

Table of Contents

Sessions

1.	Basics of C - Concepts	1
2.	Variables and Data Types - Concepts	29
3.	Variables and Data Types - Lab	47
4.	Operators and Expressions - Concepts	53
5.	Operators and Expressions - Lab	71
6.	Input and Output in 'C' - Concepts	83
7.	Condition - Concepts	107
8.	Condition - Lab	125
9.	Loop - Concepts	137
10.	Loop - Lab	159
11.	Arrays - Concepts	169
12.	Arrays - Lab	187
13.	Pointers - Concepts	201
14.	Pointers - Lab	225
15.	Functions - Concepts	235
16.	Functions - Lab	261
17.	Strings - Concepts	267
18.	Strings - Lab	283
19.	Advanced Data types and Sorting - Concepts	293
20.	Advanced Data types and Sorting - Lab	311
21.	File Handling - Concepts	321

Objectives

At the end of this session, you will be able to:

- Differentiate(*phân biệt*) between Command, Program and Software
- Explain(*giải thích*) the beginning of C
- Explain when and why is C used
- Discuss(*thảo luận*) the C program structure(*cấu trúc chương trình*)
- Discuss algorithms(*thuật toán*)
- Draw flowcharts(*sơ đồ luồng công việc*)
- List the symbols(*kí hiệu*) used in flowcharts

Introduction

1.1 Instructions to a Computer

The operating system software exists inside the computer. The operating system is referred to as system software. This software starts up the computer and performs some initial settings before giving us an operational screen. To achieve this, the operating system is made up of a set of programs. Every program tries to give solution to one or more problems. Each program is set of instruction to solve the problem it tries to address. Thus a group of instructions make up a program and a group of programs make up software.

Let's consider an analogy to get more clarity: One of our friends comes home and we serve the Strawberry Milk Shake. He finds it so tasty that he too wants the recipe. So, we give him the recipe as:

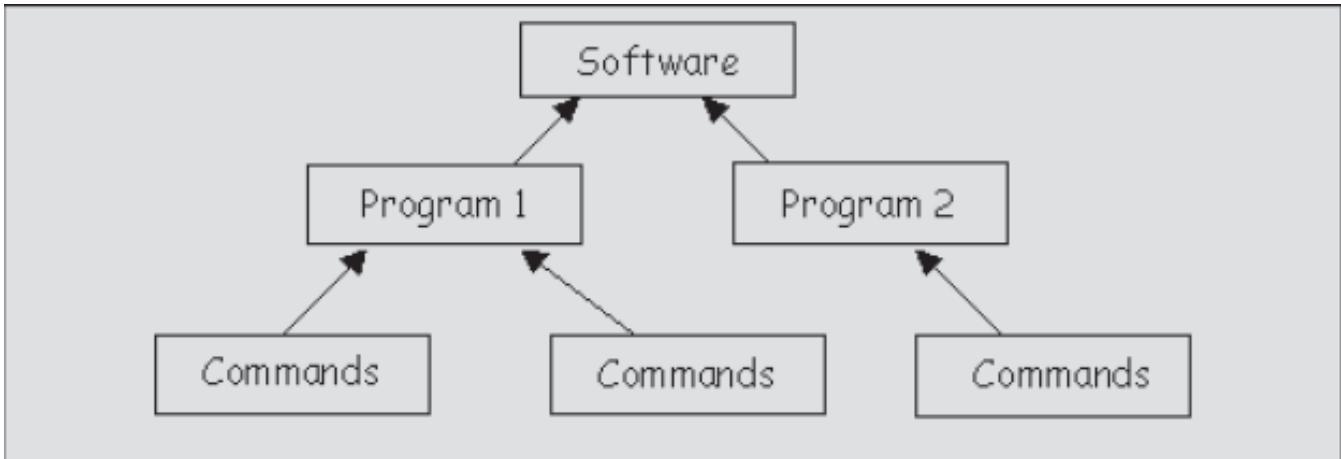


Figure 1.1: Software, Program and Commands

1.2 The C Language

While, BCPL and B do not support data-types (they are typeless), C provides a variety of data types. The main data- types are characters, integers and floating-point numbers.

C is closely associated with the UNIX system yet C is not tied to any one operating system or machine (C gắn chặt với hệ thống UNIX nhưng C không bị ràng buộc với bất kỳ hệ điều hành hoặc máy nào.). C has been effectively used to write programs in different domains (C đã được sử dụng hiệu quả để viết chương trình trong các lĩnh vực khác nhau.) .

C was used for systems programming(C c s d ng l p trình h th ng). A system program is associated with the operating system of the computer or its support utilities(M t ch ng trình h th ng c liên k t v i h i u hành c a máy tính ho c các t i n ích h tr c a nó). . Operating Systems(h i u hành), Interpreters(trình thông d ch), Editors(trình biên t p), Assembly programs(ch ng trình l p ráp) are usually called systems programs(th ng c g i là ch ng trình h th ng). UNIX Operating System was developed using C(H i u hành UNIX c phát tri n b ng C). . C is now being used by many programmers for kinds of tasks because of its portability and efficiency. C compilers are available for almost all computers. Codes written in C on a one machine can be compiled and run on another machine by making a few or no changes. C compiler produces fast and error-free object code.

C also offers the speed of an assembly language. Programmers can create and maintain library of functions, which can reused by other programs. Thus large projects can be managed easily, with minimum efforts.

1.2.1 C - A Middle Level Language

C is thought of as a middle-level language because it combines elements of high-level languages and functionalities of an assembly (low-level) language. C allows manipulation of the basic elements of a

Session 1

computer i.e. bits, bytes, addresses etc. Also, C code is very portable (mã C rất dễ di chuyển), that is, software written on one type of computer can work on another type of computer (nghỉ là phím máy tính có thể hoạt động trên máy tính khác). Although C has five basic built-in data types, it is not strongly typed language as compared to high-level languages. C allows data type conversions (C cho phép chuyển đổi dữ liệu). It allows direct manipulation of bits, bytes, words, and pointers (Nó cho phép thao tác trực tiếp các bit, byte, từ và con trỏ). Thus, it is used for system-level programming (nó sử dụng cho lập trình cấp hệ thống).

1.2.2 C - A Structured Language

C allows synthesis of code and data (C cho phép hợp mã và dữ liệu). This is a distinguishing feature of any structured language (đây là một tính năng phân biệt của ngôn ngữ có cấu trúc nào). It refers to the ability of a language to collect and hide all information and instructions, necessary to perform a specific task, from the rest of the program (Nó có phần khung ngắt am hiểu ngôn ngữ trong việc thu thập và xử lý thông tin và hành động, cẩn thận chỉ ra những mâu thuẫn, không phải còn lặp lại cách làm). This can be done using functions or code blocks (điều này có thể thực hiện bằng cách sử dụng các hàm hoặc khối lệnh). Functions (hàm) are used to define and separate, tasks required in a program (cách xác định và tách biệt các nhiệm vụ cần thiết trong chương trình). Code block (Khối mã) is a logically connected group of program statements that is treated like a unit (là một nhóm các câu lệnh chương trình có kết nối lôgic để xử lý một mảng). A code block is created by placing a sequence of statements between opening and closing curly braces as shown below (Mở khai mạc để tạo ra một cách tách rời chuỗi các câu lệnh gì đó và đóng khép ở bên dưới).

```
do
{
    i = i + 1;
    .
    .
}
while (i < 40);
```

Structured language support several loop constructs, such as while, do-while, and for. These loop constructs help the programmers to control the flow of the program.

1.3 The C Program Structure

C has few keywords, 32 to be precise (C có ít nhất 32). These keywords, combined with the formal C syntax, form the C language. But many C compilers have added more keywords to use the memory organization of certain preprocessors.

Some rules (qui tắc) for programs written in C are as follows:

- All keywords are lower cased (Tất cả các từ khóa đều là chữ thường)
- C is case sensitive, do while is different from **DO WHILE** (**C phân biệt chữ hoa chữ thường, do while khác với DO WHILE**)
- Keywords cannot be used for any other purpose, that is, they cannot be used as a variable or

Session 1

Basics of C

function name (Tên khóa không th c s d ng cho b t k m c ích nào khác, nghĩa là chúng không th c s d ng nh m t bi n ho c tên hàm)

- main() is always the first function called when a program execution begins (main luôn là hàm đầu tiên cung cấp khi chương trình bắt đầu thực thi)

Consider the following program code:

```
main ()
{
    /* This is a sample program */
    int i = 0;
    i = i + 1;
    .
    .
}
```

Note: Various aspects of a C program are discussed with respect to the above code. This code will be referred to as `sample_code`, wherever applicable.

1.3.1 Function Definition

C programs are divided into units called functions. The `sample_code` has only one function `main()`. The operating system always passes control to `main()` when a C program is executed (điều hành luôn chuyển quyền cho hàm `main()` khi mà nó được gọi).

The function name is always followed by parentheses (Tên hàm luôn紧跟在函数名后面). The parentheses may or may not contain parameters (Điều này có thể không có tham số).

1.3.2 Delimiters

The function definition is followed by an open curly brace (`{`) (nghĩa là紧跟在函数定义后面的是一个左花括号) . This curly brace signals the beginning of the function. Similarly a closing curly brace (`}`) after the statements, in the function, indicate the end of the function. The opening brace (`{`) indicates that a code of block is about to begin and the closing brace (`}`) terminates the block of code. In `sample_code`, there are two statements between the braces. In addition to functions, the braces are also used to delimit blocks of code in other situations like loops and decision-making statements.

1.3.3 Statement Terminator

Consider the line `int i = 0` in `sample_code` is a statement. A statement in C is terminated with a semicolon (`;`) (Một câu lệnh trong C kết thúc bằng dấu phẩy (;)). A carriage return, whitespace, or a tab is not understood by the C compiler (ký tự trống, khoảng trắng hoặc tab : trình biên dịch C không hiểu).

Session 1

Basics of C

There can be more than one statement on the same line as long as each one of them is terminated with a semi-colon. A statement that does not end in a semicolon is treated as an invalid line of code in C.

Concepts

1.3.4 Comment Lines

Comments are usually written to describe the task of a particular command, function or an entire program. The compiler ignores them. In C, comments begin with /* and are terminated with */(Trong C, chú thích bút lú bằng /* và kết thúc bằng */), in case the comments contain multiple lines(trong trường hợp chú thích chia thành nhiều dòng.). Care should be taken that the terminating delimiter (*/) is not forgotten(Cần lưu ý không quên dấu phân cách kết thúc (*/)). Otherwise, the entire program will be treated like a comment(Nếu không, toàn bộ chương trình sẽ coi như bình thường). In case the comment contains just a single line you can use // to indicate that it is a comment(Trong trường hợp bình thường chỉ có một dòng duy nhất, bạn có thể sử dụng // để chỉ ra rằng nó là bình thường.).

For example

```
int a=0; //Variable 'a' has been declared as an integer data type
```

1.3.5 The C Library

All C compilers come with a standard library of functions that perform the common tasks.(Tất cả các trình biên dịch C đều kèm với một thư viện tiêu chuẩn bao gồm các hàm thường dùng.)In some installations of C, the library exists in one large file while in others it is contained in numerous small files.(Trong một số cài đặt C, thư viện thường được tách thành nhiều file nhỏ, trong khi các cài

t khác nó sẽ chia thành nhiều tập riêng.).While writing a program, the functions contained in the library can be used for various tasks.(Khi viết một chương trình, các hàm có trong thư viện có thể được sử dụng cho nhiều mục đích khác nhau.)A function written by a programmer can be placed in the library and be used in as many programs as and when required.(Một hàm do lập trình viên viết có thể được

tùy vào thư viện và sử dụng trong nhiều chương trình khác khi cần)Some compilers allow functions to be added in the standard library, while some compilers require a separate library to be created.(Một số trình biên dịch cho phép thêm các hàm vào thư viện tiêu chuẩn, trong khi một số khác yêu cầu tự động thêm vào riêng biệt)

1.4 Compiling and Running a Program

The various stages of translation of a C program from source code to executable code are as follows:Các giai đoạn

khác nhau trong quá trình dịch mã thành chương trình C từ mã nguồn sang mã thi hành sau:

➤ Editor/Word Processor

The source code is written using an editor or a word processor.(Mã nguồn được viết bằng cách sử dụng trình soạn thảo hoặc trình xử lý văn bản.)The code should be written in the form of standard text files, as C accepts source code only in this form (Mã phím được viết dưới dạng tệp văn bản tiêu chuẩn, vì C chỉ chấp nhận mã nguồn dưới dạng này).Some compilers supply programming environments (see appendix) that include an editor (Một số trình biên dịch cung cấp môi trường lập trình (xem phần phụ lục) bao gồm cả trình soạn thảo).

➤ Source Code

This is the text of the program, which the user can read.(Đây là văn bản của chương trình, mà người dùng có thể đọc được.)It is the input for the C compiler.(Đây là input cho trình biên dịch C)

Session 1

➤ C Preprocessor

The source code is first passed through the C preprocessor.(Mã ngu n u tiên c chuy n qua b ti n x lý C.)Preprocessors act on statements beginning with #.(Các b ti n x lý ho t ng trên các câu l nh b t u b ng #).These statements are called directives (explained later).(Nh ng câu l nh này c g i là ch th)The directives are usually placed at the start of the program, though they can be placed anywhere else.(Các ch th th ng c t u ch ng trình, m c dù chúng có th c t b t k âu).The directives are short names given to a set of code.(Các ch th là nh ng tên ng n c gán cho m t t p h p mă l nh)

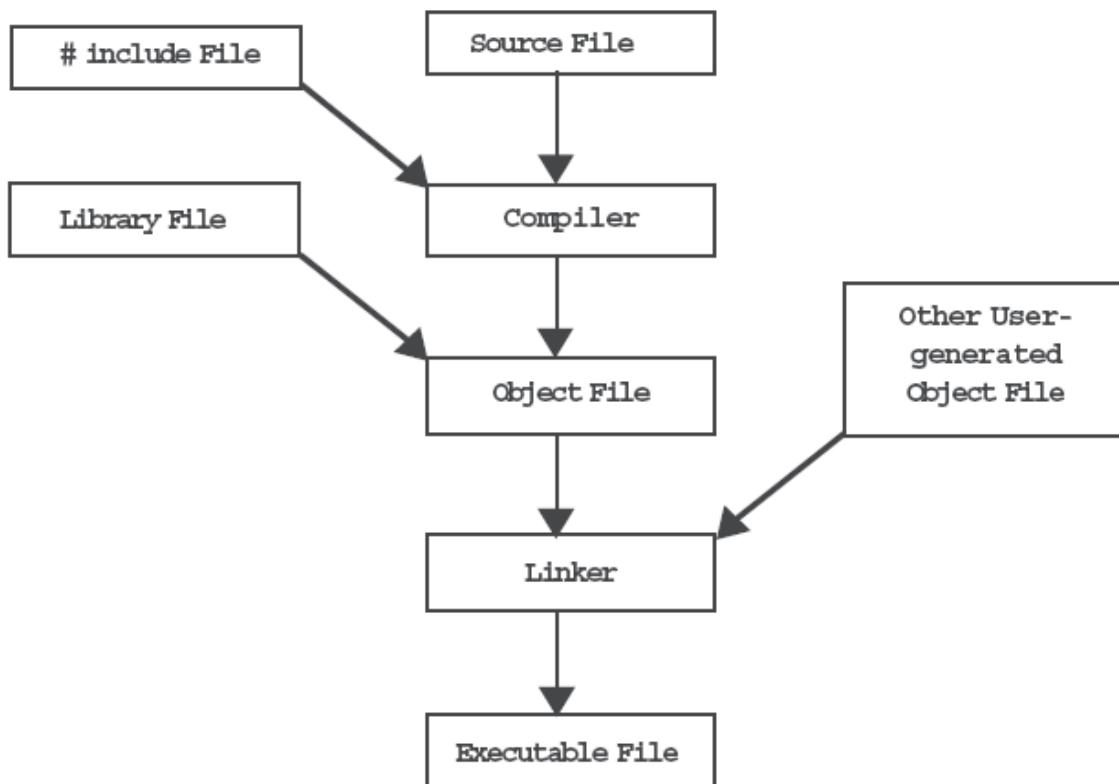


Figure 1.2: Compiling and Running a Program

1.5 The Programming Approach to Solving Problems

The steps would look as shown below:

STEP 1: Leave the room

STEP 2: Head towards the staircase

STEP 3: Go down to the basement

Session 1

Basics of C

STEP 4: Head for the cafeteria

An algorithm can be defined as a procedure, formula, or recipe for solving a problem.(Một thuật toán có thể coi là một quy trình, công thức hoặc pháp gián quyết một vấn đề.)It consists of a set of steps that help to arrive at a solution.(Nó bao gồm một tập các bước giúp tìm ra giải pháp.)

From our discussion, it is obvious that in order to solve a problem, we need to first understand the problem.(Tìm hiểu rõ ràng về giải quyết vấn đề, trước tiên chúng ta cần hiểu rõ vấn đề.)Next, we need to gather all relevant information required.(Tiếp theo, chúng ta cần thu thập thông tin liên quan cần thiết.)Once this is done, the next step would be to process these bits of information.(Sau khi hoàn thành, bước tiếp theo là xử lý những thông tin này.)Finally, we would arrive at the solution to the problem.(Cuối cùng, chúng ta sẽ tìm ra giải pháp cho vấn đề.)

The algorithm we have are a set of steps listed in simple language.(Thuật toán mà chúng ta có là một tập các bước viết bằng ngôn ngữ tự nhiên).It is very likely that though the steps written by two people may be similar, the language used to express these steps may be different.(Rất có thể hai người viết có thể dùng ngôn ngữ khác nhau).It is, therefore, necessary to have some standard method of writing algorithms so that everyone easily understands it.(Để vậy, cần phải có một phương pháp chuẩn hóa để mọi người dễ dàng hiểu).Hence, algorithms are written using two standard methods—pseudo codes and flowcharts (Vì vậy, thuật toán sẽ viết bằng ngôn ngữ chuẩn hóa: mã giả và lôgic).

1.5.1 Pseudo code

Note that ‘pseudo code’ is not actual code (pseudo=false) (Lưu ý rằng ‘mã giả’ không phải là mã thực (pseudo = sai)).Pseudo code uses a certain standard set of words, which makes it resemble a code (Mã giả sử dụng một набор từ khóa nhất định, làm cho nó giống như một ngôn ngữ).However, unlike code, pseudo code cannot be compiled or run (Tuy nhiên, không thể biên dịch và chạy mã giả).

Let us, for example, consider the pseudo code written in Example 1 for displaying a ‘Hello World!’ message .(Ví dụ, hãy xem xét mã giả trong Ví dụ 1 hiển thị thông báo ‘Hello World!’)

Example 1:

```
BEGIN
    DISPLAY 'Hello World!'
END
```

As can be seen in the simple pseudo code above, each pseudo code must start with the word BEGIN or START, and end with END or STOP (Nhưng có thể thấy trong mã giả có một số từ như BEGIN hoặc START và kết thúc bằng END hoặc STOP).To display some value, the word DISPLAY or WRITE is used (để hiển thị giá trị nào đó, ta sử dụng DISPLAY hoặc WRITE).

Session 1

Basics of C

Since the value to be displayed is a constant value in this case, the value (Hello World) is enclosed within quotes (Vì giá tr c hi n th là m t giá tr h ng trong tr ng h p này, giá tr (Hello World) c t trong d u ngo c kép). Similarly, to accept a value from the user, the word INPUT or READ is used (T ng t , nh n m t giá tr t ng i dùng, t INPUT ho c READ c s d ng).

To understand this better, let us have a look at the pseudo code (Refer to Example 2) for accepting two numbers from the user, and for displaying the sum of the two numbers (hi u i u này rõ h n, h y cung xem xét mã gi (Tham kh o Ví d 2) nh n hai s t ng i dùng và hi n th t ng c a hai s ó).

Example 2:

```
BEGIN
    INPUT A, B
    DISPLAY A + B
END
```

In this pseudo code, the user inputs two values, which are stored in memory and can be accessed as A and B respectively (Trong mã gi này, ng i dùng nh p hai giá tr, c l u tr trong b nh và có th truy c p t ng ng là A và). Such named locations in memory are called variables (Các v trí c t tên nh v y trong b nh c g i là bi n). The next step in the pseudo code displays the sum of the values present in variables A and B(B c ti p theo trong mã gi hi n th t ng các giá tr có trong các bi n A và B).

However, the same pseudo code can be modified to store the sum of the variables in a third variable and then display the value in this variable as shown in Example 3.

Tuy nhiên, cùng m t mã gi có th c s a i l u tr t ng các bi n trong bi n th ba và sau ó hi n th giá tr trong bi n này nh trong Ví d

Example 3:

```
BEGIN
    INPUT A, B
    C = A + B
    DISPLAY C
END
```

A set of instructions or steps in a pseudo code is collectively called a construct (M t t p h p các h ng d n ho c các b c trong mã gi c g i chung là m t c u trúc). There are three types of programming constructs - sequence, selection, and iteration constructs (Có ba lo i c u trúc l p trình - c u trúc tu n t , c u trúc l a ch n và c u trúc l p). In the pseudo codes, we have written above, we have used sequence constructs (Trong các mã gi mà chúng tôi ã vi t trên, chúng tôi ã s d ng các c u trúc tu n t). These are called so as they are instructions that are performed in a sequence, one after the other, starting from the top (Chúng c g i nh v y vì chúng là các h ng d n c th c hi n theo trình t , l n l t, b t u t trên cùng.).

1.5.2 Flowcharts

To understand this better, let us have a look at a flowchart, given in figure 1.3, for displaying the traditional 'Hello World!' message.

hi u rõ h n v i u này, chúng ta hãy xem s kh i c a ra trong hình 1.3
hi n th ph ng pháp truy n th ng
Tin nh n 'Xin chào th gi i!'.
Concepts

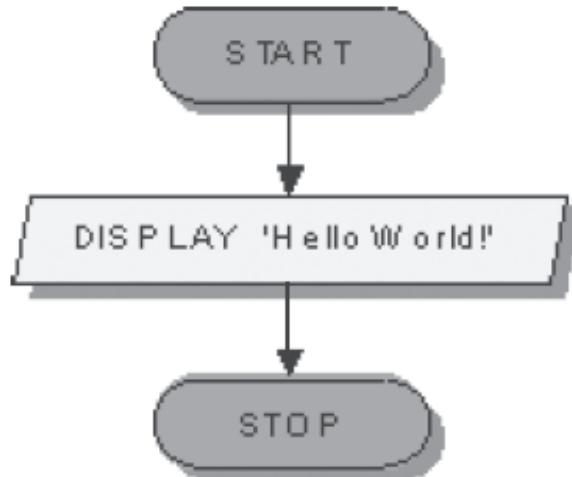


Figure 1.3: Flowchart to add two numbers

Flowcharts, like pseudo codes, begin with the START or BEGIN keyword, and end with the END or STOP keyword (L u , gi ng nh m g i , b t u v it kh o a START ho c BEGIN v k t th c v it kh o a END ho c STOP). In a similar way, the DISPLAY keyword is used to display some value to the user (Theo c ch t ng t , t kh o a DISPLAY c s d ng hi n th m t gi a tr n o cho ng i d ng). However, here, every keyword is enclosed within symbols (Tuy nh i en, ay, m i t kh o a c t trong c c k y h i u). The different symbols used in flowcharting and their relevance are tabulated in Figure 1.4 (C c k y h i u kh c nhau c s d ng trong vi c v l u v y ngh a c a chung c trinh b y trong H m 1.4).

Symbol	Description
	Start or End of the Program
	Computational Steps
	Input / Output instructions
	Decision making & Branching
	Connectors
	Flow Line

Fig 1.4: Flowchart Symbols

Session 1

Basics of C

Concepts

Example: We accepted two numbers from the user, added them up (stored the result in a third variable), and displayed the result (Ví d : Chúng ta cần nhập hai số để dùng, chúng ta sẽ (lưu trữ) vào một biến thứ ba) và hiển thị kết quả). The flowchart for this would look as shown in Figure 1.5 (Lưu ý rằng cách này không đúng kỹ thuật trong Hình 1.5).

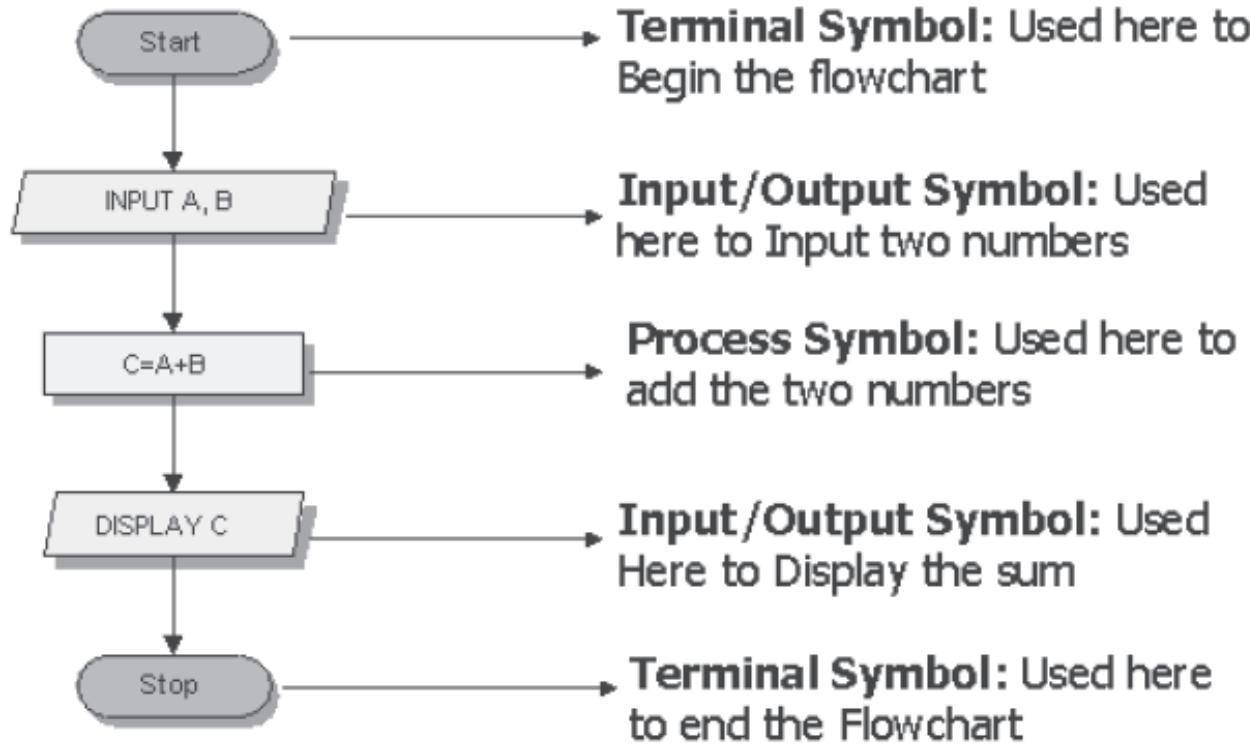


Fig 1.5: Flowchart to add two numbers

The step in which the values of the two variables are added and assigned to a third variable is considered to be a process and is shown by a rectangle (Bên trong ô giá trị có hai biến cần gán và gán cho một biến khác coi là một quá trình và có biểu bối hình chữ nhật).

Most often, flowcharts span several pages (Thường thì, lưu trữ dài qua nhiều trang). In such a case, symbols called connectors are used to indicate the location of the joins (Trong trường hợp này, các ký hiệu cũng là các kí tự có sẵn để chỉ ra vị trí của các mảnh). They help us identify the flow across the pages (Chúng giúp chúng ta xác định luồng qua các trang). Circles represent connectors and need to contain a letter or a number, as shown in Figure 1.6 (Các hình tròn để nối các kí tự và cách nhau bằng dấu cách, như hình ảnh trong Hình 1.6). Using these, we can pick up the link between two incomplete flowcharts (Bằng cách sử dụng những cách này, chúng ta có thể liên kết hai lối đi hoàn chỉnh).

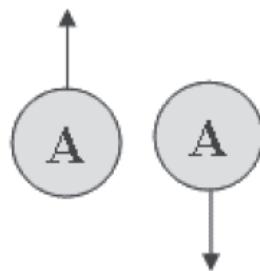


Figure 1.6: Connectors

Session 1

Some other essential things to be taken care of when drawing a flowchart are: (Một số điều cần 注意 khác cần lưu ý khi vẽ luồng công việc là):

- Initially concentrate only on the logic of the problem and draw out the main path of the flowchart (Ban đầu chỉ tập trung vào logic của nó và vẽ ra lối chính của nó.).
- A flowchart must have only one START and one STOP point (Số luồng công việc chỉ có một điểm BẮT ĐẦU và một điểm KẾT THẤT).
- It is not necessary to represent each and every step of a program in the flowchart. Only the essential and meaningful steps need to be represented (Không cần thiêt phì thịnh hành tất cả các bước chia thành luồng. Chỉ có các bước thiết yếu và có ý nghĩa cần phải có trong luồng).

We have worked with sequence constructs in which execution flows through all the instructions from the top in a sequence (Chúng ta đã làm việc với các cấu trúc tuần tự, trong đó quá trình thực thi chỉ qua tất cả các lệnh trên xuôi theo một trình tự). We can come across conditions in our program, based on which the path of execution may branch (Chúng ta có thể gặp các điều kiện trong chương trình của mình, để trên đó mà có một quá trình thực thi có thể phân nhánh). Such constructs are referred to as selection, conditional, or branching construct (Các cấu trúc này cũng là cấu trúc chia nhánh, điều kiện hoặc phân nhánh).

➤ The IF construct

A basic selection construct is an 'IF' construct (Một cấu trúc chia nhánh cơ bản là cấu trúc 'IF'). To understand this construct, let us consider an example where the customer is given a discount if he makes a purchase of above \$100 (hiểu rõ cấu trúc này, hãy xem xét một ví dụ trong đó khách hàng có giá mua hàng trên 100 đô la). Every time a customer is billed, a part of the code has to check if the bill amount exceeds \$100 (Mỗi khi khách hàng thanh toán, một phần của mã cần tra xem số tiền trên hóa đơn có vượt quá 100 đô la hay không). If it does, then deduct 10% of the total amount; otherwise, deduct nothing (Nếu có, thì trừ 10% tổng số tiền; nếu không, thì không trừ gì cả).

The general form of an IF statement or construct is as follows:

```
IF condition  
    Statements → Body of the IF construct  
END IF
```

An 'IF' construct begins with IF followed by the condition (M t c u trúc 'IF' b t u b ng t IF theo sau là i u ki n). If the condition evaluates to true, then control is passed to the statements within its body (N u i u ki n ánh giá là úng, thì quy n i u khi n s c chuy n n các câu lệnh trong thân c a nó). If the condition returns false, the statements within the body are not executed, and the program flow continues with the next statement after the END IF (N u i u ki n tr v sai, các câu lệnh trong thân s khong c th c thi, và lu ng ch ng trình s ti p t c v i câu lệnh t p theo sau END IF). The IF construct must always end with an END IF, as it marks the end of the construct (C u trúc IF ph i luon k t thuc b ng END IF, vi nó ánh d u s k t thuc c a c u trúc).

Let us take a look at Example 4, which uses IF (Hãy cùng xem xét Ví d 4, trong ó s d ng IF).

To find if a number is even, we proceed as follows (tìm xem m t s có ph i là s ch n hay khong, chúng ta t i n hành nh sau):

Example 4:

```
BEGIN
    INPUT num
    r = num MOD 2
    IF r=0
        Display "Number is even"
    END IF
END
```

The above code accepts a number from the user, performs the MOD operation on it, and checks if the remainder is zero (Má trên nh n m t s t ng i dùng, th c hi n phép MOD trên nó và ki m tra xem ph n d có b ng khong hay khong). If it is, then it displays a message; otherwise, it just exits. (N u có, thì nó hi n th m t thông báo; n u khong, nó ch thoát)

A flowchart for the above pseudo code would look as shown in Figure 1.7 (L u cho m g i trên s trông gi ng nh c hi n th trong Hình 1.7).

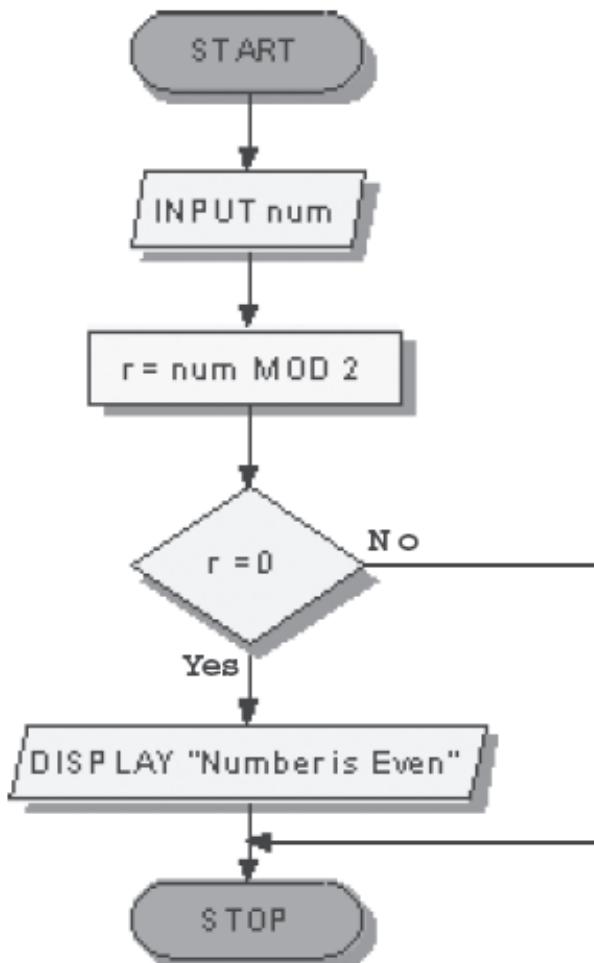


Figure 1.7 : Even Number or Not

The syntax for the IF statement in C is as follows:

```
if (condition)
{
    statements
}
```

➤ The IF...ELSE Construct

In Example 4, it will be nicer if we responded with a message saying that the number is not even (therefore odd) instead of just exiting (Trong Ví d 4, s t t h n n u chung ta ph n h i b ng m t thong i p noi r ng s ó khong ph i là s ch n (do ó là s l) thay vì ch thoát). One way of achieving this can be by adding another IF statement which checks if the number is not divisible (M t cách t c i u nà y là thêm m t câu l nh IF khác ki m tra xem s ó có ph i là s khong chia h t hay khong). Refer to Example 5:

Session 1

Basics of C

Concepts

Example 5:

```
BEGIN
    INPUT num
    r = num MOD 2
    IF r=0
        DISPLAY "Even number"
    END IF
    IF r<>0
        DISPLAY "Odd number"
    END IF
END
```

Programming languages provide us with what is known as an IF...ELSE construct (Các ngôn ngữ lập trình cung cấp cho chúng ta một cấu trúc điều kiện IF...ELSE). Using this would be a better and more efficient approach to solving the above problem (Việc sử dụng cấu trúc này sẽ là một cách tiếp cận thuận tiện và hiệu quả hơn so với cách trên). The IF...ELSE construct enables the programmer to make a single comparison and then execute the steps depending on whether the result of the comparison is True or False (Cấu trúc IF...ELSE cho phép lập trình viên thực hiện một so sánh duy nhất và sau đó thực hiện các bước tùy thuộc vào việc kết quả của phép so sánh là đúng hay sai).

The general form of the IF...ELSE statement is as follows:

```
IF condition
    Statement set1
ELSE
    Statement set2
END IF
```

The syntax for the if...else construct in C is given below.

```
if(condition)
{
    statement set1
}
else
{
    statement set2
}
```

If the condition evaluates to True, statement set1 is executed; otherwise, statement set2 is executed, but never both (Nếu điều kiện có giá trị là đúng, thì chỉ có câu lệnh set1被执行; không, tinh không, chỉ có câu lệnh set2被执行). So, a more efficient code for our even number example (Example 5) would be as shown in Example 6 (Ví dụ 5) sẽ chỉ có một câu lệnh trong Ví dụ 6).

Session 1

Basics of C

Example 6:

```
BEGIN
    INPUT num
    r=num MOD 2
    IF r=0
        DISPLAY "Even Number"
    ELSE
        DISPLAY "Odd Number"
    END IF
END
```

The flowchart for the above pseudo code is displayed in Figure 1.8.

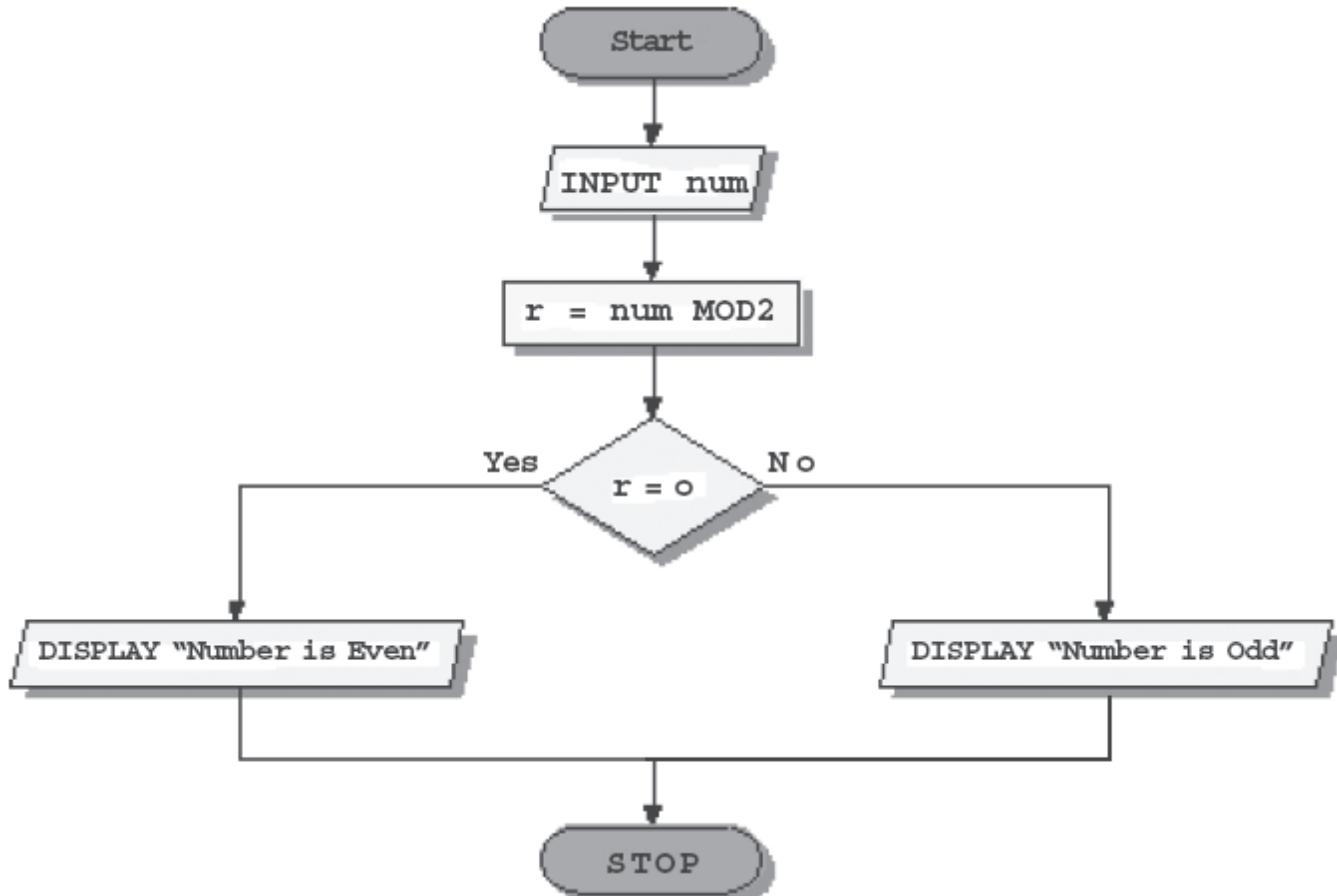


Figure 1.8: Even or Odd number

Session 1

Basics of C

Concepts

➤ Multiple criteria using AND/OR

The IF...ELSE construct helps to reduce complexity and enhances efficiency (C_ú trúc IF...ELSE giúp gi_m ph_ct p v_at ng_c ng hi_u su_t). To some extent, it also improves the readability of the code 9 m_tm_c n_{ào} ó, n_óc_{ng}c_ith_in_kh_nng_cc_am_ã). The IF examples we have discussed till now were fairly simple (Cá_vd_v IF mà ch_úng ta_ãth_olu_ncho_nbâ_ygi_{kh}á_ngi_n). They had one IF condition to evaluate 9 Ch_úng ch_có m_ti_uki_nIF ánh giá_{).} Sometimes we have to check for more than one condition (ôi khi ch_úng ta_{ph}i_{ki}m_{tra}n_{hi}u_hn_mt_iu_{ki}n₎ for example.; ví d_v. This is where the AND operator can be used with the 'IF' statement (ây là l_úc toán t_{AND} có th_cs_dng trong câu l_{nh}'IF). Refer to Example 7

Example 7:

```
BEGIN
    INPUT yearsWithUs
    INPUT bizDone
    IF yearsWithUs >= 10 AND bizDone >= 5000000
        DISPLAY "Classified as an MVS"
    ELSE
        DISPLAY "A little more effort required!"
    END IF
END
```

Even there, we can conveniently use the AND operator to connect them just like how we used it (T_i ây, ch_úng ta_{th}ay_{th} toán t_{AND}b_{ng}toán t_{OR}). Remember we had learned in our previous section that the OR operator evaluates to True even if one of the conditions is True (H_{ay} nh_rng ch_úng ta_ãh_ctrong ph_ntr_cr_{ng}toán t_{OR} ánh giá_{là} úng ngay c_khi m_ttrong cá_ci_uki_nl_à úng).

➤ Nested IFs

Another way to do Example 7 involves using Nested IFs (M_tcách_{kh}ác_{th}ch_in Ví d₇l_às_dng cá_câu_lnh IF l_{ng}nhau_u). A nested IF is an IF inside another IF statement (M_tcâu_lnh IF l_{ng}nhau_l m_tcâu_lnh IF n_mbên_{trong}m_tcâu_lnh IF kh_{ác}). Let us re-write the pseudo code in Example 7 using nested IFs (H_{ay}cÙng vi_tl_im_ãgi_{trong}Ví d₇b_{ng}cách_sd_{ng}cá_câu_lnh IF l_{ng}nhau_u). The code would look like in Example 8 (M_ãs_{tr}ông gi_{ng}nh_{trong}Ví d₈).

Example 8:

```
BEGIN
    INPUT yearsWithUs
    INPUT bizDone
```

Session 1

Basics of C

```
IF yearsWithUs >= 10
    IF bizDone >= 5000000
        DISPLAY "Classified as an MVS"
    ELSE
        DISPLAY "A little more effort required!"
    END IF
ELSE
    DISPLAY "A little more effort required!"
END IF
END
```

The above code performs the same function without an 'AND'.

The flowchart for the pseudo code in example 8 is displayed in Figure 1.9.

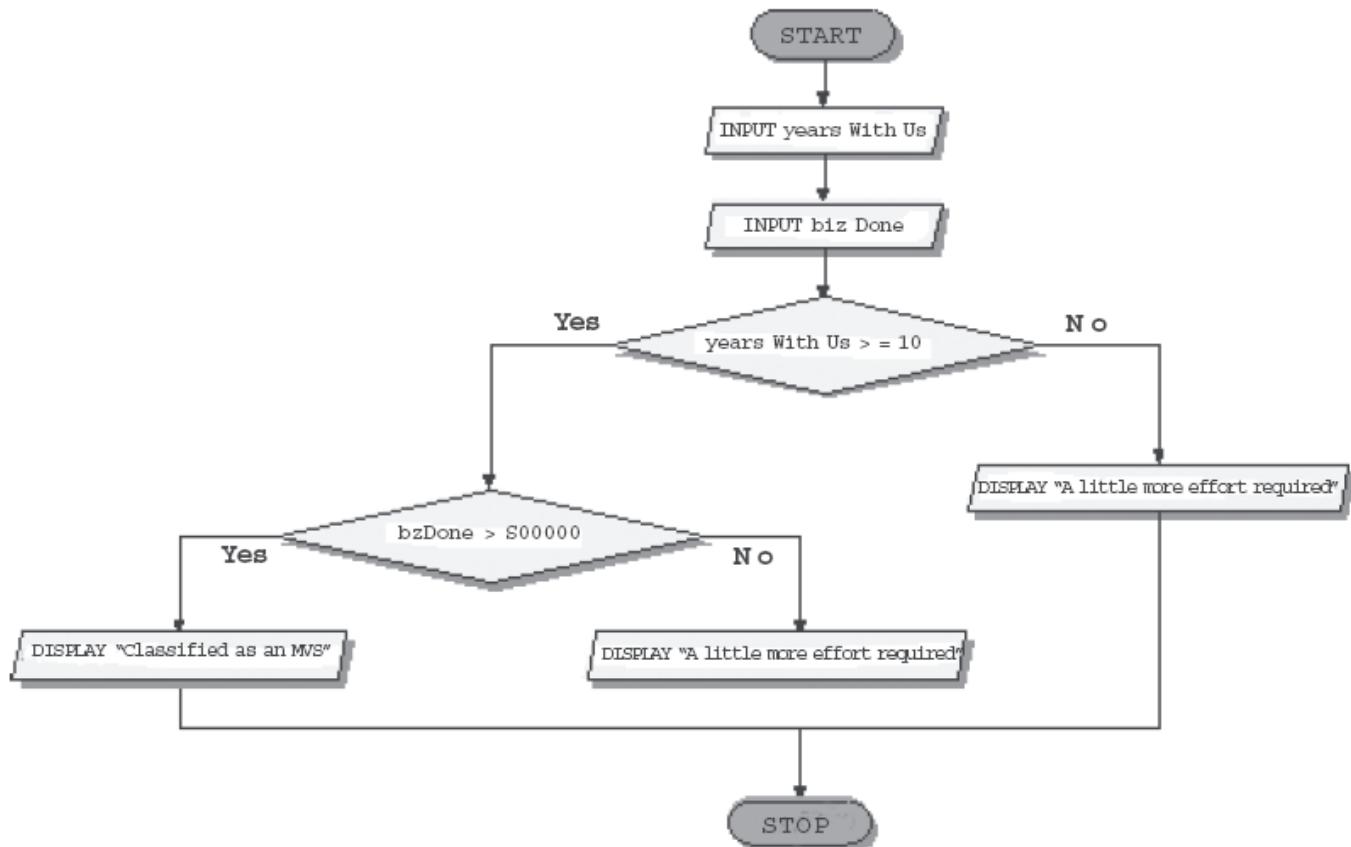


Figure 1.9: Nested IFs

Session 1

Basics of C

A company allots basic salary to its employees based on the criteria specified in Table 1.1.

Grade	Experience	Salary
E	2	2000
E	3	3000
M	2	3000
M	3	4000

Table 1.1: Basic Salary

Therefore, for example, an employee with grade E and two years of experience would receive a salary of \$2000, and an employee with the same grade but 3 years of experience would get a salary of \$3000 (Do ó, ví d , m t nhán viên có c p b c E và hai n m kinh nghi m s nh n c m c l ng 2000 ô la, và m t nhán viên có cùng c p b c nh ng có 3 n m kinh nghi m s nh n c m c l ng 3000 ô la)

The pseudo code for the problem using the AND operator would be as in Example 9 (Mã gi cho v n này s d ng toán t AND s c trình bày nh trong Ví d 9) .

Example 9:

```
BEGIN
    INPUT grade
    INPUT exp
    IF grade = "E" AND exp = 2
        salary=2000
    ELSE
        IF grade = "E" AND exp= 3
            salary=3000
        END IF
    END IF
    IF grade = "M" AND exp = 2
        salary=3000
    ELSE
        IF grade = "M" AND exp = 3
            salary=4000
    END IF
END IF
END
```

Session 1

Basics of C

Now let us consider the modified pseudo code using nested IFs in Example 10.

Example 10:

```
BEGIN
    INPUT grade
    INPUT exp
    IF grade=="E"
        IF exp=2
            salary=2000
        ELSE
            IF exp=3
                salary=3000
            END IF
        END IF
    ELSE
        IF grade=="M"
            IF exp=2
                Salary=3000
            ELSE
                IF exp=3
                    Salary=4000
                END IF
            END IF
        END IF
    END IF
```

At first look, the above code may look a little awry. Though, it is much more efficient. Tuy nhiên, nó thoát ra vì không cần thi thoảng là 2000 ô là ngay trong bước đầu tiên của câu lệnh IF đầu tiên. After this, the code is exited as it does not need to execute any of the ELSE clauses. Sau đó, mã sẽ thoát ra vì không cần thi thoảng là 2000 ô là ngay trong bước đầu tiên của câu lệnh IF đầu tiên. So the code is more efficient, and the program runs faster. Vì vậy, mã này hiệu quả hơn và chạy nhanh hơn.

➤ Loops

A computer program is a set of statements, which are normally executed one after the other. Một chương trình máy tính là một tập hợp các câu lệnh, thường được thực hiện sau nhau. It may be required to repeat certain steps a specific number of times or till a certain condition is met. Có thể cần lặp lại một số bước nhất định số lần hoặc cho đến khi một điều kiện nào đó được thỏa mãn.

It would be easier if we could write the DISPLAY statement once and ask the computer to execute it 1000 times. Số lần lặp này, chúng ta có thể viết câu lệnh DISPLAY một lần và yêu cầu máy tính thực hiện 1000 lần. In order to do this, we need to include the DISPLAY statement in a loop construct and instruct the computer to loop through the statement 1000 times. Để làm điều này, chúng ta cần viết một câu lệnh DISPLAY vào một cấu trúc vòng lặp và lặp nó 1000 lần. A rough pseudo code of what a looping construct would look like is given in Example 12. Mô tả cách mà một cấu trúc vòng lặp trông như nào sẽ được minh họa trong Ví dụ 12.

Example 12:

```
Do loop 1000 times
    DISPLAY "Scooby"
end loop
```

Session 1

Basics of C

The block of statement(s) [in this case just the DISPLAY statement] that appear in between the do loop and end loop statements get executed 1000 times.Không câu lệnh (trong trang này chỉ là câu lệnh DISPLAY) sẽ被执行 1000 lần.

These statements, along with the do loop and end loop statements, are called looping or iterative constructs.Các câu lệnh này, cùng với các câu lệnh do loop và end loop, được gọi là cấu trúc lặp hoặc có cấu trúc lặp.Loops enable programmers to develop concise programs which otherwise would require thousands of statements to perform.Các vòng lặp cho phép lập trình viên phát triển các chương trình ngắn, nhanh không yêu cầu hàng nghìn câu lệnh như thế này.

Here we have used Do loop...end loop to indicate a general form of a loop. Vậy, chúng ta đã sử dụng Do loop...end loop để chỉ ra một hình thức chung của vòng lặp.Another way of writing the same code is shown in Example 13.Một cách khác viết cùng mã ở cuối phần trong Ví dụ 13.

Example 13:

```
BEGIN
    cnt=0
    WHILE (cnt < 1000)
        DO
            DISPLAY "Scooby"
            cnt=cnt+1
        END DO
    END
```

The flowchart for the pseudo code in Example 13 would look as shown in Figure 1.10.

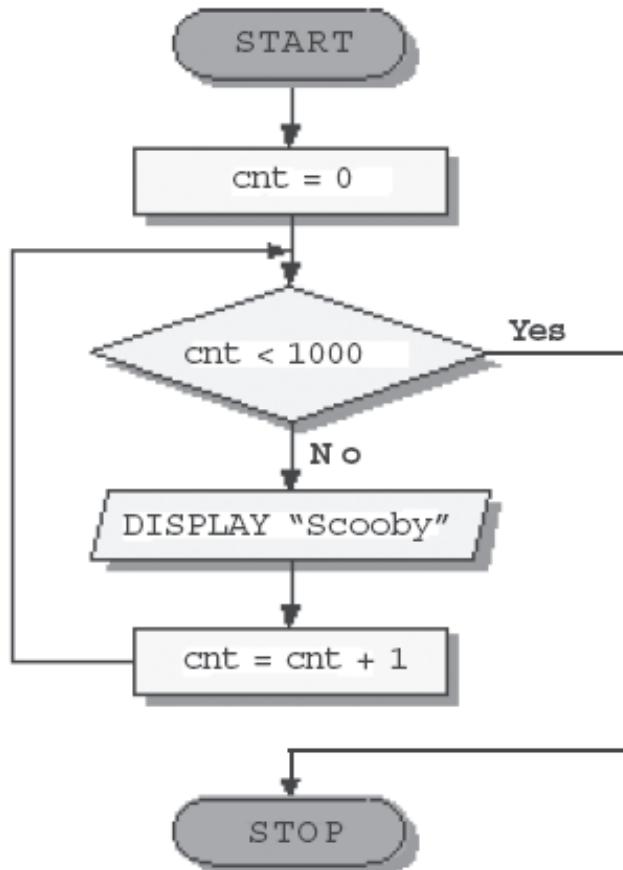


Figure 1.10: Loops

Notice that in Figure 1.10 there is no special symbol for showing loops. We use the branching symbol to check the condition and manage the flow of the program using flow-lines.

Session 1

Basics of C

Concepts



Summary

Software is a set of programs.Phần mềm là một tập hợp các chương trình.

A program is a set of instructions.Một chương trình là một tập hợp các hướng dẫn.

Code blocks form the base of any C program.Các khối mã tạo thành nền tảng của các блоки в языке C.

The C language has 32 keywords.Ngôn ngữ C có 32 từ khóa.

Steps involved in solving a problem are studying the problem in detail, gathering the relevant information, processing the information, and arriving at the results.Các bước liên quan đến việc giải quyết vấn đề bao gồm nghiên cứu chi tiết vấn đề, thu thập thông tin liên quan, xử lý thông tin và nhận kết quả.

An algorithm is a logical and concise list of steps to solve a problem.Thuật toán là một danh sách các bước có lógic và ngắn gọn để giải quyết vấn đề.

Algorithms are written using pseudo codes or flowcharts.Các thuật toán được viết bằng mã giả hoặc biểu đồ.

A pseudo code is a representation of an algorithm in a language that resembles code.Mã giả là một bài diễn đạt bằng ngôn ngữ giả định để mô phỏng thuật toán.

A flowchart is a diagrammatic representation of an algorithm.Biểu đồ dòng chảy là một bài diễn đạt hình ảnh nhằm mô tả thuật toán.

The basic selection construct is an 'IF' construct.Cấu trúc chia nhánh là một cấu trúc 'IF'. Flowcharts can be broken into parts, and connectors can be used to indicate the location of the joins.Biểu đồ dòng chảy có thể chia thành các phần, và các đường nối có thể chỉ ra vị trí của các nhánh.

When we come across a condition based on which the path of execution may branch, such constructs are referred to as selection, conditional, or branching constructs.Khi chúng ta gặp một điều kiện mà theo đó có thể chia nhánh, các cấu trúc này là các конструкции, условные или ветвящиеся конструкции.

The IF...ELSE construct enables the programmer to make a single comparison and then execute the steps depending on whether the result of the comparison is True or False.Cấu trúc IF...ELSE cho phép lập trình viên thực hiện một phép so sánh và sau đó thực hiện các bước tùy thuộc vào kết quả của phép so sánh là đúng hay sai.

A nested IF is an IF inside another IF statement.Một IF lồng nhau là một IF nằm trong một câu lệnh IF khác.

Often, it is necessary to repeat certain steps a specific number of times or till some specified condition is met.Thường thì cần phải lặp lại một số bước nhất định hoặc cho đến khi một điều kiện nào đó được thỏa mãn. Các конструкции, которые позволяют повторять определенные шаги определенное количество раз или до тех пор, пока не будет выполнено определенное условие, называются итеративными или циклическими конструкциями.



Check Your Progress

1. C allows _____ of code and data.
2. A _____ is a diagrammatic representation that illustrates the sequence of operations to be performed to arrive at a solution.
3. Flowcharts help us review and debug programs easily. (True / False)
4. A flowchart can have any number of start and stop points. (True / False)
5. A _____ is basically the execution of a sequence of statements until a particular condition is True or False.



Try It Yourself

1. Write a pseudo code and draw a flowchart to accept a value in degrees Celsius and to convert it into Fahrenheit. [Hint: $C/5 = (F-32)/9$]
2. Write a pseudo code and flowchart to accept a student's marks in Physics, Chemistry, and Biology. The total of these marks as well as the average should be displayed.

2 Variables and Data Types

Objectives

At the end of this session, you will be able to:

- *Discuss variables*
- *Differentiate between variables and constants*
- *List the different data types and make use of them in C programs*
- *Discuss arithmetic operators*

Introduction

2.1 Variables

Earlier, programmers had to write their programs in the language of the machine, t c là, b ng ngón ng c a máy tính, that is, in 1's and 0's t c là, b ng 1 và 0 . If the programmer wanted to temporarily store a value, n u l p trình viên mu n l u tr t m th i m t giá tr , the exact location where the data is stored inside the memory of the computer had to be assigned. V trí chính xác n i d li u c l u tr trong b nh c a máy tính ph i c ch nh. This storage location had to be a specific number và v trí l u tr này ph i là m t s c th or memory address. ho c a ch b nh .

Modern day languages enable us to use symbolic names known as variables, các ngôn ng hi n i cho phép chúng ta s d ng các tên ký hi u c g i là bi n. to refer to the memory location where a particular value is to be stored. ch n v trí b nh n i m t giá tr c th s c l u tr .

The data type decides the amount of memory to be assigned. Ki u d li u quy t nh l ng b nh c n c ch nh. The names that we assign to variables help us in reusing the data as and when required. Các tên mà chúng ta gán cho bi n giúp chúng ta tái s d ng d li u khi c n thi t.

Session 2

Variables and Data Types

We are familiar with the use of letters, which represent quantities in a formula. For example, the area of a rectangle is given by:

$$\text{Area} = A = \text{Length} \times \text{Breadth} = L \times B$$

The simple interest is given by:

$$\text{Interest} = I = \text{Principal} \times \text{Time} \times \text{Rate} / 100 = P \times T \times R / 100$$

The letters **A**, **L**, **B**, **I**, **P**, **T**, **R**, are all variables and are short notations that represent various values.

Consider the following example:

The sum of the marks obtained by 5 students is to be displayed. The sum can be displayed with the help of the following instruction.

Display the sum of 24, 56, 72, 36 and 82.

Once the sum is displayed, it is lost from the computer's memory. Suppose we want to calculate the average of the marks, the sum would have to be recalculated.

A better approach would be to store the result in the computer's memory, and reuse it as and when required.

sum = 24 + 56 + 72 + 36 + 82

Here, **sum** is a variable that is used to store the sum of the five numbers. When we have to calculate average, it can be done as follows:

Avg = sum / 5

In C, all variables should be declared before they can be used. Let us, for example, take the example of accepting two numbers and displaying their sum. The code for the same is provided in Example 1.

Example 1:

```
BEGIN
    DISPLAY 'Enter 2 numbers'
    INPUT A, B
    C = A + B
    DISPLAY C
END
```

Session 2

A, B, and C in the code above are variables. A, B và C trong o n mă trên là các biến. The variable names save you from remembering memory locations by address. Các tên biến giúp bạn không cần nhớ các địa chỉ bộ nhớ. When code is written and executed, the operating system takes care of allocating some memory space for these variables. Khi mã được viết và thi hành, hệ điều hành sẽ quản lý phân bổ mảng tách biệt không gian bộ nhớ cho các biến này. The operating system maps a variable name to a specific memory location. Hệ điều hành ảnh xem mảng tên biến nằm ở vị trí bộ nhớ xác định. Now, to refer to a particular value in memory, we need to specify just the variable name. Vậy giờ, chúng ta chỉ cần tên biến. In the example above, two values are accepted from the user, and these are stored in some memory location. Trong ví dụ trên, hai giá trị được nhập từ người dùng và lưu trữ trong mảng vị trí bộ nhớ. These memory locations can be accessed via the variable names A and B. Các vị trí bộ nhớ này có thể truy cập thông qua tên biến A và B. In the next step, the variables are added, and the result is stored in a third variable called C. Trong bước tiếp theo, các biến được cộng lại và kết quả lưu trữ trong một biến khác có tên là C. Finally, the value in this variable is displayed. Cuối cùng, giá trị trong biến này sẽ hiển thị.

While some languages allow the operating system to delete the contents of memory locations and allocate this memory for reuse, Trong khi một số ngôn ngữ cho phép hệ điều hành xóa nội dung của các vị trí bộ nhớ và phân bổ lại bộ nhớ này sau đó lập trình, other languages such as C require the programmer to clear unused memory locations through code. Các ngôn ngữ khác như C yêu cầu lập trình viên phải xóa các vị trí bộ nhớ không sử dụng qua mã. In both these cases, it is the operating system that takes care of allocating and deallocating memory. Trong cả hai trường hợp này, chính hệ điều hành sẽ quản lý phân bổ và giải phóng bộ nhớ.

2.2 Constants

In case of variables, the values stored in them need not be the same throughout its lifetime. Trong trường hợp các biến, các giá trị lưu trữ trong chúng không cần phải giống nhau trong suốt vòng đời chúng. The statements within this scope block can access the value of this variable. Các câu lệnh trong khung vực này có thể truy cập giá trị của biến này. They even can change the value of the variables. Chúng ta có thể thay đổi giá trị của các biến. There is need for certain items whose value can never be changed. Có những trường hợp mà giá trị của chúng không bao giờ có thể thay đổi.

A constant is a value whose worth never changes. Một số là một giá trị mà giá trị của nó không bao giờ thay đổi. For example, 5 is a constant, whose mathematical value is always 5 and can never be changed by anybody. Ví dụ, 5 là một số, có giá trị toán học luôn là 5 và không thể thay đổi bởi bất kỳ ai. Similarly, 'Black' is a constant that represents black color alone. Tuy nhiên, 'Black' là một số chỉ định cho màu đen. While 5 is termed as numeric constant, 'Black' is termed as string constant. Trong khi 5 là một số, 'Black' là một chuỗi.

2.3 Identifier

The names of variables, functions, labels, and various other user-defined objects are called identifiers. Tên của các biến, hàm, nhãn và biến riêng lẻ có thể là các ký tự danh. These identifiers can contain one or more characters. Các ký tự danh này có thể chứa một hoặc nhiều ký tự. It is compulsory that the first character of the identifier is a letter or an underscore (_). ** Sau đó, các ký tự sau có thể là chữ cái, số hoặc dấu gạch dưới (_). The subsequent characters can be alphabets, numbers, or underscores. Các ký tự tiếp theo có thể là chữ cái, số hoặc dấu gạch dưới.

Some correct identifier names are arena, s_count, marks40, and class_one. Examples of wrong identifier names are 1sttest, oh!god, and start... end.

Identifiers can be of any convenient length, but the number of characters in a variable that are recognized by a compiler varies from compiler to compiler. Các ký tự trong tên biến mà một trình biên dịch nhận ra có thể khác nhau tùy vào trình biên dịch. For example, if a given compiler recognizes the first 31 significant digits of an identifier name, then the following will appear the same to it. Ví dụ, nếu một trình biên dịch nhận ra 31 chữ số quan trọng đầu tiên của tên biến, thì các tên sau đây sẽ được xem là giống nhau.

Session 2

Variables and Data Types

```
This is a testing variable .... testing
This is a testing variable .... testing ... testing
```

Identifiers in C are case sensitive, that is, **arena** is different from **ARENA**.

2.3.1 Guidelines for Specifying Identifier Names

The rules for naming variables are different for each programming language. Các quy tắc đặt tên biến khác nhau cho mỗi ngôn ngữ lập trình. However, some conventions that are typically followed are: Tuy nhiên, một số quy tắc chung cần tuân theo là:

- Variable names must begin with an alphabet. Tên biến phải bắt đầu bằng một chữ cái.
- The first character may be followed by a sequence of letters or digits and can also include a special character like Ký tự tiên có thể theo sau bao gồm các ký hiệu như _ và # có thể bao gồm ký tự đặc biệt.
- Avoid using the letter O in places where it can be confused with the number 0, Tránh sử dụng chữ O nhầm lẫn với số 0, and similarly, the lowercase letter L can be mistaken for the number 1. và tránh sử dụng L nhầm lẫn với số 1
- Proper names should be avoided while naming variables. Nên tránh sử dụng tên riêng khi đặt tên cho các biến.
- Typically, uppercase and lowercase letters are treated as different, Thông thường, chữ hoa và chữ thường coi là khác nhau, i.e., variables ADD, add, and Add are all different. tức là, các biến ADD, add và Add đều là khác nhau.
- As case-sensitivity considerations vary with programming languages, Vì các quy tắc phân biệt chữ hoa và chữ thường khác nhau giữa các ngôn ngữ lập trình, it is advisable to maintain a standard way of naming variables. nên tên thường là duy trì một cách đặt tên biến theo tiêu chuẩn
- A variable name should be meaningful and descriptive; it should describe the kind of data it holds. Tên biến nên có ý nghĩa và mô tả; nó nên mô tả rõ ràng mà không chứa nghĩa. For example, if the sum of two numbers is to be found, Ví dụ, nếu muốn cộng hai số vào nhau, the variable storing the result may be called sum. biến lưu trữ kết quả có thể là sum. Naming it s or ab12 is not a good idea. Vì cách đặt tên là s hoặc ab12 không phải là tốt hay.

2.3.2 Keywords

All languages reserve certain words for their internal use. Tất cả các ngôn ngữ đều dành riêng một số từ cho mục đích riêng của chúng. These words hold a special meaning within the context of the particular language, and are referred to as 'keywords'. Những từ này có ý nghĩa riêng biệt trong ngữ cảnh của một ngôn ngữ và có nghĩa là 'từ khóa'. While naming variables, we need to ensure that we do not use one of these keywords as a variable name. Khi đặt tên cho các biến, chúng ta cần nhớ rằng không sử dụng một trong những từ khóa này làm tên biến.

For example, all the data types are reserved as keywords. Ví dụ, tất cả các kiểu dữ liệu đều được dành riêng làm từ khóa.

Hence, naming a variable int will generate an error, Vì vậy, việc đặt tên biến là int sẽ lỗi, but naming a variable integer will not. nhưng việc đặt tên biến là integer sẽ không gây lỗi.

Some programming languages require the programmer to specify the name of the variable as well as the type of data that is to be stored in it, before actually using a variable. This step is referred to as

Session 2

Variables and Data Types

'variable declaration'.

Concepts

2.4 Data types

Different types of data that can be stored in variables are :

➤ **Numbers**

- Whole numbers.

Example, 10 or 178993455

- Real numbers.

Example, 15.22 or 15463452.25

➤ **Names**

Example, John

➤ **Logical values**

Example, Y or N

As the data that stored in variables is of different types, it requires different amounts of memory.

The amount of memory to be assigned to a variable depends on the data type stored in it.

To assign memory for a particular piece of data, we have to declare a variable of a particular data type.

The term, declaring a variable, means that some memory is assigned to it. Thu t ng khai báo m t bi n có ngh a là m t s b nh c gán cho nó. This portion of memory is then referred to by the variable name. Ph n b nh này sau ó c tham chi u b i tên bi n. The amount of memory assigned by the operating system depends on the type of the data that is going to be stored in the variable. L ng b nh c phân b b i h i u hành ph thu c vào lo i d li u s c l u tr trong bi n. A data type, therefore, describes the kind of data that will fit into a var. Vì v y, ki u d li u mô t lo i d li u nào s phù h p v i m t bi n.

The general form of declaring a variable is:

Session 2

Variables and Data Types

Data type (variable name)

Data types, which are commonly used in programming tools, can be classified as:

1. Numeric data type - stores numeric value
2. Alphanumeric data type - stores descriptive information

These data types can have different names in different programming languages. Các kiểu dữ liệu này có thể có tên khác nhau trong các ngôn ngữ lập trình khác nhau. For example, a numeric data type is called int in C language while Visual Basic refers to it as integer. Ví dụ, một kiểu dữ liệu số là int trong ngôn ngữ C trong khi Visual Basic gọi nó là integer. Similarly, an alphanumeric data type is named char in C while in Visual Basic it is named string. Tuy nhiên, một kiểu dữ liệu alphanumeric trong C có tên là char trong C trong khi trong Visual Basic nó là string. In any case, the data stored is the same. Trong bất kỳ trình lập trình nào, dữ liệu cũng như nhau. The only difference is that the variables used in a tool have to be declared with the name of the data type supported by that tool. Số khác biệt duy nhất là các biến có thể có tên trong một công thức hoặc khai báo với tên của kiểu dữ liệu công thức.

C has five basic data types. All other data types are based upon one of these types. The five basic types are:

- `int` is an integer, typically representing the natural size of **integers**.
- `float` and `double` are used for floating point numbers. `float` occupies 4 bytes and can hold numbers with six digits of precision, whereas `double` occupy eight bytes and can hold numbers with ten digits of precision.
- `char` type occupies one byte long and is capable of holding one **character**.
- `void` is typically used to declare a function as returning **no value**. This will be clearer after the session on functions.

The size and range of these types vary with each processor type and with every implementation of the C compiler.

Note: Floating point numbers are used to represent values, which require a decimal point precision.

- **Type int**

The data type, which stores numeric data, is one of the basic data types in any programming language. It consists of a sequence of one or more digits.

For example in C, to store an integer value in a variable called ‘num’, the declaration would be as follows:

```
int num;
```

The variable ‘num’ cannot store any other type of data like “Alan” or “abc”. This numeric data type

Session 2

Variables and Data Types

allows integers in the range `-32768` to `32767` to be stored. The OS allocates 16 bits (2 bytes) to a variable that is declared to be of type `int`. Examples: `12322`, `0`, `-232`.

If we assign the value `12322` to `num`, then `num` is a float variable and `12322` is a float constant.

➤ Type `float`

A variable of type `float` is used for storing values containing decimal places. The compiler differentiates between the data types, `float` and `int`.

The main difference between the two is that `int` data type includes only whole numbers, whereas `float` data type can hold either whole or fractional numbers.

For example, in C, to store a `float` value in a variable called ‘`num`’, the declaration would be as follows:

```
float num;
```

Variable declared to be of type `float` can store decimal values with a precision of upto 6 digits. The variable is allocated 32 bits (4 bytes) of memory. Examples: `23.05`, `56.5`, `32`.

If we assign the value `23.5` to `num`, then `num` is a float variable and `23.5` is a float constant.

➤ Type `double`

The type `double` is used when the value to be stored exceeds the size limit of type `float`. Variables of the type `double` can roughly store twice the number of digits as `float` type.

The precise number of digits the `float` or `double` data type can store depends upon the particular computer system.

Numbers stored in the data type `float` or `double` are treated identically by the system in terms of calculation. However, using the type `float` saves memory as it takes only half the space as a `double` would.

`double` data type, allows a greater degree of precision (upto 10 digits). A variable declared of type `double` is allocated 64 bits (8 bytes).

For example in C, to store a `double` value in a variable called ‘`num`’, the declaration would be as follows:

```
double num;
```

Session 2

Variables and Data Types

If we assign the value `23.34232324` to `num`, then `num` is a `double` variable and `23.34232324` is a double constant.

➤ Type char

The data type `char` is used to store a single character of information.

A `char` type can store a single character enclosed within two single quotation marks. Some valid examples of `char` types are '`a`', '`m`', '`$`' '`%`'.

It is also possible to store digits as characters by enclosing them within single quotes. C ng có th l u tr các ch s d i d ng k y t b ng cách t chung trong d u nh y n. These should not be confused with the numeric values. Nh ng k y t n y kh ng n nh m l n v i c g i tr s . For example, '`1`', '`5`' and '`9`' are not to be confused with the numbers `1`, `5` and `9`. Ví d , '`1`', '`5`' và '`9`' kh ng n nh m l n v i c s `1`, `5` và `9`.

Consider the statements of C code provided below.

```
char gender;
gender='M';
```

The first line declares the variable '`gender`' of data type `char`. Dòng u ti ên khai b áo bi n '`gender`' có ki u d li u `char`. The second line stores an initial value of '`M`' in it. Dòng th hai l u tr m t g i tr kh i t o là '`M`' trong ó. The variable `gender` is a character variable and '`M`' is a character constant. Bi n `gender` là m t bi n k y t và '`M`' là m t h ng s k y t . The variable is allocated 8 bits (one byte). Bi n n y c ph àn b 8 bit (m t byte)

➤ Type void

C has a special data type called `void`. C có m t ki u d li u c bi t g i là `void`. This data type indicates to the C compiler that there is no data of any type. Ki u d li u n y cho bi t v i trinh bi en d ch C r ng kh ng có d li u c a b t k lo i n ào. In C, functions return data of some type. Trong C, c ác h àm tr v d li u c a m t ki u n ào ó. However, when a function has nothing to return, the `void` data type can be used to indicate this. Tuy n hi ên, khi m t h àm kh ng có g i tr v , ki u d li u `void` có th c s d ng ch nh i u n ày.

2.4.1 Basic and Derived Data types

The four (`char`, `int`, `float` and `double`) data types that we have discussed above are used for actual data representation in the computer's memory. These data types can be modified to fit various situations more precisely. The resulting data types are said to have been derived from these basic types.

A modifier is used to alter the base type to fit various situations. Except for `void`, all other data types can have modifiers preceding them. Modifiers used with C are `signed`, `unsigned`, `long` and `short`. All of these can be applied to character and integer data type. The `long` modifier can also be applied to `double`.

Session 2

Variables and Data Types

Concepts

Some of the modifiers are:

1. unsigned
2. long
3. short

To declare a variable of a derived type, we need to precede the usual variable declaration with one of the modifiers. Khai báo một biến có kiểu dữ liệu khác, chúng ta cần添上 khai báo biến thông thường và thêm trong các từ sau. A detailed explanation of these modifiers and how they can be used is provided below. Mời bạn đọc chi tiết về các từ sau này và cách chúng có thể sử dụng cung cấp bên dưới.

The signed and unsigned Types

- Default integer declaration assumes a signed number. Khai báo số nguyên mặc định là có dấu. The most important use of signed is to modify the char data type, `Signed` là số ký tự dữ liệu char, where char is unsigned by default. trong số char mặc định là không có dấu.

The unsigned type specifies that a variable can take only positive values. Khi không có dấu xác định rõ ràng mà biến có thể nhận được các giá trị là số. This modifier may be used with the int and float data types. Bởi vì nó có thể sử dụng với các kiểu dữ liệu int và float. Unsigned can be applied to floating data types in some cases, but this reduces the portability of the code. tính di động của mã.

By prefixing the `int` type with the word `unsigned`, the range of positive numbers can be doubled.

Consider the statements of C code provided below, which declares a variable of type `unsigned int`, and initializes this variable to `23123`.

```
unsigned int varNum;  
varNum = 23123;
```

Note that the space allocated to this type of variable remains the same. Lưu ý rằng không gian phân bổ cho kiểu biến này vẫn giữ nguyên. That is, the variable `varNum` is allocated 2 bytes since it is an `int`. Có nghĩa là, biến `varNum` chiếm 2 byte vì nó là một `int`. However, the values that an `unsigned int` supports range from 0 to 65535, Tuy nhiên, các giá trị mà một `unsigned int` hỗ trợ nằm trong khoảng từ 0 đến 65535, instead of -32768 to 32767 that an `int` supports. thay vì từ -32768 đến 32767 mà một `int` hỗ trợ. By default, an `int` is a signed `int`. Theo mặc định, một `int` là một `int` có dấu.

The long and short Types

- They are used when an integer of length shorter or longer than its normal length is required. Chúng có thể sử dụng khi cần một số nguyên có độ dài ngắn hơn hoặc dài hơn so với độ dài bình thường của nó. A short modifier is applied to the data type when a length shorter than the normal integer length is sufficient, Một biến có độ dài ngắn hơn so với độ dài bình thường là đủ. and a long modifier is used when a length of more than the normal integer length is required. và một biến có độ dài dài hơn so với độ dài bình thường là cần thiết.

The `short` modifier is used with the `int` data type. It modifies the `int` data type so that it occupies

Session 2

Variables and Data Types

less memory place. Therefore, while a variable of type `int` occupies 16 bits (2 bytes), a variable of type `short int` (or simply `short`), occupies 8 bits (1 byte), and allows numbers in the range `-128` to `127`.

The `long` modifier is used to accommodate a wider range of numbers. It can be used with the `int` as well as the `double` data types. When used with the `int` data type, the variable supports numbers between `-2,147,483,647` and `2,147,483,647`, and occupies 32 bits (4 bytes). Similarly, a ‘long double’ type of variable occupies 128 bits (16 bytes).

A `long int` variable is declared as follows:

```
long int varNum;
```

It can also be declared simply as `long varNum` as well. A long integer can be declared as `long int` or just `long`. Similarly, a short integer can be declared as `short int` or `short`.

Given below is a table, which shows the range of values for the different data types and the number of bits taken. This is based on the ANSI standard.

Type	Approximate Size in Bits	Minimal Range
char	8	-128 to 127
unsigned	8	0 to 255
signed char	8	-128 to 127
int	16	-32,768 to 32,767
unsigned int	16	0 to 65,535
signed int	16	Same as int
short int	16	Same as int
unsigned short int	8	0 to 65,535
signed short int	8	Same as short int
signed short int	8	Same as short int
long int	32	-2,147,483,647 to 2,147,483,647
signed long int	32	0 to 4,294,967,295
unsigned long int	32	0 to 4,294,967,295
float	32	Six digits of precision
double	64	Ten digits of precision
long double	128	Ten digits of precision

Table 2.1: Data Types and their range

The following example shows declarations of the above-discussed types.

Session 2

Example 2:

```
main ()  
{  
    char abc; /*abc of type character */  
    int xyz; /*xyz of type integer */  
    float length; /*length of type float */  
    double area; /* area of type double */  
    long liteyrs; /*liteyrs of type long int */  
    short arm; /*arm of type short integer*/  
}
```

Concepts

Let us take the same example which we dealt with in the last session to add two numbers and display their sum. Hãy l y ví d gi ng nh chung ta ā th o lu n trong phiên tr c c ng hai s và hi n th t ng c a chung. The pseudo code for the same is given in example 3. Mă gi cho i u này c a ra trong ví d 3.

Example 3:

```
BEGIN  
    INPUT A, B  
    C = A + B  
    DISPLAY C  
END
```

In this example, the values for two variables, A and B, are accepted. Trong ví d này, các giá tr cho hai bi n A và B c ch p nh n. The values are added, and the sum is stored in C using the statement $C = A + B$. Các giá tr c c ng l i, và t ng c l u tr trong C b ng cách s d ng câu l nh $C = A + B$. In this statement, A and B are variables, and the symbol + is called an operator. Trong câu l nh này, A và B là các bi n, và k y hi u + c g i là toán t . Let us discuss the various arithmetic operators available in C in the following section. Hãy cùng th o lu n v các toán t s h c khac nhau có s n trong C ph n ti p theo. However, there are other types of operators available in C which will be covered in the next session. Tuy nhiên, còn có các lo i toán t khac có s n trong C mà s c c p trong phiên ti p theo.

2.5 Arithmetic Operators

Arithmetic operators are used to perform numerical operations. They are divided into two classes: **unary** and **binary** arithmetic operators.

Table 2.2 lists the arithmetic operators and their action.

Unary Operators	Action	Binary Operators	Action
-	Unary minus	+	Addition
++	Increment	-	Subtraction
--	Decrement	*	Multiplication
		%	Modulus

Session 2

Variables and Data Types

Unary Operators	Action	Binary Operators	Action
		/	Division
		^	Exponentiation

Table 2.2: Arithmetic operators and their action

➤ Binary Operators

In C, the binary operators work as they do in other languages. Trong C, các toán tử nhau phân họ tinh gi ng nhau trong các ngôn ngữ khác. Operators like +, -, * and / can be applied to almost any built-in data type allowed by C. Các toán tử nhau +, -, * và / có thể áp dụng cho hằng số và các biến số. Khi / áp dụng cho một số nguyên hoặc ký tự, kết quả sẽ là số không có phần thập phân nào sau dấu chấm. For example, 5/2 will equal to 2 in integer division. Ví dụ, 5/2 sẽ bằng 2 trong phép chia số nguyên. The modulus operator (%) gives the remainder of an integer division. Toán tử modulus (%) cho phép chia số nguyên. For example, 5%2 will give 1. Ví dụ, 5%2 sẽ cho 1. However, % cannot be used with floating point types. Tuy nhiên, % không thể sử dụng với các kiểu số thực.

Let us consider an example for the exponentiation operator.

9 ^ 2

Here, 9 is the base and 2 is the exponent.

The number to the left of '^' is known as the base and the number to the right of '^' is known as the exponent.

The result of 9^2 evaluates to is $9 * 9 = 81$.

Another example is:

5 ^ 3

This effectively means:

$5 * 5 * 5$

Thus, $5^3 = 5 * 5 * 5 = 125$.

Note: The programming languages like Basic, supports the ^ operator to carry out exponentiation. However, ANSI C does not support the symbol, ^, for exponentiation. The alternate way to calculate exponentiation in C is to use `pow()` function defined in `math.h`. The syntax to carry out the same is as follows:

Session 2

Variables and Data Types

Concepts

```
#include<math.h>
void main(void)
{
    ...
    /* The following function will calculate x to the power y */
    z = pow(x,y);
    ...
}
```

The following example shows all the binary operators used in C. Note that the functions `printf()` and `getchar()` are not yet covered. We will be discussing the same in the forthcoming sessions.

Example 4:

```
#include<stdio.h>
main()
{
    int x,y;
    x = 5;
    y = 2;
    printf("The integers are : %d & %d\n", x, y);
    printf("The addition gives : %d\n", x+y);
    printf("The subtraction gives : %d\n", x-y);
    printf("The multiplication gives : %d\n", x*y);
    printf("The division gives : %d\n", x/y);
    printf("The modulus gives : %d\n", x%y);
    getchar();
}
```

The above code will produce the following output:

```
The integers are : 5 & 2
The addition gives : 7
The subtraction gives : 3
The multiplication gives : 10
The division gives : 2
The modulus gives : 1
```

Session 2

Variables and Data Types

➤ Unary Arithmetic Operators

The unary arithmetic operators are unary minus (-), increment operator (++) and decrement operator (--).

- **Unary Minus (-)**

The symbol is the same as used for binary subtraction. Unary minus is used to indicate or change the algebraic sign of a value. For Example,

```
a = -75;
```

```
b = -a;
```

The above assignments result in `a` being assigned the value - 75 and `b` being assigned the value 75 ($-(-75)$). The minus sign used in this way is called the unary operator because it takes just one operand.

Strictly speaking, there is no unary + in C. Hence, an assignment statement like,

```
invld_pls = +50;
```

where, `invld_pls` is an integer variable, is not valid in standard C. However, many compilers do not object to such usage.

- **Increment and Decrement Operators**

C includes two useful operators that are generally not found in other computer languages. These are the increment operator (++) and the decrement operator (--). The operator ++ adds 1 to its operand, whereas the operator -- subtracts 1 from its operand.

To be precise,

```
x = x + 1;
```

can be written as

```
x++;
```

and

```
x = x - 1;
```

Session 2

Variables and Data Types

can be written as

```
x--;
```

Both these operators may either precede or follow the operand, i.e.

```
x = x + 1;
```

can be represented as

```
x++ or ++x;
```

Similar operations hold true for -- operator.

The difference between pre-fixing and post-fixing the operator is useful when it is used in an expression. S khác bi t gi a vi c ti nt và h u t toán t là h u ích khi nó c s d ng trong m t bi u th c. When the operator precedes the operand, C performs the increment or decrement operation before using the value of the operand. Khi toán t ng tr c toán h ng, C th c hi n phép t ng ho c gi m tr c khi s d ng giá tr c a toán h ng. This is known as pre-fixing. i u này c g i là ti nt . If the operator follows the operand, the value of the operand is used before incrementing or decrementing. N u toán t ng sau toán h ng, giá tr c a toán h ng s c s d ng tr c khi t ng ho c gi m nó.

Consider the following:

```
a = 10;  
b = 5;  
c = a * b++;
```

In the above expression, the current value of b is used for the product and then the value of b is incremented. That is, c is assigned 50 (a*b) and then the value of b is incremented to 6.

If, however, the above expression was

```
c = a * ++b;
```

the value stored in c would be 60, because b would first be incremented by 1 and then the value stored in c (10*6).

In a context where the incrementing or the decrementing effect alone is required, the operators can be used either as postfix or prefix.

Most C compilers produce very fast, efficient object code for increment and decrement operations. H u h t các trình biên d ch C t o ra mă it ng r t nhanh và hi u qu cho các phép t ng và gi m. This code is better than that generated by using the equivalent assignment statement. Mă này t t h n mă c t o ra b ng cách s d ng câu l nh gán t ng ng. So, increment and decrement operators should be used whenever Vì v y, các toán t t ng và gi m nén c s d ng b t c khi nào co the



Summary

- Most often, an application needs to handle data; it needs some place where this data can be temporarily stored. This 'place' where this data is stored is called the memory.
- Modern day languages enable us to use symbolic names known as variables, to refer to the memory location where a particular value is to be stored.
- There is no limit to the number of memory locations that a program can use.
- A constant is a value whose worth never changes.
- The names of variables, functions, labels, and various other user-defined objects are called identifiers.
- All languages reserve certain words for their internal use. They are called keywords.
- The main data types of C are character, integer, float, double float and void.
- Modifiers are used to alter the basic data types so as to fit into various situations. Unsigned, short and long are the three modifiers available in C.
- C supports two types of Arithmetic operators: Unary and Binary.
- Increment(++) and decrement(--) are unary operators acting only on numeric variables.
- Arithmetic binary operations are + - * / % which act only on numeric constants, variables or expressions.
- The modulus operator % acts only on integers and results in remainder after integer division.



Check Your Progress

1. C is case sensitive. (**True / False**)
2. The number 10 is a _____.
3. The first character of the identifier can be a number. (**True / False**)
4. Using the type _____ saves memory as it takes only half the space as a _____ would.
5. The _____ data type is used to indicate the C compiler that no value is being returned.
6. _____ and _____ are the two classes of arithmetic operators.
 - A. Bitwise & and |
 - B. Unary and Binary
 - C. Logical AND
 - D. None of the above
7. The unary arithmetic operators are ___ and ___ .
 - A. ++ and --
 - B. % and ^
 - C. ^ and \$
 - D. None of the above



Try It Yourself

- Match the following:

Column A	Column B
8	
10.34	Invalid Identifier Names
A B C	Integer Constants
abc	Character Constants
23	Double
12112134.86868686886	Floating Point Numbers
_A1	Valid Identifier Names
\$abc	
'A'	

Hint: Multiple items in Column A can map to a single element in Column B.

- What will be the value of the variables at the end in each of the following code statements:
 - `int a=4^4`
 - `int a=23.34`
 - `a = 10`
`b = a + a++`
 - `a=-5`
`b=-a`

Objectives

At the end of this session, you will be able to:

- Use variable, data type and arithmetic expression

Part I – For the first 1 Hour and 30 Minutes :

3.1 Variables

As we already know, variables are names given to locations in the memory of a computer which may store different values at different instances of time. Nh_ă ch_{úng} ta_ã bi_t, bi_n l_à t_{ên} c_á g_{án} cho c_ác v_ị tr_i trong b_o n_h_a c_a m_áy t_{ính} c_ó th_ể l_ú tr_ả c_ác gi_a tr_í kh_{ác} nhau v_{ào} nh_éng th_i i_m kh_{ác} nhau. In this session, let us focus on how to create and use variables. Trong phiên này, h_ãy c_Ùng ch_{úng} ta t_p trung v_{ào} c_ách t_ov_as_d ng bi_n.

3.1.1 Creating a variable

Variable creation involves the data type and the logical name for that variable. For example,

```
int currentVal;
```

In the example above, the name of variable is `currentVal` which is of integer data-type.

3.2 Data Type

A data type defines what kind of value will be stored by a particular variable. For example,

```
int currentVal;
```

In the example above, `int` specifies that `currentVal` variable will store integer value.

3.3 Arithmetic Expression

A C arithmetic expression consists of a variable name on the left-hand side of “=” and variable names and constants on the right-hand side of “=”. M_t bi_u th_c s_h c_tr_{ong} C_bao_g m_m t_tên_b i_n ph_{ía} b_{ên} tr_{ái} c_a “=” v_à c_ác t_{ên}_b i_n c_Ùng v_i c_ác h_éng_s ph_{ía} b_{ên} ph_i c_a “=” . The variables and constants appearing on the right-hand side of “=” are connected by arithmetic operators like +, -, *, and /. *C_ác b_i n_và h_éng_s xu_t hi_n ph_{ía} b_{ên} ph_i c_a “=” c_lien_k t_b i_các_toán_t s_h c_nh₊, ₋, _{*}, _/. For example, Ví d_, ,

```
delta = alpha * beta / gamma + 3.2 * 2 / 5;
```

Session 3

Variables and Data Types (Lab)

Now, Let us look at a complete program to calculate the simple interest.

- 1. Invoke the editor in which you can type the C program.**
- 2. Create a new file.**
- 3. Type the following code:**

```
#include<stdio.h>
void main()
{
    int principal, period;
    float rate, si;
    principal = 1000;
    period = 3;
    rate = 8.5;
    si = principal * period * rate / 100;
    printf("%f", si);
}
```

To see the output, follow these steps:

- 4. Save the file with the name myprograml.C**
- 5. Compile the file, myprograml.C**
- 6. Execute the program, myprograml.C**
- 7. Return to the editor.**

The sample output of the above program is shown in Figure 3.1.

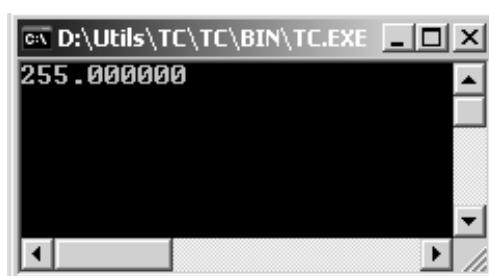


Figure 3.1: Output of myprograml.C

Session 3

Variables and Data Types (Lab)

Lab Guide

1. Create a new file.

2. Type the following code :

```
#include<stdio.h>
void main()
{
    int a, b, c, sum;
    printf("\n Enter any three numbers: ");
    scanf("%d %d %d", &a, &b, &c);
    sum = a + b + c;
    printf("\n Sum = %d", sum);
}
```

3. Save the file with the name myprogramII.C

4. Compile the file, myprogramII.C

5. Execute the program, myprogramII.C

6. Return to the editor.

The sample output of the above program will be as shown in Figure 3.2.

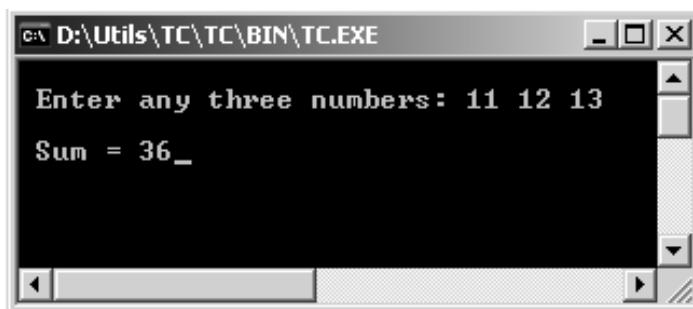


Figure 3.2: Output I of myprogramII.C

Session 3

Variables and Data Types (Lab)

Part II – For the next 30 Minutes:

Lab Guide

1. Write a C program that accepts a number and square the number.

To do this,

- a. Accept the number.
- b. Multiply the number with itself and display the square.



Try It Yourself

1. Write a C program to find the area and perimeter of a circle.
2. Write a C program that accepts the salary and age from the user and displays the same on the screen as output.

‘Information’s pretty thin stuff
unless mixed with experience’

4 Operators and Expressions

Objectives

At the end of this session, you will be able to:

- Explain Assignment Operators
- Understand Arithmetic Expressions
- Explain Relational and Logical Operators
- Understand Bitwise Logical Operators and Expressions
- Explain Casts
- Understand the Precedence of Operators

Introduction

C has a very rich set of operators. C có m t b toán t r t phong phú. Operators are the tools that manipulate data. Các toán t là công c thao tác d li u. An operator is a symbol, which represents some particular operation to be performed on the data. M t toán t là m t ký hi u, i di n cho m t phép toán c th nào ó c th c hi n trên d li u. C defines four classes of operators: arithmetic, relational, logical, and bitwise. C nh ngh a b n l p toán t : toán h c, so sánh, logic và theo bit. In addition, C has some special operators for special tasks. Ngoài ra, C còn có m t s toán t c bi t cho các tác v c bi t.

Operators operate on constants or variables, which are called operands. Các toán t ho t ng trên các h ng s ho c bi n, c g i là toán h ng. Variables have already been discussed in the previous session. Các bi n ā c th o lu n trong phiên tr c. Constants are fixed values that the program cannot alter. Các h ng s là các giá tr c nh mà ch ng trình không th thay i. C constants can be of any of the basic data types. Các h ng s trong C có th thu c b t k ki u d li u c b n nào. Operators can be classified as either unary, binary, or ternary operators. Các toán t có th c phân lo i thành toán t n, toán t nh phân ho c toán t tam. Unary operators act on only one data element, Toán t n ho t ng trên ch m t ph n t d li u, binary operators act on two data elements, toán t nh phân ho t ng trên hai ph n t d li u, and ternary operators operate on three data elements. và toán t tam ho t ng trên ba ph n t d li u.
 $c = a + b;$

Here **a**, **b**, **c** are the operands and '=' and '+' are the operators.

4.1 Expressions

An expression can be any combination of operators and operands. M t bi u th c có th là b t k s k t h p nào c a các toán t và toán h ng. Operators perform operations like addition, subtraction, comparison, etc. Các toán t th c hi n các phép toán nh c ng, tr , so sánh, v.v. Operands are the variables or values on which the operations are performed. Toán h ng là các bi n ho c giá tr mà trên ó các phép toán c th c hi n. For example, in $a + b$, a and b are operands and $+$ is the operator. Ví d , trong $a + b$, a và b là các toán h ng và $+$ là toán t . The whole thing together is an expression. Toàn b k t h p này c g i là m t bi u th c.

Session 4

Operators and Expressions

During the execution of the program, the actual value of the variable (if any) is used along with the constants that are present in the expression. Trong quá trình thi chương trình, giá trị thay thế của biến (nếu có) sẽ cùng với các hằng số có trong biểu thức. Evaluation takes place with the help of operators. Vì vậy giá trị này giúp calculate các toán tử. Thus, every C expression has a value. Vì vậy, mỗi biểu thức trong C có một giá trị.

The following are examples of expressions:

```
2
x
3 + 7
2 * y + 5
2 + 6 * (4 - 2)
z + 3 * (8 - z)
```

Roland weighs 70 kilograms, and Mark weighs k kilograms. Write an expression for their combined (or total) weight. The combined weight in kilograms of these two people is the sum of their weights, which is $70 + k$.

Evaluate the expression $4 \times z + 12$ when $z = 15$.

We replace each occurrence of z with the number 15, and simplify using the usual rules: parentheses (or brackets) first, then exponents, multiplication and division, then addition and subtraction.

```
4 * z + 12 becomes
4 * 15 + 12 = (multiplication comes before addition)
60 + 12 =
72
```

The Assignment Operator

Before looking into other operators, it is essential to discuss the assignment operator ($=$). Trước khi xem xét các toán tử khác, ta cần thiết lập một toán tử gán ($=$). It is the most common operator in any language and well known to everyone. Đây là toán tử phím bấm thường trong bất kỳ ngôn ngữ nào và cũng là biến số. In C, the assignment operator can be used with any valid C expression. Trong C, toán tử gán có thể sử dụng với bất kỳ biểu thức C hợp lệ nào. The general form of the assignment operator is: Công thức chung của toán tử gán là:

```
variable_name = expression;
```

Multiple Assignment

Many variables can be assigned the same value in a single statement. This is done by using the multiple assignment syntax. For example:

```
a = b = c = 10;
```

Session 4

Operators and Expressions

The above line of code assigns the value 10 to **a**, **b**, and **c**. However this cannot be done at the time of declaration of the variables. For example,

```
int a = int b = int c = 0;
```

This above statement gives an error because the syntax for multiple assignments is wrong here.

Arithmetic Expressions

The operations are carried out in a specific (or a particular) order, to arrive at the final value. This order is known as order of precedence (discussed later).

Mathematical expressions can be expressed in C using the arithmetic operators with numeric and character operands. Such expressions are known as **Arithmetic Expressions**. Examples of arithmetic expressions are:

```
a * (b+c/d)/22;  
++i % 7;  
5 + (c = 3+8);
```

As seen above, operands can be constants, variables, or a combination of the two. Nh [đã](#) th y [trên](#), toán h ng có th là các h ng s , bi n ho c s k t h p c a c hai. Also, an expression can be a combination of several smaller sub-expressions. Ngoài ra, m t bi u th c có th là s k t h p c a nhi u bi u th c con nh h n. For instance, in the first expression, c/d is a sub-expression, and in the third expression $c = 3 + 8$ is also a sub-expression. Ch ng h n, trong bi u th c u tiên, c/d là m t bi u th c con, và trong bi u th c th ba, $c = 3 + 8$ c ng là m t bi u th c con.

4.2 Relational Operators and Expressions

Relational operators are used to test the relationship between two variables, or between a variable and a constant. For example, the test for the larger of the two numbers, **a** and **b**, is made by means of the greater than (**>**) sign between the two operands **a** and **b** (**a > b**).

In C, **true** is any value other than zero, and **false** is zero. Expressions that use relational operators, return 0 for false and 1 for true. For example, consider the following expression:

```
a == 14;
```

This expression checks for equality, in the sense that it checks whether the value contained in the variable **a** is equal to 14 or not. The value of this expression is said to be 0 (**false**) if **a** has a value that is not equal to 14 and 1 (**true**) if it is 14.

Session 4

Operators and Expressions

The relational operators and the operation they perform are listed in Table 4.1 below.

Operator	Relational Operators Action
>	Greater than
> =	Greater than or equal
<	Less than
< =	Less than or equal
==	Equal
!=	Not equal

Table 4.1: Relational operators and their action

4.3 Logical Operators and expressions

Logical operators are symbols that are used to combine or negate expressions containing relational operators.

Expressions that use logical operators return 0 for `false` and 1 for `true`.

The logical operators and the operation they perform are listed in Table 4.2 below.

Operator	Logical Operators Action
&&	A N D
	O R
!	NOT

Table 4.2: Logical operators and their action

As for the explanation of the logical operators, consider the following. Any operator that uses two characters as a symbol should have no space between the two characters, that is `==` is not correct if it is written as `= =`.

Suppose a program has to perform certain steps if the conditions `a < 10` and `b == 7` are satisfied. This condition is coded using the relational operator in conjunction with the logical operator AND. This AND operator is represented by `&&`, and the condition will be written as:

```
(a < 10) && (b == 7);
```

Similarly the OR is operator used to check whether one of the conditions is true. It is represented by two consecutive pipe signs (`||`). If the above example is considered in such a way that the steps in the program have to be performed if either of the statements are true then the condition will be coded as:

```
(a < 10) || (b == 7);
```

Session 4

Operators and Expressions

The third logical operator `NOT` is represented by a single exclamation mark (`!`). This operator reverses the true value of the expression. For example, if a program has to do some steps if a variable `s` is less than 10, the condition can be written as

```
(s < 10);
```

as well as

```
(! (s >= 10)) /* s is not equal to or more than 10 */
```

Both relational and logical operators are lower in precedence than the arithmetic operators. For example, `5 > 4 + 3` is evaluated as if it is written as `5 > (4 + 3)`, that is, `4+3` will be evaluated first and then the relational operation will be carried out. The result for this will be false, that is 0 will be returned.

The following statement

```
printf("%d", 5 > 4 + 3);
```

will give

0

as 5 is less than 7 (`4 + 3`).

Concepts

4.4 Bitwise Logical Operators and Expressions

Consider an operand 12, for example. Xem xét mảng toán học là 12, chúng là số nguyên. Bitwise operators will treat this number 12 as 1100. Các toán tử bitwise sẽ coi số 12 này là 1100. Bitwise operators treat the operands as bits rather than numerical values. Các toán tử bitwise xem xét các toán học như là các bit thay vì các giá trị số. The numerical values could be of any base: decimal, octal, or hexadecimal. Các giá trị có thể thu được từ các số nào: thập phân, bát phân hoặc nhị phân. The Bitwise operator will convert the operand to its binary representation and accordingly return 1 or a 0. Toán tử bitwise sẽ chuyển đổi toán học thành biến dữ liệu phân cấp nó và trả về 1 hoặc 0.

Bitwise logical operators are `&`, `|`, `^`, `~`, etc., the details of each of which are summarized in Table 4.3 below.

Operator	Description
Bitwise AND (<code>x & y</code>)	Each bit position returns a one if bits of both operands are ones.
Bitwise OR (<code>x y</code>)	Each bit position returns a one if bits of either operand are ones.
Bitwise NOT (<code>~ x</code>)	Reverses the bits of its operand.
Bitwise XOR (<code>x ^ y</code>)	Each bit position returns a one if either bits of the operands are ones but not both.

Table 4.3: Bitwise logical operators

Session 4

Operators and Expressions

Bitwise operators treat number types as a 32-bit value, which changes bits according to the operator and then converts them back to the number, when done. For example,

The expression `10 & 15` implies `(1010 & 1111)` which returns a value `1010` which means `10`.

The expression `10 | 15` implies `(1010 | 1111)` which returns a value `1111` which means `15`.

The expression `10 ^ 15` implies `(1010 ^ 1111)` which returns a value `0101` which means `5`.

The expression `~10` implies `(~1010)` which returns a value `-11`.

4.5 Mixed Mode Expressions & Type Conversions

A mixed mode expression is one in which the operands of an operator belong to different data types. M t bi u th c h n h p là bi u th c mà các toán h ng c a m t toán t thu c các ki u d li u khác nhau. These operands are generally converted to the same type. Các toán h ng này th ng c chuy n i v cùng m t ki u. All operands are converted to the data type of the largest operand. T t c các toán h ng c chuy n i v ki u d li u c a toán h ng l n nh t. This is called type promotion. i u này c g i là khuy n khích ki u d li u.

The order of the various data types is: Th t c a các ki u d li u khác nhau là:

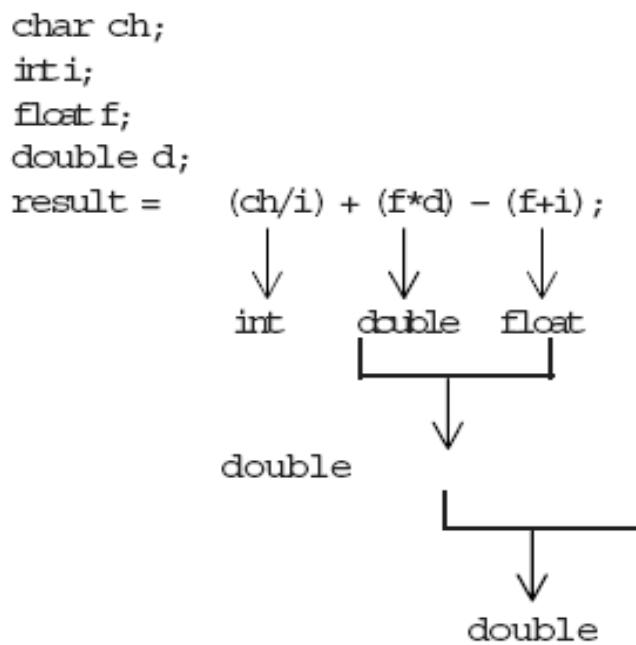
`char < int < long < float < double`

The **automatic type conversions** for evaluating an expression are tabulated below

- a. `char` and `short` are converted to `int`, and `float` is converted to `double`
- b. If either operand is `double`, the other is converted to `double`, and the result is `double`
- c. If either operand is `long`, the other is converted to `long`, and the result is `double`
- d. If either operand is `unsigned`, the other is also converted to `unsigned`, and the result is also `unsigned`
- e. Otherwise all that are left are the operands of type `int`, and the result is `int`

One rule not discussed above is when one operand is `long` and the other is `unsigned`, and the value of the `unsigned` cannot be represented by `long`. In this case, both operands are converted to `unsigned long`.

Once these conversion rules have been applied, each pair of operands is of the same type and the result of each operation is the same as the type of both operands.



In the above example, first the character `ch` is converted to an integer and float `f` is converted to double. Then the outcome of `ch/i` is converted to a double because `f*d` is double. The final result is double because, by this time, both operands are double.

4.5.1 Casts

It is normally a good practice to change all `integer` constants to `float` if the expression involves arithmetic operations with real values; otherwise some expressions may lose their true value. For example, consider

```
int x,y,z;
x = 10;
y = 100;
z = x/y;
```

In this case `z` will be assigned 0 as integer division takes place and the fractional part (0.10) is truncated.

An expression can be forced to be of a certain type by using a **cast**. The general syntax of cast is:

(type) cast

where `type` is a valid C data type. For example to make sure that an expression `a/b`, where `a` and `b` are integers, evaluates to a float, the following line of code can be written:

Session 4

Operators and Expressions

```
(float) a/b;
```

The cast operator can be used on constants, expressions as well as variables as shown below:

```
(int) 17.487;
(double) (5 * 4 / 8);
(float) (a + 7));
```

In the second example, the cast operator does not serve its purpose because it is performed only after the entire expression within the parentheses is evaluated. The expression `5 * 4 / 8` evaluates to `2` (due to integer truncation), so the resultant `double` value is equal to `2.0`.

Another example:

```
int i = 1, j = 3;
x = i / j; /* x 0.0 */
x = (float) i/(float) j; /* x 0.33 */
```

4.6 Precedence of Operators

Precedence establishes the hierarchy of one set of operators over another when an arithmetic expression has to be evaluated. Quy n u tiên xác nh th b cc am tt ph p các toán t so v it p h p khc khi m t bi u th c s h c c n c ánh giá. In short, it refers to the order in which C evaluates operators. Tóm l i, nó c p n th t mà C ánh giá các toán t . The precedence of the arithmetic operators is given in Table 4.4 below. Quy n u tiên c a các toán t s h c c a ra trong B ng 4.4 d i áy.

Operator Class	Operators	Associativity
Unary	- ++ --	Right to left
Binary	^	Left to Right
Binary	* / %	Left to Right
Binary	+ -	Left to Right
Binary	=	Right to Left

Table 4.4: Order of precedence of the arithmetic operators

Operators in the same line in the above table have the same precedence. Các toán t cung m t hàng trong b ng trên có cùng quy n u tiên. The evaluation of operators in an arithmetic expression takes place from left to right for operators having equal precedence. Vi c ánh giá các toán t trong m t bi u th c s h c di n ra t trai sang ph i i v i các toán t có cùng quy n u tiên. The operators *, /, and % have the same precedence, which is higher than that of + and - (binary). *Các toán t , / và % có cùng quy n u tiên, cao h n so v i quy n u tiên c a + và - (nh phân).

The precedence of these operators can be overridden by enclosing the required part of the expression in parentheses (). Quy n u tiên c a các toán t này có th b ghi è b ng cách t ph n c n thi t c a bi u th c trong d u ngo c n (). An expression within parentheses is always evaluated first. M t bi u th c trong d u ngo c n luôn c ánh giá tr c tiên. One set of parentheses may be enclosed within another. M t t p h p d u ngo c n có th c bao g m trong m t t p h p khc. This is called nesting of parentheses. i u này c g i là l ng ghép d u ngo c n. In such a case, evaluation is done by expanding the innermost parentheses first and then working to the outermost pair of parentheses. Trong tr ng h p này, vi c ánh giá c th c hi n b ng cách m r ng các d u ngo c n trong cung tr c và sau ó làm vi c ra ngoi c p d u ngo c n ngoi cung.

Session 4

Operators and Expressions

If there are several sets of parentheses in an expression then evaluation takes place from left to right.

Associativity tells how an operator associates with its operands. Tính k t h p cho bi t cách mà m t toán t li ên k t v i các toán h ng c a nó. For example, the unary operator associates with the quantity to its right, and in division, the left operand is divided by the right. Ví d , toán t n li ên k t v i giá tr bên ph i c a nó, và trong ph ép chia, toán h ng bên trái c chia cho toán h ng bên ph i. The assignment operator (=) associates from right to left. Toán t gán (=) li ên k t t ph i sang trái. Hence, the expression on the right is evaluated first and its value is assigned to the variables on the left. Do ó, bi u th c bên ph i c ánh giá tr c tiên và giá tr c a nó c gán cho các bi n bên trái.

Associativity also refers to the order in which C evaluates operators in an expression having same precedence. Such operators can operate either left to right or vice versa as shown in Table 4.5.

For example,

a = b =10/2;

assigns the value 5 to b which is then assigned to a. Hence the associativity is said to be from right to left. Also,

-8 * 4 % 2 - 3

is evaluated in the following sequence:

Sequence	Operation done	Result
1.	- 8 (unary minus)	negative of 8
2.	- 8 * 4	- 32
3.	- 32 % 2	0
4.	0-3	-3

In the above, the unary minus is evaluated first as it has highest precedence. The evaluation of * and % takes place from left to right. This is followed by evaluation of the binary – operator.

Order of Sub-expressions

Complex expressions could contain smaller expressions within them called sub expressions. C does not specify in what order the sub-expressions are evaluated. An expression like

a * b / c + d * c;

guarantees that the sub-expression a * b / c and d * c will be evaluated before addition. m b o r ng các bi u th c con a * b / c và d * c s c ánh giá tr c ph ép c ng. Furthermore, the left-to-right rule for multiplication and division guarantees that a will be multiplied by b and then the product will be divided by c. H n n a, quy t c t trái sang ph i cho ph ép nhân và chia m b o r ng a s c nhân v i b và sau ó tích s c chia cho c. But there is no rule for governing whether a * b / c is evaluated before or after d * c. Nh ng kh ng có quy t c nào quy nh li u a * b / c s c ánh giá tr c hay sau d * c. This option is left open to the designers of the compiler to decide. Tùy ch n này l i cho các nh à thi t k trìngh biên d ch quy t nh. The left-to-right rule just applies to a sequence of operators at the same level. Quy t c t trái sang ph i ch áp d ng cho m t chu i các toán t cung m t c p . That is, it applies to multiplication and division in a * b / c. Có ngh a là, nó áp d ng cho ph ép nhân và chia trong a * b / c. But it ends there since the next operator + has a different precedence level. Nh ng nó d ng l i ó vì toán t ti p theo + có m t c p u tiên kh ác.

Session 4

Operators and Expressions

Because of this indeterminacy, expressions whose final value depends upon the order in which the sub-expressions are evaluated should not be used. Consider the following example,

```
a * b + c * b ++ ;
```

One compiler might evaluate the left term first and use the same value of **b** in both the subexpressions. But, a second compiler might evaluate the right terms first and increment **b** before it is used in the left term.

It is preferable that increment or decrement operators should not be used on a variable that appears more than once in an expression.

Precedence between Comparison Operators

Some arithmetic operators, as we saw in the previous section, have some kind of precedence over others. There is no such precedence among comparison operators. They are therefore always evaluated left to right.

Precedence for Logical Operators

The table below presents the order of precedence for logical operators.

Precedence	Operator
1	NOT
2	AND
3	OR

Table 4.5: Order of precedence for logical operators

When multiple instances of a logical operator are used in a condition, they are evaluated from right to left.

For example, consider the following condition:

False **OR** True **AND** NOT False **AND** True

This condition gets evaluated as shown below:

1. False **OR** True **AND** [NOT False] **AND** True **NOT** has the highest precedence.
2. False **OR** True **AND** [True **AND** True]
Here, **AND** is the operator of the highest precedence and operators of the same precedence are evaluated from right to left.

Session 4

Operators and Expressions

3. False **OR** [True **AND** True]
4. [False **OR** True]
5. True

Concepts

Precedence among the Different Types of Operators

When an equation uses more than one type of operator then the order of precedence has to be established with the different types of operators.

The given table presents the precedence among the different types of operators.

Precedence	Type of Operator
1	Arithmetic
2	Comparison
3	Logical

Table 4.6: Order of precedence among different types of operators

Thus, in an equation involving all three types of operators, the arithmetic operators are evaluated first followed by comparison and then logical. The precedence of operators within a particular type has been defined earlier.

Consider the following example:

$2 * 3 + 4 / 2 > 3 \text{ AND } 3 < 5 \text{ OR } 10 < 9$

The evaluation is as shown:

1. $[2 * 3 + 4 / 2] > 3 \text{ AND } 3 < 5 \text{ OR } 10 < 9$

First the arithmetic operators are dealt with. The order of precedence for the evaluation arithmetic operators is as in Table 6.

2. $[[2 * 3] + [4 / 2]] > 3 \text{ AND } 3 < 5 \text{ OR } 10 < 9$
3. $[6 + 2] > 3 \text{ AND } 3 < 5 \text{ OR } 10 < 9$
4. $[8 > 3] \text{ AND } [3 < 5] \text{ OR } [10 < 9]$

Next to be evaluated are the comparison operators all of which have the same precedence and so

Session 4

Operators and Expressions

are evaluated from left to right.

5. True AND True OR False

The last to be evaluated are the logical operators. AND takes precedence over OR.

6. [True AND True] OR False

7. True OR False

8. True

The Parentheses (or brackets)

The precedence of operators can be modified by using the parentheses to specify which part of the equation is to be solved first.

- When there are parentheses within parentheses, the innermost parentheses are to be evaluated first.
- When an equation involves multiple sets of parentheses, they are evaluated left to right.

Consider the following example:

5+9*3^2-4 > 10 AND (2+2^4-8/4 > 6 OR (2<6 AND 10>11))

The solution is:

1. 5+9*3^2-4 > 10 AND (2+2^4-8/4 > 6 OR (True AND False))

The inner parenthesis takes precedence over all other operators and the evaluation within this is as per the regular conventions.

2. 5+9*3^2-4 > 10 AND (2+2^4-8/4 > 6 OR False)

3. 5+9*3^2-4 > 10 AND (2+16-8/4 > 6 OR False)

Next the outer parentheses is evaluated Refer to the tables for the precedence of the various operators used.

4. 5+9*3^2-4 > 10 AND (2+16-2 > 6 OR False)

Session 4

Operators and Expressions

5. $5+9*3^2-4 > 10$ AND $(18-2 > 6$ OR False)
6. $5+9*3^2-4 > 10$ AND $(16 > 6$ OR False)
7. $5+9*3^2-4 > 10$ AND (True OR False)
8. $5+9*3^2-4 > 10$ AND True
9. $5+9*9-4>10$ AND True

The expression to the left is evaluated as per the conventions.

10. $5+81-4>10$ AND True
11. $86-4>10$ AND True
12. $82>10$ AND True
13. True AND True
14. True



Summary

- C defines four classes of operators: arithmetic, relational, logical, and bitwise.
- All operators in C follow a precedence order.
- Relational operators are used to test the relationship between two variables, or between a variable and a constant.
- Logical operators are symbols that are used to combine or negate expressions containing relational operators.
- Bitwise operators treat the operands as bits rather than numerical value.
- Assignment (=) is considered an operator with right to left associativity.
- Precedence establishes the hierarchy of one set of operators over another when an expression has to be evaluated.



Check Your Progress

1. _____ are the tools that manipulate data.
A. Operators B. Operands
C. Expressions D. None of the above

2. An _____ consists of a combination of operators and operands.
A. Expression B. Functions
C. Pointers D. None of the above

3. _____ establishes the hierarchy of one set of operators over another when an arithmetic expression has to be evaluated.
A. Operands B. Precedence
C. Operator D. None of the above

4. _____ is one in which the operands of an operator belong to different data types.
A. Single Mode Expression B. Mixed Mode Expression
C. Precedence D. None of the above

5. An expression can be forced to be of a certain type by using a _____.
A. Cast B. Precedence
C. Operator D. None of the above

6. _____ are used to combine or negate expressions containing relational operators.
A. Logical Operators B. Bitwise Operators
C. Complex Operators D. None of the above



Check Your Progress

7. Bitwise logical operators are __, __, __ and __.
 - A. % , ^ , * and @
 - B. &, |, ~ and ^
 - C. !,], & and *
 - D. None of the above
8. The precedence of operators can be overridden by enclosing the required part of the expression in _____.
 - A. Curly Brackets
 - B. Caret Symbol
 - C. Parentheses
 - D. None of the above



Try It Yourself

1. Write a program to accept and add three numbers.
2. For the following values, write a program to evaluate the expression

$z = a * b + (c / d) - e * f;$

a=10

b=7

c=15.75

d=4

e=2

f=5.6

3. Write a program to evaluate the area and perimeter of the rectangle.
4. Write a program to evaluate the volume of a cylinder.
5. Write a program to evaluate the net salary of an employee given the following constraints:

Basic salary : \$ 12000

DA : 12% of Basic salary

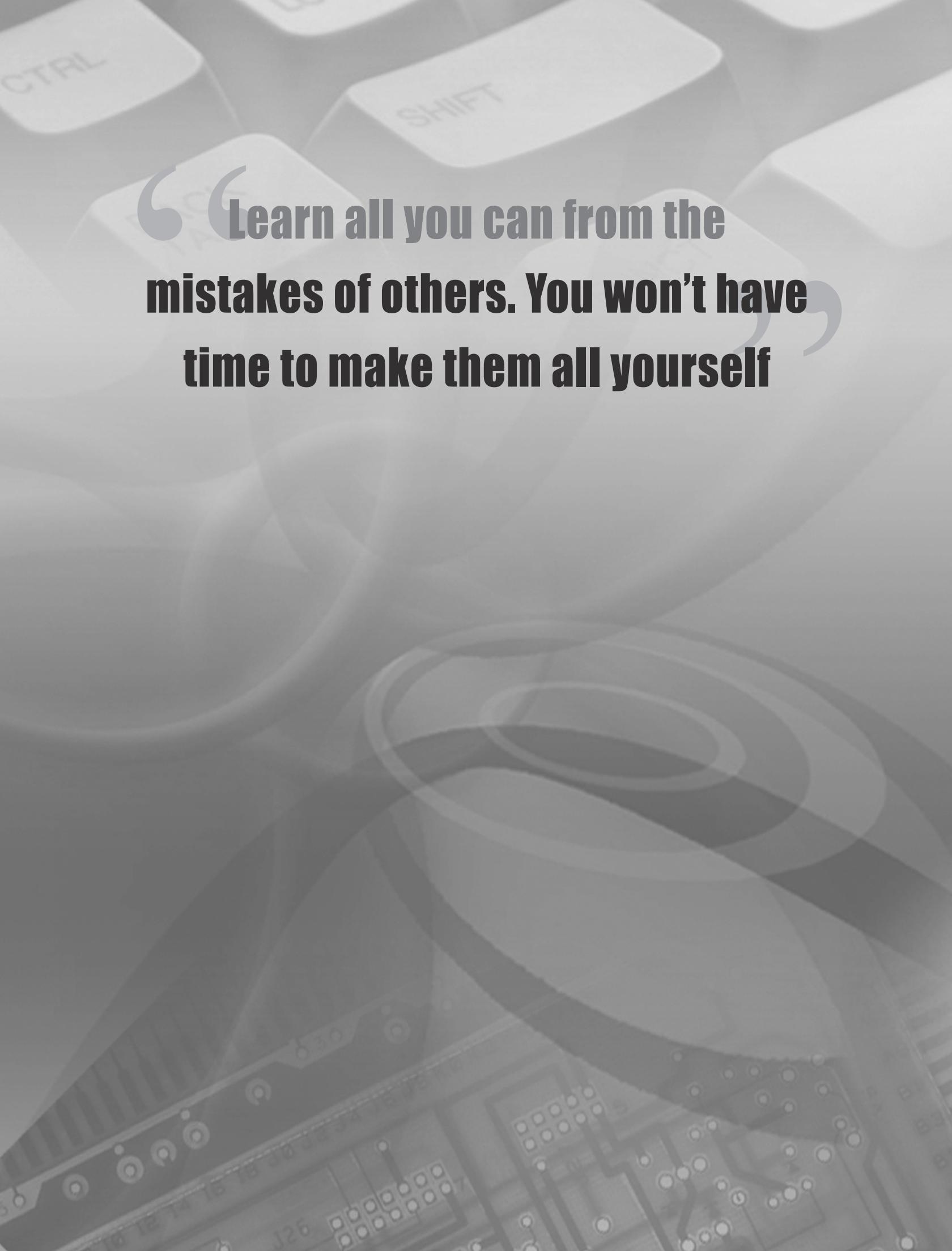
HRA : \$150

TA : \$120

Others : \$450

Tax cuts – a) PF :14% of Basic salary and b) IT: 15% of Basic salary

Net Salary = Basic Salary + DA + HRA + TA + Others – (PF + IT)



“ Learn all you can from the mistakes of others. You won't have time to make them all yourself ”