value stored at the memory location `variablename` on the output device like a monitor. As we want to display the contents of a variable `sum`, the instruction is

```
printf("%d", sum);
```

It is mandatory to declare a function (command) before we use it in a C program. Declaration of a `printf` function is given in the header file¹² `stdio.h`. Following is the format to include any standard file like `stdio.h`

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

When compiler compiles the instruction containing `printf` function, it tries to find the meaning of the symbol `printf`. As the declaration of `printf` is given, it compiles this instruction successfully.

The complete program of addition of 13579, 2468 and displaying the result can be written as below.

Program 1.1:

```
#include <stdio.h>
int main()
{
    int Number1;           // step 1 Variable Declaration
    int Number2;          // step 3 Variable Declaration
    int sum;               // step 5 Variable Declaration

    Number1=13579;        // step 2
    Number2=2468;         // step 4

    sum = Number1 + Number2; // step 6
    printf("%d",sum);      // step 7
    return 0;
}
```

Output:

16047

¹² Header file contains the declarations of a standard functions.

FAQ:

Q. Every memory address is uniquely identified by a number like 0x0042103c. Then, why we are referring allocated memory by a name?

As it's difficult to refer memory locations by a number, variable names are used to refer memory locations where data is stored. As the program size increases, it becomes difficult to refer memory locations by an address. Hence, names are used. It makes easier to read and write a program.

Q. What is a keyword? What is an int keyword? Enlist the keywords in c.

Keywords are identifiers (names) reserved by the language for special use. Every keyword has a special meaning. C has 32 keywords as given in the below chart.

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

Q. Can I use variable names a, b instead of Number1, Number2?

Yes, you can use it. However, as the program size increases, it becomes difficult to understand the meaning of the data the variable is containing. It's better programming practice to use relevant variable names to improve the readability of a program.

Q. What is a function?

Function¹³ is a set of instructions arranged in a sequence to perform certain task.

Function is component of a program that provides particular functionality. e.g. printf function used in above program is used to display the value of the variable sum on the screen. Every function has input and gives output accordingly. We will discuss about printf function later.

Q. What is the meaning of the “// step 1 Variable Declaration”?

It is a comment.

Q. What is a comment? What are the different types of comments?

It is information written in a source code that is ignored by a preprocessor¹⁴ when the source file is pre-processed.

¹³ Function will be discussed in the 4th chapter in detail

¹⁴ It will be discussed in the next chapter.

Comments improve the readability of the program. It is easy to read and understand a program with comments. Especially, comments help when a new programmer works on the source code and also at the time of maintenance¹⁵/enhancement of the application.

There are 2 types of comments.

i. Single line comment:

Any text written after // is considered as a single line comment.

ii. Multi line comment:

Multi line comments begin with /* and ends with */

Program 1.2:

```

/* This program does addition of 13579 ,2468 and display the output. As a beginner, you should
thoroughly grasp it. */
#include <stdio.h>
int main()
{
    int Number1;           // step 1 Variable Declaration
    int Number2;           // step 3 Variable Declaration
    int sum;               // step 5 Variable Declaration
    Number1=13579;         // step 2
    Number2=2468;          // step 4
    sum = Number1 + Number2; // step 6
    printf("%d",sum);      // step 7
    return 0;
}

```

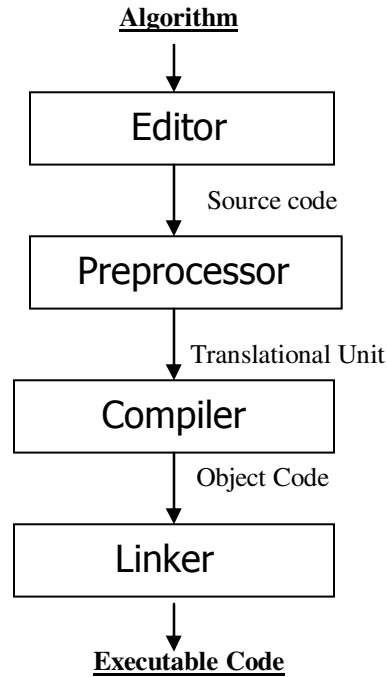
Readers are advised to compile and execute program 1.2 before reading below section.

1.6 Typical C Language Environment

Following are the steps involved in the development of an executable file from writing a C language program in an editor.

- **Editor** is a program (tool) where one can write instruction/ instructions.
- **Preprocessor** searches included files and copies the content into a translation unit which is its output. It also removes the comments written in the source program file.
- **Compiler** checks the grammar of every instruction, produces corresponding binary instructions, creates corresponding object file and stores these translated binary instruction in it. In case of the standard readymade functions like printf, executable instructions are present in other object or library files, it creates a missing hole in the corresponding object file.
- **Linker** searches the binary instructions for the referred symbols like printf and finally creates an executable file.

¹⁵ Maintenance is a process in which defects in the program are found and fixed. Concept of maintenance of a vehicle is similar to a maintenance of a program.



1.6.1 Editor

Editor is a standard program that allows you to edit your source program code. Program is written in a file in an editor.

1.6.2 Preprocessor

Once the source program is written in an editor, preprocessor substitutes the text for included files, removes comments from the source file and produces translation unit as output.

In case of this program, preprocessor searches and copies the contents of the header file `stdio.h` into the translation unit. Moreover, it removes all the comments used in the source file where program is written. The output of the preprocessor (i.e. translational unit) unit is an input to the compiler.

1.6.3 Compiler

Every language has a grammar. Once the program is written, we need to check whether it follows grammar rules. In case of English, if someone says to you

“program to required is a compiler compile”.

You won't understand it. This is actually “Compiler is required to compile a program.” in a correct English format as per the grammar rules. Hence, we must have to give instructions to a computer which follows language grammar rules. e.g. Every instruction should be terminated by a semicolon (;).

If you forget to write a semicolon at the end of any instruction in the above program, compiler produces syntax error. Let's remove a semicolon of any line, let us say line 6 and compile the program, you will get syntax error. If the program doesn't obey one or more grammar rules, it gives syntax error.

Compiler is a standard program that checks whether every instruction in the program is written as per the (C) language standard grammar rules or not, translates it into machine language code and also provides references for referred symbols like printf. Compiler writes its output in the object file.

In case of the above program, compiler provides references to the printf command (function). Object file contains a missing hole referring the printf function.

In a nut shell, compiler is a program that takes translation unit (file) as input and produces object file as output. The extension of object file is obj.

1.6.4 Linker

Linker receives object code file (generated by a compiler) as input. It searches the binary instructions for the referred functions (symbols) like printf and assembles all parts of the program into an executable file which is its output. Executable file ends with an extension .exe.

Executable file contains only binary instructions/machine language instructions that can be directly understood and executed by a computer.

Program	Input	Output	Functionality
Editor	Algorithm	Source Program	Instructions in C language are written in a file by using an editor. A file where a program is written is known as a source program .
Preprocessor¹⁶	Source Program	Translation Unit	Preprocessor takes source file as input and produces translation unit as output where it copies the contents of the header file. It also removes the text written inside comments and generates a file known as a translation unit.
Compiler	Translation Unit	Object Code File	<p>It checks whether every instruction in a program is written as per the C language standard grammar. It generates compile time error/warnings in case of any discrepancy.</p> <p>If there is NO grammatical error in the instructions in the source file, it produces binary instruction for the same instruction in a program and produces an object file. Extension of object file is .obj.</p> <p>In case of referred symbols like printf function, it won't generate binary instructions. It just produces holes in the object file for the same.</p> <p>*Binary instructions for the referred functions (symbols) are present in other object or library files. It will be discussed in a function chapter in detail.</p>
Linker	Object Code File	Executable File	Linker searches binary instructions for the referred functions like printf in the other object or library files and assembles the same to produce a complete file containing binary instructions. This file is known as an executable file.

¹⁶ Details of a preprocessor are discussed in preprocessor chapter.

Assume that you have saved this program in a file `add.c` in a directory, say “programs” in a C drive. The complete path of the program is “C:\ programs \add.c”. Please refer the appendix for more information to generate executable¹⁷ file. Oh, but What is a directory?

We organize various books in various shelves in a systematic manner. It’s possible to locate a particular book because of this systematic arrangement of books. Similarly, directory provides us a systematic way to organize data in a computer.

Directory is a location on a computer where files and other directories (sub-directories) can be stored.

A directory helps to manage the data systematically on a computer.

If you keep your books and other belongings all over the house, it will be tough time for you to locate it later. Similarly, if you save your C programs at different places on a computer, you will mess up and will not be able to locate it when you need it. Better way is to save all C programs in the same directory and retrieve them as and when required.

Every C source program file ends with `.c` extension¹⁸.

Congrats!! Finally, you have created executable file `add.exe`. This is your first achievement. This executable file contains machine language instructions which add 2 numbers as defined in the above program and displays it on the console (screen). Now, you can click on the executable `add.exe` and see the output on the screen.

Following errors can cause during the conversion of C program into an executable file.

1. Compile time / Syntax Error

When any instruction in the program is not as per the C language grammar rule, compile time error occurs. e.g.

- i) When semicolon is not present at the end of any instruction.
- ii) When any curly brace is missing.

There are many ways by which this type of error can occur. You can observe this error by removing semicolon from an instruction or by erasing one of the curly brace in the above program.

2. Linking Error

When linker is unable to locate definition of any unknown symbol used in a program, linking error is generated. e.g. When linker is unable to locate the definition of referred symbol like `printf`, it produces linking error.

You can observe linking error by changing `printf` to `xprintf`.

¹⁷ It will be discussed in next section.

¹⁸ It is possible to use other extensions to c language source files which are not covered.

3. Runtime Error

Runtime error occurs while executing any instruction in the executable file. e.g. Dividing a number by zero. i.e `int a=0; int k=90/a;` Execution of this instruction will give you runtime error.

4. Logical Error

The aim of the above program 1.1 is to display the sum of the 2 numbers. i.e. 13579 and 2468. If we write the instruction `printf("%d", Number1)` instead of `printf("%d", sum);` you will get incorrect result. If you use `-` (minus) symbol instead of `+` (plus) symbol by mistake, it will give you wrong results which is also a logical error.

- Logical error can occur if and only if the program successfully passes compilation, linking and execution of the program.
- Runtime error can cause if and only if no error is caused during compilation, linking.
- Linking error can cause if and only if no error is caused during compilation.
- Compilation error can cause if there is no error observed during preprocessing of the program.

Do not get discouraged while compiling your first program. You will learn more if you come across many errors initially.

You must be familiar with the name **Thomas Edison** who invented the bulb. He failed 10,000 times to produce an electric bulb and then successfully invented first electric bulb. When someone asked him about his failure, he said "Now, I know 10,000 different ways by which bulb cannot be produced."

Take motivation from him and do not get nervous even if you initially fail many times to compile a C program. You will understand different types of errors that become hurdle to compile a C program. This experience will help you to develop big sized programs with ease.

1.7 Points to Remember

1. The source code program (e.g. `add.c`) is stored in a secondary memory. i.e. inside `C:\` or any other drive. Hard disk is an example of secondary memory.
2. When executable file (e.g. `add.exe`) is executed (just click on it), it gets copied into the RAM.
 - **The program that copies the executable file from secondary memory to a primary memory (RAM) is called as a loader.**
3. Comments are ignored by a preprocessor and not present in the translation unit which is the output of the preprocessor. The output of the preprocessor is an input to the compiler.
4. Remember, we cannot define one multiline comment inside other multiline comment. However, multiline comments can contain single line comment/comments.
5. C language is a case sensitive. Following instruction won't compile.

```
INT Number1=13579;
```

This is because compiler won't understand the meaning of the word INT and it will generate a compile time error.

6. You can always declare and assign the value of a variable in the instruction itself.

```
#include <stdio.h>
int main()
{
    int Number1=13579; // Definition of Number 1
    int Number2=2468; // Definition of Number 2
    int sum= Number1+ Number2;
    printf("%d", sum);
    return 0;
}
```

You compile the above program and execute the same for better understanding. Further, you can combine three declarations in one line as below.

```
#include <stdio.h>
int main()
{
    int Number1,Number2,sum;

    Number1=13579;
    Number2=2468;
    sum = Number1+ Number2;
    printf("%d", sum);
    return 0;
}
```

Even you can write the same program as below.

```
#include <stdio.h>
int main()
{
    int Number1=13579,Number2=2468,sum;
    sum = Number1+ Number2;
    printf("%d", sum);
    return 0;
}
```

7. Variable names correspond to locations in the computer's memory. In memory, every variable has assigned a particular address. Every variable has a name, a type, a size and a value.

```
type variable_name;
```

Whenever a new value is placed into a variable, it replaces (and destroys) the previous value.

22 C for Beginners

8. To make the program add.c more user friendly, you can change the printf instruction as below.

```
printf(" The sum is %d",sum);
```

Whatever characters we write inside the quotes of the first parameter of the printf function gets printed as output. If any format specifier like %d encounters, then printf function reads the value stored at corresponding variable of the data type indicated by that format specifier. e.g. After printing

"The sum is", format specifier %d occurs. The corresponding variable to %d is sum. Hence, the value stored by a variable sum is displayed as integer.

printf("Enthusiasm is a key to success.") displays "Enthusiasm is a key to success" as output. We will discuss printf function in Input Output chapter in detail.

9. When variable is declared and not explicitly initialised, it contains earlier existing value in that memory location. This value can be any meaningless value. It is known as a **junk or garbage value**.
10. C is a freeform language. It means you can write 1 instruction on 1 or more lines as shown below.

```
int  
Number1;
```

Q. Guess the output of the following programs.

I.

```
#include <stdio.h>  
int main()  
{  
    int Number1;  
    printf("%d", Number1);  
}
```

Output:

Any value (Junk Value)

When a variable is declared inside a function (in this case main function), memory is allocated by a compiler. As explicitly no value is specified, earlier existing value gets displayed as output when we try to display the content of the variable that is not explicitly initialised.

II.

```
#include <stdio.h>  
int main()  
{  
    int Number1=13579;  
    int Number2=2468;  
    int Number1=4517;  
    int sum= Number1+ Number2;  
    printf("%d", sum);  
    return 0;  
}
```

Output:

The output is a compile time error because the variable Number1 is declared twice in the same function (main in this case) which is not allowed in C as per grammar rule.

```
III. #include <stdio.h>
int main()
{
    int Number1=13579;
    int Number2=2468;
    int sum= Number1+ Number2;
    sum=156;
    printf("%d", sum);
    return 0;
}
```

Output:

156

The addition of 13579 and 2468 is first stored in a variable sum. However, the next instruction sum=256; changes the contents of the variable sum to 156. Hence, the value stored in the variable sum i.e. 156 is displayed as output.

```
IV. #include <stdio.h>
int main()
{
    int Number1;
    int Number2;
    int sum;
    Number1=13579;
    Number2=2468;
    sum= Number1+ Number2;
    printf("%d", sum);
    return 0;
}
```

Output:

16847

In case of the above program variables Number1, Number2 are declared at the beginning of the main function block. The values 16847 and 2468 are stored later which is fine.

```
V. #include <stdio.h>
int main()
{
    int Number1;
    int Number2;
    Number1=13579;
    Number2=2468;
    int sum= Number1+ Number2;
    printf("%d", sum);
    return 0;
}
```

Output:

This will give you a compile time error as the variable sum is not declared at the beginning of the main function block.

In C language, it is mandatory to declare variables at the beginning of a function (block). Otherwise, compile time error is generated.

```
VI. #include <stdio.h>
int main()
{
    int Number1=13579;
    int Number2=2468;
    int sum;
    sum= Number1+ Number2;
    printf("%d", sum);
    return 0;
}
```

Output:

16047

Variable can be declared and initialised with a value in the same instruction as above. It is known as a definition of a variable.

```
VII /* This is the first program written by me in C language.
/* Add.c */
I have learned so many fundamentals because of this program.
*/
#include <stdio.h>
int main()
{
    int Number1;
    int Number2;
    int sum;
    Number1=13579;
    Number2=2468;
    sum= Number1+ Number2;
    printf("%d", sum);
    return 0;
}
```

Output:

As a multiline comment is declared inside other multiline comment, hence it is an error.

FAQ:

Q. Can computer directly understand the above add.c program and display the addition of 2 numbers as output?

Basic language of a computer is a binary language and it can directly understand binary instructions. Computer cannot directly understand C language instruction. Hence, we need to convert these instructions in binary language.

Q. Why computer can directly understand 0 and 1 and not any other numbers or alphabets?

Computer is made up of transistors which are used in on-off mode. When transistor is on, it is treated as 1 and when transistor is off, it is treated as 0. That's why computer can only understand 0 and 1.

Q. Who finally executes binary instructions inside executable file?

CPU reads every instruction from the instruction register and executes it one by one.

Q. Why should we use C language for writing this program, let us say, addition of 2 numbers as compared to machine and assembly language?

If you use machine language to write this program, you had to write instructions in machine language which is cumbersome and error prone.

If you use assembly language to write this code, you would have to write instructions in assembly language using mnemonics. Number of the instructions in assembly language are less than machine language. However, still, it is tedious job.

If you see C language program, it is hardly few lines of code as compared to programs written in machine and assembly language instructions. This reduces the chances of doing mistakes while writing a program because the size of the program is small.

The biggest advantage of C language is that program developed in C language is portable. It means it can be compiled on different processors and executed. Machine and assembly languages are specific to the machine.

Q. Why C is a middle level language?

C is often called a middle level computer language, because it combines the best elements of high level languages with the control and flexibility of assembly language.

High level features

- Data types: C supports built in, derived and user defined data types like any other high level language.
- Portable: It is possible to compile and execute C language program on any machine.
- Partially support abstraction: It supports structure, function concept which is abstraction.

Low Level features

- o It is possible to access any memory location.
- o Type conversion is allowed. e.g. int can be treated as float.
- o Number of keywords are only 32.
- o No run time checking. e.g. When any value outside array bound is accessed, it doesn't give any error many times.

As a middle level language, C allows the manipulation of bits, bytes and addresses –the basic elements with which a computer functions.

C also provides programmers with many methods to achieve the same thing. For Example, while typical high level languages only allow a single way to access an array element addressing an array element in C/C++ can be done in 4 different ways! This means programmer does not have to stick to single method that he/she does not like. C/C++ allows programmers to express their creativity in their own different

Note: One can easily understand the meaning of above features till chapter 7.

Q. Where one can see contents of stdio.h file?

The easiest way is to search a file stdio.h in your computer by using “Find” command on Windows. Otherwise, you can directly search in the directory where C language compiler is installed.

Q. Where C language grammar rules are defined?

The C language rules are defined in a standard. First standard published was known as C89 or ANSI C. The latest standard is C99. However, current compilers do not support it.

Note: All the programs in this book are compiled using Microsoft Visual Studio.

Q. Computer understand only 0 and 1. Then, how Number1 is stored in a decimal format as shown in below figure?

Number1

13579

0x0042103c

The above representation is only for understanding purpose at a beginner level.

Computer understand only 0 and 1. Data is stored in a memory. Memory is divided into units referred as memory locations. Each memory location can store maximum 8 bits. Every memory location is identified by an unique address. e.g. 0x0042103c is an address. Address is an unique integer number.

When any data is stored in a computer, memory should be allocated. In case of an integer, 4 consecutive memory locations are allocated. i.e. the size of an integer data type is 4.

The integer value 13579 is converted into binary form. i.e.00110101 00001011. Hence, it is actually stored as

00000000	00000000	00110101	00001011
0x0042103f	0x0042103e	0x0042103d	0x0042103c

Little Endian Address Representation of 13579

Integer value can be positive or negative. Hence, the last bit i.e. 32th bit indicates the sign of the integer value stored. If 32th bit is 0, the integer value is positive, else it is negative.

1.8 Deep Knowledge Section¹⁹

Q 1. Why RAM is faster than hard disk memory?

RAM is IC based data storage. Hence, it allows to access the stored data in any order- means randomly and **without any physical movement of the storage medium / reading head**. In case of secondary memory like hard disk, data is accessed sequentially. Hence, it takes time to move the read/write head from one location to other memory location from where data needs to be read. Physical movement of head and physical location of data causes more access time for secondary memory data access.

Q 2. Guess the output of the following program?

```
I. #include<stdio.h>
int main()
{
    unsigned int a1=4147483641;
    unsigned int b1=2147483642;
    unsigned int c1=0;
    c1=a1+b1;
    printf("%d" ,c1 );
    return 0;
}
```

Output:
1999999987

Are you surprised? As mentioned above bytes, unsigned int has size of 4 bytes on the windows. Hence, it can store maximum value $2^{32} = 4294967296$. However, the addition of above numbers is 6294967283 which is beyond the capacity of storage limit of a variable c1 which can contain maximum value 4294967296. Therefore, 4294967296 is subtracted from the value 6294967283 that is equal to 1999999987. Hence, 1999999987 is displayed on the screen.

You can simulate the above simulation for various data types that you will come across soon.

Q 3. Is the following instruction declaration or definition of a variable?

```
int var;
```

It is definition of a variable and not a declaration. Variable is said to be defined when memory is allocated and it is initialised with value.

In the above case, compiler allocates memory location and refers it as var in the program. Actually when the memory location is allocated, this memory location contain some unknown data (value) known as junk value.

¹⁹ Deep knowledge Section should be read after developing average level in C. Beginner may skip this section in first reading.

Q 4. What is the correct declaration of a main() function?

```
int main();  
or  
int main(int argc, char* argv[]);
```

main function always returns an integer value.

- If the program successful executes, it returns zero to the operating system.
- And in case of any run-time error, it returns integer error code number to the operating system.

It is one of the common mistakes done in many books. Following declarations are undefined.

```
void main();  
void main(int argc, char* argv);
```

Q 5. How computer starts?

- i. When you turn on your computer, the **microprocessor** passes control to the **BIOS** program, which is always located at the same place on EPROM.
- ii. Bios does power-on self test (POST) to make sure all a computer's components are operational.
- iii. BIOS looks for the system files at a specific place on your hard drive and copies information from it into specific locations in RAM. This information is known as the **boot record** or Master Boot Record.
- iv. The boot sector on a disk is always the first sector on the first track on the first head.
- v. It then loads the boot record into a specific place (hexadecimal address 7C00) in RAM.
- vi. The boot record contains a program that BIOS now branches to, giving the boot record control of a computer.
- vii. The boot record loads the **initial system file** (for example, for DOS systems, IO.SYS) into RAM from the diskette or hard disk.
- viii. The initial file (for example, IO.SYS, which includes a program called SYSINIT) then loads the rest of the operating system into RAM .
- ix. The initial file (for example, SYSINIT) loads a system file (for example, MSDOS.SYS) that knows how to work with the BIOS.
- x. One of the first operating system files that is loaded is a system configuration file (for DOS, it's called CONFIG.SYS).
- xi. Another special file that is loaded is one that tells which specific applications or commands the user wants to have included or performed as part of the boot process. In DOS, this file is named AUTOEXEC.BAT. In Windows, it's called WIN.INI.
- xii. After all operating system files have been loaded, the operating system is given control of a computer and performs requested initial commands and then waits for the first interactive user input .

Q 6. Which file contains executable instructions of the function printf in TurboC/Visual Studio Compiler? How can you find the names of these files for any compiler which you are using?

cos.obj , kernel32.lib are the files in case of Turbo C and Microsoft Visual Studio compiler. Remove/delete all the library files directories from the setting and compile the program. You will get linking error mentioning the name of the library file where executable code of the function printf is present. In case of TC, you will get linking error saying that “ Unable to open a file cos.obj”.

Questions

Q.1 State true or false.

- i. Computer cannot think.
- ii. Computer can make a mistake.
- iii. Data stored on a RAM can be changed.
- iv. Information stored on ROM cannot be erased.
- v. Data on secondary memory is volatile in nature.
- vi. ALU performs arithmetic and logical calculations on data.
- vii. Program is a set of logical instructions used to perform a certain task.
- viii. Computer can only understand binary language known as a machine language.
- ix. Machine and assembly languages are low level languages.
- x. Computer can directly understand the assembly language code.
- xi. Assembler is a program that converts assembly language code into machine language instructions.
- xii. An operating system (OS) is a set of computer programs that manage the hardware and software resources of a computer.
- xiii. C was developed at AT&T Bell Laboratories in 1972.
- xiv. Algorithm is a set of logical steps to perform a certain task.
- xv. It is possible to write a C program without a main function.
- xvi. printf is a standard command that displays the value stored at that memory location.
- xvii. Function is a group of set of instructions arranged in a sequence to perform certain tasks.
- xviii. Comment is information written inside the source code that is ignored by a preprocessor when the source file is preprocessed.
- xix. Directory is the location on a computer where files and other directories can be stored.
- xx. C language is not a case sensitive language.
- xxi. Every C program source file ends with an extension .c
- xxii. Every instruction in c should be terminated by a semicolon (;).
- xxiii. Preprocessor copies the contents of a header file into a source file.
- xxiv. In C language, it is mandatory to declare variables at the beginning of a block, otherwise, compile time error is generated.

Q.2 Guess the output of the following programs.

I. `#include <stdio.h>`
`int main()`
`{`
`int one=1947;`
`int two=191;`
`int subtraction = one- two;`
`printf(“%d”, subtraction);`
`return 0;`
`}`

II. `#include <stdio.h>`
`int add()`
`{`

30 C for Beginners

```
    int one=1980;
    int two=191;
    int subtraction = one- two;
    printf("%d", subtraction);
}
```

III. `#include <stdio.h>`
`int main()`
{
 `int one=1947;`
 `int two=191;`
 `int subtraction = one- two,`
 `printf("%d", subtraction);`
}

IV. `#include <stdio.h>;`
`int main();`
{
 `int one=1947;`
 `int two=191;`
 `int subtraction = one- two,`
 `printf("%d", subtraction);`
}

V `#include <stdio.h>`
`int main()`
{
 `int one=1947;`
 `int two=191;`
 `int subtraction = one- two,`
 `xprintf("%d", subtraction);`
}

VI `#include <stdio.h>`
`int main()`
{
 `int Number1;`
 `printf("%d", Number1);`
}

VII `#include <stdio.h>`
`int main()`
{
 `int Number1;`
 `int Number2;`
 `Number1=13579;`
 `Number2=2468;`
 `int sum= Number1+ Number2;`

```
printf("%d", sum);
return 0;
}
```

- Q.3** What is a computer? Mention few applications where computer is used.
- Q.4** Explain the block diagram of a computer?
- Q.5** Differentiate the followings.
- I. Primary and Secondary Memory
 - II. RAM and ROM
 - III. Machine and Assembly Language
 - IV. Assembly and C language
 - V. Machine and C language
 - VI. Compiler and Linker
- Q.6** Explain the following terms.
- I. Operating system
 - II. Program
 - III. Algorithm
 - IV. Compiler
 - V. Linker
 - VI. Preprocessor
 - VII. Editor
 - VIII. Loader
 - IX. Executable file
 - X. Object code file.
- Q.7** What is a machine language? What are its advantages and disadvantages?
- Q.8** What is an assembly language? What are its advantages and disadvantages?
- Q.9** What is a data type? Give example.
- Q.10** Create 3 integer variables that stores 10, 20 and 30 respectively.
- Q.11** Explain C language environment in detail.
- Q.12** Write a program that does multiplication of 128 and 3876?
- Q.13** What is the basic language of a computer? Why?
- Q.14** Who invented C? Discuss the history of C.
- Q.15** Who is a father of a computer?
- Q.16** Why one should use C language for programming?
- Q.17** What is primary memory? What are RAM and ROM?
- Q.18** What is secondary memory?
- Q.19** What is ALU?
- Q.20** What is program? What language computer can understand directly?
- Q.21** What is an assembler? What it does?
- Q.22** What is an operating system?
- Q.23** What is a function?
- Q.24** What is a comment? What are the different types of comments?
- Q.25** What is editor, preprocessor, compiler, and linker?
- Q.26** What are the different types of programming errors? Explain in detail.

General Information²⁰

A. A Brief History of Computer Technology

A complete history of computing would include a multitude of diverse devices such as the **ancient Chinese abacus**. Abacus was the first device known to carry out the calculations. It works by sliding beads back and forth on a frame with the beads on the top of the frame representing fives and on the bottom ones.

Charles Babbage's analytical engine (1834). This machine can perform all the four operations that are addition, subtraction, multiplication and division.

In 1960s, analog computers were routinely used to solve systems of finite difference equations arising in oil reservoir modeling. In the end, digital computing devices proved to have the power, economics and scalability necessary to deal with large scale computations. Digital computers now dominate the computing world in all areas ranging from the hand calculator to the supercomputer and are pervasive throughout society.

The advent of **the first electromechanical computers** was an exciting time, because these scamps could actually provide significant amounts of computational power. Development of electromechanical computers continued into the 1960's. The Minivac 601 was a model introduced in 1961 with its name derived from the Sperry Rand Univac computer.

The **evolution of digital computing** is often divided into generations. Each generation is characterized by dramatic improvements over the previous generation in the technology used to build computers, the internal organization of computer systems, and programming languages. Although not usually associated with computer generations, there has been a steady improvement in algorithms, including algorithms used in computational science.

The above information is taken from <http://library.albany.edu/usered/wwwdex/au/au3.html>

B. History of computers

ABACUS(2400 B.C.E.)	Anonymous Chinese	It is the earliest known tool for use in computation.
Calculating Clock (1623)	Wilhelm Schickard	First Automatic Calculator. Performing Addition and Subtraction
Pascal's Calculator (1643)	Blaise Pascal	Used For Taxes In France Until 1799. Performing Addition ,Subtraction
Analytical Engine Difference Engine.	Charles Babbage	First to conceptualize and design a fully programmable computer early as 1820

²⁰ Reader may skip general information section.

Hollrith and tabulating Machine.(1890)	Harman Hollrith	Device could automatically read census information which had been punched onto card.
First Electronic Digital Computing Device(1937)	Atanasoff-Berry	The machine, conceived in 1937, was capable of solving up to 29 simultaneous linear equations and was successfully tested.

C. History of Algorithms

The word algorithm comes from the name of the 9th century Persian mathematician Abu Abdullah Muhammad bin Musa al-Khwarizmi. The word algorithm was only originally referred to the rules of performing arithmetic calculations using Hindu-Arabic numerals. The first case of an algorithm written for a computer was Ada Byron's notes on the analytical engine written in 1842. The lack of mathematical rigor in the "well-defined procedure" definition of algorithms posed some difficulties for mathematicians and logicians. This problem was largely solved with the description of the Turing machine, an abstract model of a computer formulated by Alan Turing.

Appendix

The Turbo C compiler is freely available on the websites

www.pitt.edu/~stephen/misc/turboC.exe
<http://www.pitt.edu/~stephen/misc/downloadTC.html>

Turbo C is an old compiler, but it should work fine for the purposes of our class. These are instructions for downloading and installing v2.01 of the Borland Turbo C compiler on Windows 95/98/NT. It should also work on Windows NT and Windows 2000.

These instructions seem involved but they are fairly simple, if you follow them exactly.

1. Click here to download the [zipped Free Turbo C Compiler](#)
2. Go to where you downloaded the file, and double click on the self-extracting file (turboC.exe) in Windows to extract it. This will bring up a WinZip Self-Extractor window (you do **NOT** need WinZip installed on your machine).
By default, this will extract the files to **C:\tctemp** directory. You may designate a different location.
Hit return to extract the files.
3. Exit the WinZip Self-Extractor window (by hitting return twice).
4. Once the files have been extracted, go to the directory **c:\tctemp** (or wherever you put the unzipped files), and double click on the file called install (it may be called install.exe).

34 C for Beginners

5. This will step you through the installation.
 - Hit enter to start the installation
 - Select the **drive** where the unzipped files are. The default is "A", so you should enter "**C**".
Then hit return.
 - Hit return, again. This installs from the directory, **\tctemp**.
 - Hit return, again. This says to **Install Turbo C on a Hard Drive**.
 - Use the Up/down arrow keys (hit the up arrow once) to select **Start Installation**, and then hit return, again.
 - At this point, the v2.01 Turbo C compiler is installed in C:\TC. That is where the tcc.exe executable is.
6. That's everything, but you must add C:\TC to your executable search path. The easiest way to do this:
 - Bring up an MS-DOS window.
 - Position the cursor on the title bar of the MS-DOS window and right-click.
 - This pops up a menu. Click on **Properties**.
 - At the top of the window, click on the **Program** folder.
 - In the middle field, labeled **Batch file:** enter the following:
C:\tctemp\init.bat
(This assumes that you unzipped the files into C:\tctemp.)
 - Click **OK** at the bottom of the window.
 - Close the MS-DOS window.
7. Now you're done! Try it out by creating a small C program and compile it.
 - Bring up the MS-DOS window in the same way as you did above. (NOTE: You must close the first MS-DOS window.)
 - Type **tcc** and hit return. You should see a usage message describing all the many compiler options.
 - Now create, compile, and run the addition of two numbers program.