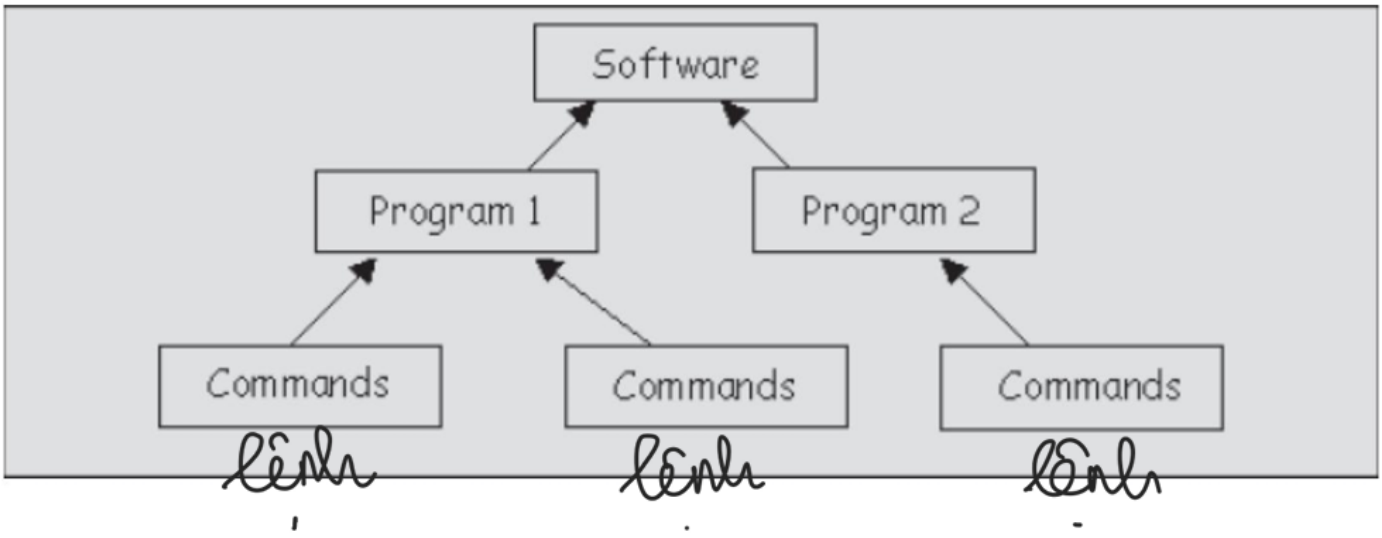
Objectives

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 symbo(kí hiệu)ls used in flowcharts*

1.1 Instructions to a Computer



Hình 1.1: Phần mềm, Chương trình và Lệnh

1.2 The C Language

C was used for systems programming(*C được sử dụng để lập trình hệ thống)*

Operating Systems(*hệ điều hành*), Interpreters(*trình thông dịch*), Editors(*trình biên tập*), Assembly programs(*chương trình lắp ráp*) are usually called systems programs(*thường được gọi là chương trình hệ thống)*

UNIX Operating System was developed using C(*Hệ điều hành UNIX được phát triển bằng C*)

C compiler produces fast and error-free object code. (*Trình biên dịch C tạo ra mã đối tượng nhanh và không có lỗi)*

C code is very portable(*mã C rất dễ di chuyển*),

1.2.2 C - A Structured Language

Ngôn ngữ có cấu trúc

C allows synthesis of code and data (*C cho phép tổng hợp mã và dữ liệu*.).

Functions (*hàm)* are used to define and separate, tasks required in a program (*được sử dụng để xác định và tách biệt các nhiệm vụ cần thiết trong một 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 được kết nối logic được xử lý như một đơn vị)*. A code block is created by placing a sequence of statements between opening and closing curly braces as shown below (*Một khối mã được tạo ra bằng cách đặt một chuỗi các câu lệnh giữa dấu ngoặc nhọn mở và đóng như minh họa bên dưới)*

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

1.3 The C Program Structure

Cấu trúc chương trình

C has few keywords, 32 to be precise (*C có ít từ khóa, chính xác là 32).*

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

- All keywords are lower cased (Tất cả các từ khóa đều được viết 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 function name (Từ khóa không thể được sử dụng cho bất kỳ mục đích nào khác, nghĩa là chúng không thể được sử dụng như một biến hoặc tên hàm)
- main() is always the first function called when a program execution begins (main luôn là hàm đầu tiên được gọi khi chương trình bắt đầu thực thi

1.3.1 Function Definition

Định nghĩa hàm

The function name is always followed by parentheses .(*Tên hàm luôn được theo sau bởi dấu ngoặc đơn).* The parentheses may or may not contain parameters (*Dấu ngoặc đơn có thể chứa hoặc không chứa tham số).*

1.3.2 Delimiters

Dấu phân cách

The function definition is followed by an pen curly brace ({) *(Định nghĩa hàm được theo sau bởi dấu ngoặc nhọn mở ({)).*This curly brace signalsthe beginning of the function (*Dấu ngoặc nhọn này báo hiệu sự bắt đầu của hàm*). Similarly a closing curly brace (}) after the statements, in the function, indicate the end of the function (một dấu ngoặc nhọn đóng (}) sau các câu lệnh trong hàm, chỉ ra sự kết thúc của hàm)

1.3.3 Statement Terminator

Bộ kết thúc câu lệnh

A statement in C is terminated with a semicolon (;) *(Một câu lệnh trong C được kết thúc bằng dấu chấm phẩy (;))*A carriage return, whitespace, or a tab is not understood by the C compiler (*ký tự trả về, khoảng trắng hoặc tab : trình biên dịch C không hiểu được.*

There can be more than one statement on the same line as long as each one of them is terminated with a semi-colon (*Có thể có nhiều hơn một câu lệnh trên cùng một dòng miễn là mỗi câu lệnh được kết thúc bằng dấu chấm phẩy.)* . A statement that does not end in a semicolon is treated as an invalid line of code in C (*Một câu lệnh không kết thúc bằng dấu chấm phẩy được coi là một dòng mã không hợp lệ trong C.)*

1.3.4 Comment Lines

Dòng chú thích

The compiler ignores them. In C, comments begin with /* and are terminated with */(*Trong C, chú thích bắt đầu bằng /* và kết thúc bằng */),* in case the comments contain multiple lines(t*rong trường hợp chú thích chứa 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 được coi như một bình luận).* In case the comment contains just a single line you can use // to indicate that it is a comment*(Trong trường hợp bình luận chỉ chứa một dòng duy nhất, bạn có thể sử dụng // để chỉ ra rằng đó là một bình luận.).*

For example

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

1.4 Compiling and Running a Program

Biên dịch và chạy chương trình

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 một chương trình C từ mã nguồn sang mã thực thi như sau:*

¬ Editor/Word Processor (*Biên tập viên/Bộ xử lý văn bản)*

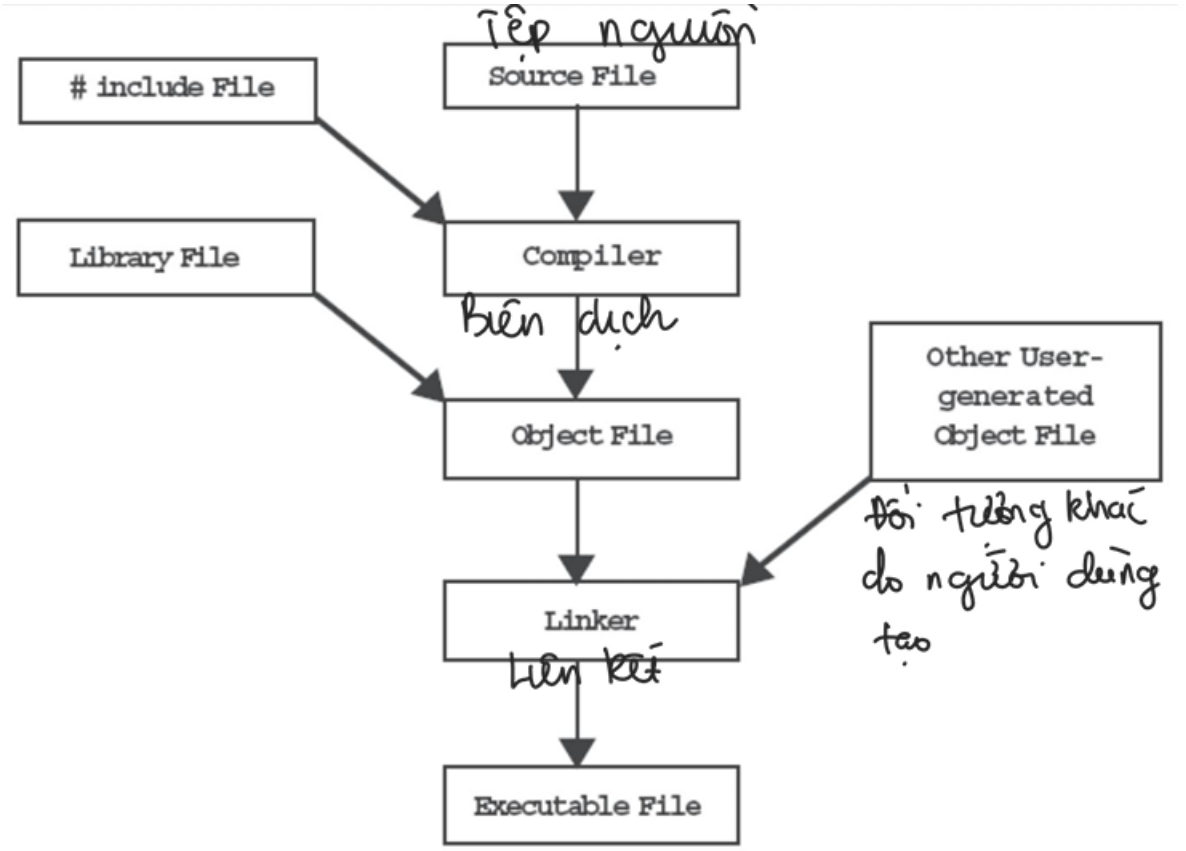
The source code is written using an editor or a word processor*.(Mã nguồn được viết bằng một trình soạn thảo hoặc chương trình xử lý văn bản.).* The code should be written in the form of standard text files, (*Mã phải được viết dưới dạng tệp văn bản tiêu chuẩn)*.Some compilers supply programming environments that include an editor (*Một số trình biên dịch cung cấp môi trường lập trình bao gồm cả trình soạn thảo văn bản)*

¬ Source Code (*mã nguồn*)

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 đầu vào cho trình biên dịch C)*

¬ C Preprocessor (*Bộ tiền xử lý)*

The source code is first passed through the C preprocessor.(*Mã nguồn đầu tiên được chuyển qua bộ tiền xử lý C.*)Preprocessors act on statements beginning with #.(*Các bộ tiền xử lý hoạt động trên các câu lệnh bắt đầu bằng #).*These statements are called directives (explained later).(*Những câu lệnh này được gọi là chỉ thị)*. The directives are usually placed at the start of the program, though they can be placed anywhere else.(*Các chỉ thị thường được đặt ở đầu chương trình, mặc dù chúng có thể được đặt ở bất kỳ đâu).*The directives are short names given to a set of code.(*Các chỉ thị là những tên ngắn được gán cho một tập hợp mã lệnh)*



Hình 1.2: Biên dịch và chạy chương trình

1.5.1 Pseudo code

Mã giả

.Note that ‘pseudo code’ is not actual code (pseudo=false) (*Lưu ý rằng 'mã giả' không phải là mã thực sự (pseudo = sai)).*Pseudo code uses a certain standard set of words, which makes it resemble a code (*Mã giả sử dụng một tập hợp từ chuẩn nhất định, khiến nó trông giống như mã lệnh*).However, unlike code, pseudo code cannot be compiled or run (*Tuy nhiên, không giống như mã lệnh, mã giả không thể được biên dịch hoặc chạy).*

Example 2:

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

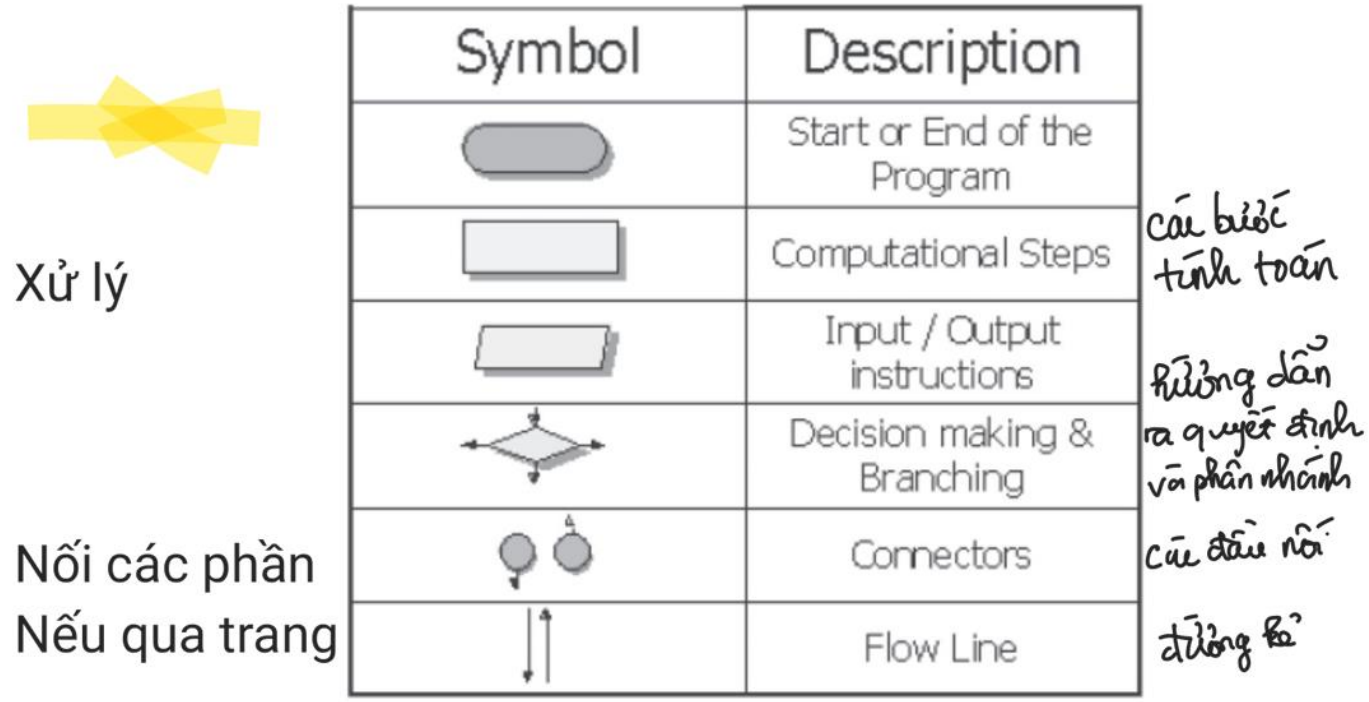
In this pseudo code, the user inputs two values, which are stored in memory and can be accessed as A and B respectively *(Trong mã giả này, người dùng nhập hai giá trị, được lưu trữ trong bộ nhớ và có thể truy cập tương ứng là A và).* Such named locations in memory are called variables *(Các vị trí được đặt tên như vậy trong bộ nhớ được gọi là biến).*

A set of instructions or steps in a pseudo code is collectively called a construct (*Một tập hợp các hướng dẫn hoặc các bước trong mã giả được gọi chung là một cấu trúc*). There are three types of

programming constructs - sequence, selection, and iteration constructs (*Có ba loại cấu trúc lập trình - cấu trúc tuần tự, cấu trúc lựa chọn và cấu trúc lặp)*

1.5.2 Flowcharts

Sơ đồ luồng



Hình 1.4: Biểu tượng sơ đồ luồng

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

- Initially concentrate a only on the logic of the problem and drawout the main path of the f lowchart (*Ban đầu chỉ tập trung vào logic của vấn đề và vạch ra lộ trình chính của sơ đồ.)*
- A flowchart must have only one STARTand one STOP point (*Sơ đồ luồng công việc chỉ được có một điểm BẮT ĐẦU và một điểm DỪNG.)*
- It is not necessary to represent each and every step of aprogram in the flowchart. Only the essential and meaningful steps need to be represented (*Không cần thiết phải thể hiện từng bước của một chương trình trong sơ đồ luồng. Chỉ có các bước thiết yếu và có ý nghĩa cần phải được trình bày)*

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 bất kỳ chương trình C nào.

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 một vấn đề bao gồm nghiên cứu chi tiết vấn đề, thu thập thông tin liên quan, xử lý thông tin và đi đến kết quả.

An algorithm is a logical and concise list of steps to solve a problem.Thuật toán là một danh sách các bước hợp lý và ngắn gọn để giải quyết một vấn đề.
Algorithms are written using pseudo codes or flowcharts.Các thuật toán được viết bằng mã giả hoặc biểu đồ dòng chảy.

A pseudo code is a representation of an algorithm in a language that resembles code.Mã giả là một biểu diễn của một thuật toán trong một ngôn ngữ giống như mã.
A flowchart is a diagrammatic representation of an algorithm.Biểu đồ dòng chảy là một biểu diễn hình ảnh của một thuật toán.

The basic selection construct is an 'IF' construct.Cấu trúc chọn cơ bản 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 được chia thành các phần, và các đầu nối có thể được sử dụng để chỉ ra vị trí của các điểm nối.

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 đường đi của việc thực hiện có thể phân nhánh, các cấu trúc như vậy được gọi là cấu trúc lựa chọn, điều kiện hoặc phân nhánh.

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 đơn lẻ và sau đó thực hiện các bước tùy thuộc vào việc 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 lặp lại một số bước nhất định một số lần cụ thể hoặc cho đến khi một điều kiện nhất định được đáp ứng.The constructs which achieve these are known as iterative or looping constructs.Các cấu trúc đạt được điều này được gọi là các cấu trúc lặp hoặc vòng lặp.

Ex01: tính tổng A+B=C

Ex02: tính tổng A+B=C vs A, B nhập từ user

assigament operator: toán tử gán

trái (ghi) = phải (đọc)

Hàm getch() - 1 hàm trong thư viện conio.h

- dùng để đọc 1 ký tự từ bàn phím mà ko cần sd phím Enter
- nó hữu ích khi cần nhận 1 ký tự đầu vào và thực hiện 1 hành động ngay lập tức

Đặc điểm

- getch() ko hiển thị ký tự mà nhập trên màn hình
- Ko yêu cầu nhấn Enter sau khi nhập
- Dùng trong các ứng dụng điều khiển = phím chẳng hạn trò chơi console

Ex03: Số giai thừa - factorial number

$$n! = n(n-1)(n-2)(n-3)\ldots\ldots 1$$

Ex04: fibonacci number - là 1 dãy số trong toán học bắt đầu = 0 và 1, mỗi số tiếp theo

bằng cộng 2 số liền kề trước nó

VD: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...

Công thức tổng quát

$F = F(n-1) + F(n-2)$

Operator $sum = n_1 + n_2$

sau đó gán $n_1 = n_2$

$n_2 = sum$

| rồi lặp lại

Ex05: amstrong number - là 1 số tự nhiên mà tổng các lũy thừa các chữ số của nó = chính nó

lũy thừa đc tính theo số lượng chữ số

VD: $153 = 1^3 + 5^3 + 3^3$

$9474 = 9^4 + 4^4 + 7^4 + 4^4$

Operator : VD: 9474

+ Tách các đơn vị chữ số:

• Hàng đơn vị 9474 % 10 (phép chia lấy dư)

9474 | 10 → dư 4

④ | 947 → gán biến mới

• Hàng chục 947 % 10 dư 7

947 | 10

⑦ | 94 → gán biến mới

. Hàng trăm 94 % 10 dư 4

94 | 10

(4) (9) → gán biến mới

. Hàng nghìn 9 % 10 dư 9

9 | 10

(9) 0

→ lấy ra đc 9 - 4 - 7 - 4

+ Đếm số chữ số : 4

+ lũy thừa các đơn vị chữ số $9^4 - 4^4 - 7^4 - 4^4$

Tính tổng sum = $9^4 + 4^4 + 7^4 + 4^4$

. So sánh sum = số đã nhập

⇒ Đây là số amstrong

Lab home - nhập 1 số và tính bình phương

Cách 1: pown = num * num

Cách 2: Dùng hàm lũy thừa pown trong thư viện math.h

```
# include <math.h>
    double result = pow (2, 9)
```

pow (số mũ, cơ số)

lab Rome _ tính chu vi _ diện tích hình tròn

$C = 2\pi r$

$S = \pi r^2$

Khai báo hằng số pi

define pi 3.14

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*

2.1 Variables

Modern day languages enable us to use symbolic names known as variables*, (các ngôn ngữ hiện đại cho phép chúng ta sử dụng các tên ký hiệu được gọi là biến*). to refer to the memory location where a particular value is to be stored. *(để chỉ đến vị trí bộ nhớ nơi một giá trị cụ thể sẽ được lưu trữ)*

2.2 Constants

Hằng số

A constant is a value whose worth never changes.*(Một hằng số là một giá trị mà giá trị của nó không bao giờ thay đổi.)*

2.3 Identifier

Mã định danh

The names of variables, functions, labels, and various other user-defined objects are called identifiers.(*Tên của các biến, hàm, nhãn và nhiều đối tượng do người dùng định nghĩa khác được gọi là các định danh*). These identifiers can contain one or more characters. (*Các định 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 (). *(**Điều bắt buộc là ký tự đầu tiên của định danh phải là một chữ cái hoặc một 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.)*

2.3.1 Guidelines for Specifying Identifier Names

Hướng dẫn chỉ định tên định danh

However, some conventions that are typically followed are (*Tuy nhiên, một số quy ước thường được 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 đầu tiên có thể được theo sau bởi một chuỗi các chữ cái hoặc số và cũng có thể bao gồm một ký tự đặc biệt như.)*
- Avoid using the letter O in places where it can be confused with the number 0,_ *Tránh sử dụng chữ O ở những chỗ có thể bị nhầm với số 0,* and similarly, the lowercase letter L can be mistaken for the number 1.(*và tương tự, chữ cái thường l có thể bị nhầm 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 được coi là khác nhau,).,* variables ADD, add, and Add are all different._ t*ức 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 chữ thường khác nhau giữa các ngôn ngữ lập trình,* it is advisable to maintain a standard way of naming variables. *nên tốt nhất 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 loại dữ liệu mà nó chứa.* For example, if the sum of two numbers is to be found, *Ví dụ, nếu tổng của hai số cần được tìm*, the variable storing the result may be called sum. *biến lưu trữ kết quả có thể được gọi là sum*. Naming it s or ab12 is not a good idea. *Việc đặt tên là s hoặc ab12 không phải là một ý hay*

2.3.2 Keywords

Từ khóa

we need to ensure that we do not use one of these keywords as a variable name*(chúng ta cần đảm bảo rằng không sử dụng một trong những từ khóa này làm tên biến.)*

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 (*Một số ngôn ngữ lập trình yêu cầu lập trình viên phải chỉ định tên biến cũng như loại dữ liệu sẽ được lưu trữ trong đó trước khi thực sự sử dụng biến*.). This step is referred to as ‘variable declaration’ *(Bước này được gọi là 'khai báo biến).*

2.4 Data types

Kiểu dữ liệu

The general form of declaring a variable is (*Dạng chung để khai báo một biến là*:):

```
Data type (variable name)
```

Notes

Kiểu dữ liệu

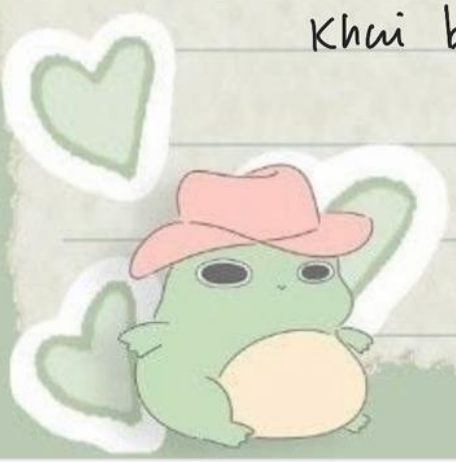
- int (số nguyên): phạm vi -32768 đến 32767
  hiển thị = %d 2 byte = 16 bit
- float (số thực): gồm số nguyên và thập phân
  hiển thị = %f 4 byte = 32 bit
- double (số thực rộng = 2 float)
  hiển thị = %lf 8 byte = 64 bit
- char (1 ký tự): lưu trữ 1 ký tự
  bao quanh bởi ngoặc đơn ' '
  hiển thị = %c
- short int = int = %d
  long int = %ld

Khai báo chuỗi dữ liệu

char tenbien [] = "...."

Hiển thị

printf ("%s", tenbien)



2.4.1 Basic and Derived Data types

Kiểu dữ liệu cơ bản và phái sinh

Modifiers used with C are signed, unsigned, long and short (*Các trình sửa đổi được sử dụng với C là signed, unsigned, long và ngắn)*

The signed and unsigned Types (*Các loại có dấu và không dấu***)**

Default integer declaration assumes a signed number.*(Khai báo số nguyên mặc định giả định một số có dấu*.) The most important use of signed is to modify the char data type, (*Sự sử dụng quan trọng nhất của signed là để sửa đổi kiểu dữ liệu* char), where char is unsigned by default. (*trong đó char mặc định là không có dấu.)*

The unsigned type specifies that a variable can take only positive values. Kiểu không có dấu xác định rằng một biến chỉ có thể nhận các giá trị dương.

By prefixing the int type with the word unsigned, the range of positive numbers can be doubled (Bằng cách thêm tiền tố unsigned vào kiểu int, phạm vi số dương có thể được nhân đôi.)

The long and short Types (Các loại dài và ngắn)

A long int variable is declared as follows (Biến int dài được khai báo như sau):

```
long int varNum;
```

It can also be declared simply as long varNum as well (*Một số nguyên dài có thể được khai báo là long int hoặc chỉ là long).* A long integer can be declared as long int or just long (*Nó cũng có thể được khai báo đơn giản là long varNum.).* Similarly, a short integer can be declared as short int or short (. *Tương tự, một số nguyên ngắn có thể được khai báo là short int hoặc short.).*

Notes

Mixed Mode Expression

(Biểu thức hỗn hợp)

thứ tự các kiểu dữ liệu

char < int < long < float < double

Type Conversions

(chuyển dữ liệu)

+ char , short → int
+ float → double
+ double + double → double
+ long + long → double
+ unsigned + unsigned → unsigned
+ long + unsigned → unsigned long

Cast (ép kiểu)

(type) cast

Notes

Ex01: khai báo biến : Kiểu dữ liệu (tên biến)

Ex02: dấy số

Ex03: nhập 1 ký tự thường, hiển thị ký tự hoa

- gán biến char ch;
- Hiển thị printf ("...", ch (ch - 32));

Ex04: nhập 1 ký tự thường, hiển thị ký tự hoa bằng hàm toupper trong thư viện ctype.h

```
# include <ctype.h>
printf ("...", toupper (ch));
```

→ chuyển từ hoa sang thường = hàm tolower()

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.

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*

C defines four classes of operators: arithmetic, relational, logical, and bitwise. *C định nghĩa bốn lớp toán tử: toán học, so sánh, logic và theo bit.*

Operators operate on constants or variables, which are called operands. C*ác toán tử hoạt động trên các hằng số hoặc biến, được gọi là toán hạng*

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

Here a, b, c are the operands and '=' and '+' are the operators *(Ở đây a, b, c là các toán hạng và '=' và '+' là các toán tử.)*

4.1 Expressions

Biểu thức

An expression can be any combination of operators and operands. Một biểu thức có thể là bất kỳ sự kết hợp nào của các toán tử và toán hạng.

Operators perform operations like addition, subtraction, comparison, etc. Các toán tử thực hiện các phép toán như cộng, trừ, so sánh, v.v. Operands are the variables or values on which the operations are performed. Toán hạng là các biến hoặc giá trị mà trên đó các phép toán được thực hiện.

The whole thing together is an expression.Toàn bộ kết hợp này được gọi là một biểu thức.

simplify using the usual rules (các quy tắc thông thường): parentheses (or brackets) first (trước tiên là ngoặc đơn), then exponents (số mũ), multiplication (phép nhân) and division (phép chia), then addition (phép cộng) and subtraction (phép trừ).

The Assignment Operator

_ toán tử gán

The general form of the assignment operator is: Công thức tổng quát của toán tử gán là:

```
variable_name = expression;
```

Many variables can be assigned the same value in a single statement (Nhiều biến có thể được gán cùng một giá trị trong một câu lệnh duy nhất)

Arithmetic Expressions _biểu thức số học

C using the arithmetic operators with numeric and character operands (sử dụng các toán tử số học với các toán hạng số và ký tự. Các biểu thức như vậy được gọi là Biểu thức số học).

4.2 Relational Operators and Expressions

Toán tử quan hệ và biểu thức

Relational operators are used to test the relationship between two variables, or between a variable and a constant (Toán tử quan hệ được sử dụng để kiểm tra mối quan hệ giữa hai biến hoặc giữa một biến và một hằng số.)

In C, 0 for false and 1 for true.

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

Toán tử logic và biểu thức

Logical operators are symbols that are used to combine or negate expressions containing relational operators (Toán tử logic là các ký hiệu được sử dụng để kết hợp hoặc phủ định các biểu thức chứa toán tử quan hệ).

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

Table 4.2: Logical operators and their action

This AND operator is represented by &&, and the condition will be written as (AND này toán tử được biểu diễn bằng && và điều kiện sẽ được viết như sau):

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

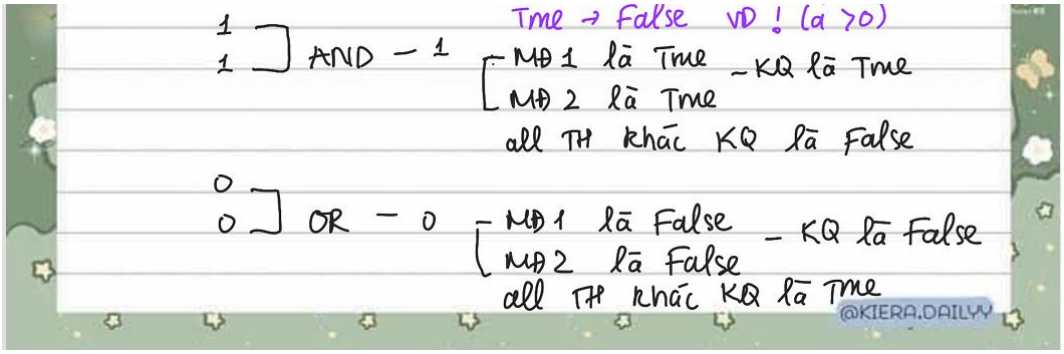
Similarly the OR is operator used to check whether one of the conditions is true (toán tử OR được sử dụng để kiểm tra xem một trong các điều kiện có đúng không.). It is represented by two consecutive pipe signs (||) (Nó được biểu diễn bằng hai dấu ống liên tiếp (||)).

if either of the statements are true then the condition will be coded as (nếu một trong hai câu lệnh là đúng thì điều kiện sẽ được mã hóa như sau):

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

The third logical operator NOT is represented by a single exclamation mark (!) (Toán tử logic thứ ba NOT được biểu diễn bằng một dấu chấm than (!).). This operator reverses the true value of the expression (Toán tử này đảo ngược giá trị thực của biểu thức).

Toán tử logic bitwise và biểu



4.4 Bitwise Logical Operators and Expressions

Bitwise operators treat the operands as bits rather than numerical values. Các toán tử bitwise xem xét các toán hạng 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 số có thể thuộc bất kỳ hệ số nào: thập phân, bát phân hoặc thập lục 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 hạng thành biểu diễn nhị phân của nó và trả về 1 hoặc 0 tương ứng.

Operation	Description
AND (N1 & N2)	Mỗi vị trí trả về 1 nếu các bit đều là 1
OR (N1 | N2)	Mỗi vị trí trả về 1 nếu 1 trong 2 bit là 1
NOT (~ N1)	Đảo ngược các bit của cái toán hạng
XOR (N1 ^ N2)	Mỗi vị trí trả về 1 nếu 1 trong 2 bit là 1 nhưng Ko phải cả 2
(a << n)	Dịch các bit sang trái, thêm bit 0 bên phải Thường sd nhân 1 số vs lũy thừa 2
(a >> n)	Dịch phải

4.5 Mixed Mode Expressions & Type Conversions

_ Biểu thức chế độ hỗn hợp & Chuyển đổi kiểu

All operands are converted to the data type of the largest operand. Tất cả các toán hạng được chuyển đổi về kiểu dữ liệu của toán hạng lớn nhất. This is called type promotion. Điều này được gọi là khuyến khích kiểu dữ liệu.

The order of the various data types is: Thứ tự của các kiểu dữ liệu khác nhau là

```
char < int < long < loat < double
```

The automatic type conversions for evaluating an expression are tabulated below (Các chuyển đổi kiểu tự động để đánh giá một biểu thức được liệt kê dưới đây)

a. char and short are converted to int, and float is converted to double (char và short được chuyển đổi thành int, và float được chuyển đổi thành double)
b. If either operand is double, the other is converted to double, and the result is double (Nếu một trong hai toán hạng là double, toán hạng kia sẽ được chuyển đổi thành double và kết quả là double)
c. If either operand is long, the other is converted to long, and the result is double (Nếu một trong hai toán hạng là dài, toán hạng kia sẽ được chuyển thành dài và kết quả là double)
d. If either operand is unsigned, the other is also converted to unsigned, and the result is also unsigned (Nếu một trong hai toán hạng không có dấu, toán hạng kia cũng được chuyển đổi thành không có dấu và kết quả cũng không có dấu.)
e. Otherwise all that are left are the operands of type int, and the result is in (Nếu không thì tất cả những gì còn lại là các toán hạng kiểu int và kết quả là int)

when one operand is long and the other is unsigned (một toán hạng là long và toán hạng kia là unsigned,), both operands are converted to unsigned long (cả hai toán hạng đều được chuyển đổi thành unsigned dài)

4.5.1 Casts

ép kiểu

The general syntax of cast is:

```
(type) cast
```

where type is a valid C data type

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

Thứ tự ưu tiên của các toán tử

4.6 Precedence of Operators

Operator Class	Operators	Associativity
Unary Binary	- ++ – – ^	Right to left 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

If there are several sets of parentheses in an expression then evaluation takes place from left to right (Nếu có nhiều bộ dấu ngoặc đơn trong một biểu thức thì việc đánh giá sẽ diễn ra từ trái sang phải.)

The assignment operator (=) associates from right to left. Toán tử gán (=) liên kết từ phải sang trái. Hence, the expression on the right is evaluated first and its value is assigned to the variables on the left. Do đó, biểu thức bên phải được đánh giá trước tiên và giá trị của nó được gán cho các biến bên trái.

the unary minus is evaluated first as it has highest precedence (phép trừ đơn được đánh giá đầu tiên vì nó có mức độ ưu tiên cao nhất). The evaluation of * and % takes place from left to right (Đánh giá * và % diễn ra từ trái sang phải). This is followed by evaluation of the binary – operator (Tiếp theo là việc đánh giá toán tử nhị phân).

Precedence between Comparison Operators: There is no such precedence among comparison operators (Không có loại ưu tiên nào như vậy giữa các toán tử so sánh). They are therefore always evaluated left to right (Do đó, chúng luôn được đánh giá từ trái sang phải.)

Precedence for Logical Operators

Precedence	Operator
1	NOT
2	A N D
3	OR

Table 4.5: Order of precedence for logical operators

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

The Parentheses (or brackets) (Dấu ngoặc đơn (hoặc dấu ngoặc vuông))

- When there are parentheses within parentheses, the innermost parentheses are to be evaluated first (Khi có dấu ngoặc đơn bên trong dấu ngoặc đơn, dấu ngoặc đơn trong cùng sẽ được đánh giá trước).
- When an equation involves multiple sets of parentheses, they are evaluated left to right (Khi một phương trình có nhiều cặp dấu ngoặc đơn, chúng sẽ được tính từ trái sang phải.)

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.

Ex1 False OR True AND NOT False AND True
Trong đó 1_ NOT , 2_ AND , 3_ OR

NOT : đảo ngược giá trị => NOT False = True

Viết lại False OR True AND True AND True
2_ AND : từ phải qua trái:
True AND True → True

T
T] AND – T

Viết lại False OR True AND True
True AND True → True

Viết lại False OR True

F
F] OR – F

↓

TRUE

Ex2: 2* 3 + 4 / 2 > 3 AND 3 < 5 OR 10 < 9
Trong đó 1_ arithmetic , 2_ relational , 3_ logic
2 * 3 + 4 / 2
nhân chia trước => 6 + 2
cộng trừ sau => 8
Viết lại 8 > 3 AND 3 < 5 OR 10 < 9
2_ relational 8 > 3 : True , 3 < 5 : True , 10 < 9 : False
Viết lại T AND T OR F
1_ NOT, 2_ AND, 3_ OR
T AND T → T
Viết lại T OR F

T
T] AND – T

F
F] OR – F

↓

T

Ex3: 5+9*3^2-4 > 10 AND (2+2^4-8|4 > 6 OR (2<6 AND 10>11))

Ngoặc đơn (ngoặc vuông) → lũy thừa → nhân, chia → cộng trừ

2<6 → TRue
10>11 → False
=) True AND False => False

Xét () thứ 2 (2+2^4-8|4 >6 OR False)

$2^4 = 16$ =) 2 + 16 - 8|4

=) 2 + 16 - 2 => 16 > 6 => True

=) True OR False

Xét 5+9*3^2-4 =) $3^2 = 9$ => 5+9*9-4

=) 5+81-4 =) 82

Viết lại 82 > 10 AND True

True AND True => True

Toán tử 1 ngôi: a + b = c

- abc : toán hạng
- =, + : toán tử

VD: a = 10 , b = a

Case 1: a ++ => Cộng sau thì thêm sau
b = a = 10
a = 11 (a += 1)

Case 2: ++a => Cộng trước thì thêm trước
a = 11 (a += 1)
b = a = 11

Ex02: ép kiểu dữ liệu

gán int n_1 | printf ("....", n_1, n_2, n_1 + (int) n_2);
float n_2

Ex03: Cấu trúc hàm printf (), scanf ()

- printf () - hàm sd hiển thị dữ liệu đầu ra

printf (" control string", argument list);

control string - chuỗi điều kiện - nằm trong ngoặc kép
- bao gồm: text character - kí tự văn bản
format commands - lệnh định dạng
non character - kí tự ko in
(tab, blanks, new lines)

Ex04: special character - kí tự đặc biệt

	printf
\\	\
\	" to printf "
%%	%

Ex05: modifier – bộ điều chỉnh
- modifier – mục dữ liệu đc căn trái
Field Width Modifier – bộ sửa đổi độ rộng trường
là 1 số nguyên, xđ độ rộng trường tối thiểu
VD %10f – lệnh định dạng kiểu float có chiều rộng 10
Precision Modifier – bộ sửa đổi chính xác
đc viết .m trong đó m là số nguyên
VD %10.3f – lệnh định dạng kiểu float có chiều rộng 10
và 3 vị trí sau dấu thập phân
'0' Modifier – đệm mặc định thực hiện = khoảng trắng
'l' Modifier – hiển thị các số nguyên dưới dạng long int
hoặc đối số có độ chính xác kép. Mặc định dạng %ld
'h' Modifier – hiển thị các số nguyên ngắn. Mặc định dạng %hd
'*' Modifier – sd nếu ng dùng Ko muốn chỉ định trước độ rộng trường
nhưng muốn CT chỉ định

Ex08: getchar() | CT nhập xuất ký tự
putchar() | đơn giản
getchar() – đọc 1 ký tự từ bàn phím
Syntax (cú pháp) int getchar(void);
Cách hoạt động – trả về mã ASCII của ký tự
– giá trị là int
VD: ch = 'A'; ch = getchar(); | output: 97
putchar(ch);
putchar() – in 1 ký tự ra màn hình
Syntax int putchar(int ch);
Cách hoạt động – nhận mã ASCII
– trả về chính ký tự đó
VD: putchar(70); | Output: F

Objectives

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

- *To understand formatted I/O functions scanf() and printf()*
- *To use character I/O functions getchar() and putchar()*

Standard input is usually the keyboard. Đầu vào tiêu chuẩn thường là bàn phím. Standard output is usually the monitor (also called the console). Đầu ra tiêu chuẩn thường là màn hình (còn gọi là bảng điều khiển).

6.1 The Header File <stdio.h>

6.1 Tệp Tiêu đề <stdio.h>

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

This is a preprocessor command. Đây là một lệnh tiền xử lý.

In standard C, the # should be in the first column. Trong ngôn ngữ C tiêu chuẩn, dấu # phải nằm ở cột đầu tiên. stdio.h is a file and is called the header file. stdio.h là một tệp và được gọi là tệp tiêu đề. It contains the macros for many of the input/output functions used in C. Nó chứa các macro cho nhiều hàm nhập/xuất được sử dụng trong C. The printf(), scanf(), putchar(), and getchar() functions are designed in such a way that they require the macros in stdio.h for proper execution. Các hàm printf(), scanf(), putchar(), và getchar() được thiết kế sao cho chúng yêu cầu các macro trong stdio.h để thực thi đúng.

6.2 Input and Output in C

The standard library in C provides two functions that perform formatted input and output. They are:

- `printf()` – for formatted output
- `scanf()` – for formatted input

These functions are called formatted functions. Các hàm này được gọi là hàm định dạng.

6.2.1 printf()

The function printf() is used to display data on the standard output – console. Hàm printf() được sử dụng để hiển thị dữ liệu trên đầu ra chuẩn – bảng điều khiển. The general format of the function is: Định dạng chung của hàm là:

```
printf("control string", argument list);
```

The argument list consists of constants, variables, expressions or functions separated by commas. Danh sách đối số bao gồm các hằng số, biến, biểu thức hoặc hàm được ngăn cách bằng dấu phẩy.

Các lệnh định dạng phải khớp với danh sách đối số về số lượng, loại và thứ tự. The control string must always be enclosed within double quotes (“ ”), which are its delimiters. Chuỗi điều khiển phải luôn được đặt trong dấu ngoặc kép (“ ”), là dấu phân cách của nó. The control string consists of one or more of three types of items which are explained below: Chuỗi điều khiển bao gồm một hoặc nhiều mục thuộc ba loại được giải thích dưới đây:

- Text characters – This consists of printable characters that are to be printed as they are. Spaces are often used to separate output fields.

- Format commands define the way the data items in the argument list are to be displayed. Các lệnh định dạng xác định cách mà các mục dữ liệu trong danh sách đối số sẽ được hiển thị. A format command begins with a % sign and is followed by a format code appropriate for the data item. Một lệnh định dạng bắt đầu với dấu % và được theo sau bởi mã định dạng phù hợp với mục dữ liệu. % is used by the printf() function to identify conversion specifications. Dấu % được sử dụng bởi hàm printf() để xác định các thông số chuyển đổi. The format commands and the data items are matched in order and typed from left to right. Các lệnh định dạng và mục dữ liệu được khớp theo thứ tự và nhập từ trái sang phải. One format code is required for every data item that has to be printed. Cần một mã định dạng cho mỗi mục dữ liệu phải được in ra.
- Nonprinting Characters – This includes tabs, blanks and new lines.

Each format command consists of one or more format codes. A format code consists of a % and a type specifier. Table 6.1 lists the various format codes supported by the printf() statement:

Format	printf()	scanf()
Single Character	%c	%c
String	%s	%s

Format	printf()	scanf()
Signed decimal integer	%d	%d
Floating point (decimal notation)	%f	%f or %e
Floating point (decimal notation)	%lf	%lf
Floating point (exponential notation)	%e	%f or %e
Floating point (%f or %e , whichever is shorter)	%g	
Unsigned decimal integer	%u	%u
Unsigned hexadecimal integer (uses “ABCDEF”)	%x	%x
Unsigned octal integer	%o	%o

Table 6.1: printf() Format Codes

In the above table c, d, f, lf, e, g, u, s, o and x are the type specifiers (Trong bảng trên , c, d, f, lf, e, g, u, s, o và x là các chỉ định kiểu).

The printing conventions of the various format codes are summarized in Table 6.2 (Các quy ước in ấn của các mã định dạng khác nhau được tóm tắt trong Bảng 6.2):

Format Code	Printing Conventions
%d	The number of digits in the integer (số chữ số trong số nguyên).
%f	The integer part of the number will be printed as such. The decimal part will consist of 6 digits. If the decimal part of the number is smaller than 6, it will be padded with trailing zeroes to the right, else it will be rounded at the right.
%e	One digit to the left of the decimal point and 6 places to the right , as in %f above (1 chữ số bên trái dấu thập phân và 6 chữ số bên phải dấu thập phân).

Table 6.2: Printing Conventions

Since %, \ and “ have special uses in the control string, if they have to be inserted as part of a text string and printed on the console, they must be used as seen in Table 6.3: Vì %, \ và " có những công dụng đặc biệt trong chuỗi điều khiển, nếu chúng cần được chèn vào như một phần của chuỗi văn bản và in ra màn hình, chúng phải được sử dụng như đã thấy trong Bảng 6.3. These symbols are interpreted differently by the printf() function unless escaped

correctly. Các ký hiệu này sẽ được hàm printf() giải thích khác đi nếu không được thoát đúng cách. To print them as literal characters, special escape sequences are used. Để in chúng như các ký tự thực, cần sử dụng các chuỗi thoát đặc biệt.For example, to print a percentage sign, you need to use %% in the control string. Ví dụ, để in ký hiệu phần trăm, bạn cần sử dụng %% trong chuỗi điều khiển.

\\	to print \ character
\	" to print " character
% %	to print % character

Table 6.3: Control String Special Characters

No	Statements	Control String	What the control string contains	Argument List	Explanation of the argument list	Screen Display
1.	`printf("%d",300);`	`%d`	Consists of format command only	300	Constant	300
2.	`printf("%d",10+5);`	`%d`	Consists of format command only	10 + 5	Expression	15
3.	`printf("Good Morning Mr. Lee.");`	`Good Morning Mr. Lee.`	Consists of text characters only	Nil	Nil	Good Morning Mr. Lee.
4.	`int count = 100;` `printf("%d",count);`	`%d`	Consists of format command only	Count	Variable	100
5.	`printf("\nhello");`	`\nhello`	Consists of nonprinting character & text characters	Nil	Nil	hello on a new line

6.	`#define str "`
	`Good Apple "`
	`……..`
	`printf("%s",str);`
		`%s`	Consists of format command only	Str	Symbolic constant	Good Apple

7.	`……`
	`int count,stud_num;`
	`count=0;`
	`stud_nim=100;`
	`printf("%d %d\n",`
	`count, stud_num);`
		`%d %d`	Consists of format command and escape sequence	count, stud_num	Two variables	0 , 100

Table 6.4: Control Strings and Format Codes

- **Modifiers for Format Commands in printf()**

The format commands may have modifiers, to suitably modify the basic conversion specifications. Các lệnh định dạng có thể có các bộ sửa đổi để điều chỉnh các thông số chuyển đổi cơ bản. The following are valid modifiers acceptable in the printf() statement. Dưới đây là các bộ sửa đổi hợp lệ có thể chấp nhận trong câu lệnh printf(). If more than one modifier is used, then they must be in the same order as given below. Nếu sử dụng nhiều hơn một bộ sửa đổi, chúng phải được đặt theo thứ tự như dưới đây.

'-' Modifier

The data item will be left-justified within its field; the item will be printed beginning from the leftmost position of its field.

Field Width Modifier

They can be used with type float, double or char array (string). The field width modifier, which is an integer, defines the minimum field width for the data item. Data items for smaller width will be output right-justified within the field. Larger data items will be printed by using as many extra posi tions as required. e.g. %10f is the format command for a type float data item with minimum field width 10.

Precision Modifier

This modifier can be used with type float, double or char array (string). Bộ sửa đổi này có thể được sử dụng với kiểu float, double hoặc mảng char (chuỗi). The modifier is written as .m where m is an integer. Bộ sửa đổi được viết dưới dạng .m, trong đó m là một số nguyên.

If used with data type float or double, the digit string indicates the maximum number of digits to be printed to the right of the decimal. Nếu sử dụng với kiểu dữ liệu float hoặc double, chuỗi số chỉ ra số lượng tối đa chữ số sẽ được in bên phải dấu thập phân. When used with a string, it indicates the maximum number of characters to be printed. Khi được sử dụng với chuỗi, nó chỉ ra số lượng ký tự tối đa sẽ được in.

If the fractional part of a type float or double data item exceeds the precision modifier, then the number will be rounded. If a string length exceeds the specified length, then the string will be trun cated (cut of at the end). Padding with zeroes occurs for numbers when the actual number of digits in a data item is less than that specified by the modifier. Similarly blanks are padded for strings. e.g. %10.3f is the format command for a type float data item, with minimum field width 10 and 3 places after the decimal. ‘0’ Modifier The default padding in a field is done with spaces. If the user wishes to pad a field with zeroes, this modifier must be used.

‘1’ Modifier

This modifier can be used to display integers as long int or a double precision argument. The cor responding format code is %ld.

‘h’ Modifier

This modifier is used to display short integers. The corresponding format code is %hd.

‘*’ Modifier

This modifier is used if the user does not want to specify the field width in advance, but wants the program to specify it. But along with this modifier an argument is required which tells what the field width should be.

Let us now see how these modifiers work. First we see their effect when used with integer data items.

We have used [and] to show where the field begins and where it ends. Chúng tôi đã sử dụng [và] để chỉ ra nơi mà trường bắt đầu và nơi nó kết thúc. When we use %d with no modifiers, we see that it uses a field with the same width as the integer. Khi chúng ta sử dụng %d mà không có bộ sửa đổi, chúng ta thấy rằng nó sử dụng một trường có cùng chiều rộng với số nguyên. When using %10d, we see that it uses a field 10 spaces wide and the number is right-justified, by default. Khi sử dụng %10d, chúng ta thấy rằng nó sử dụng một trường rộng 10 khoảng trống và số được căn bên phải theo mặc định. If we use the – modifier, the number is left-justified in the same field. Nếu chúng ta sử dụng bộ sửa đổi - , số sẽ được căn bên trái trong cùng một trường. If we use the 0 modifier, we see that the number is padded with 0 instead of blanks. Nếu chúng ta sử dụng bộ sửa đổi 0, chúng ta thấy rằng số được lấp đầy bằng 0 thay vì khoảng trắng.

In the default version of %f, we can see that there are six decimal digits, and the default specification of %e is one digit to the left of the decimal point and six digits to the right of the decimal point. Trong phiên bản mặc định của %f, chúng ta thấy có sáu chữ số thập phân, và quy định mặc định của %e là một chữ số bên trái dấu thập phân và sáu chữ số bên phải dấu thập phân.

How can you use the printf() statement to print a string which extends beyond a line of 80 characters? (Làm thế nào bạn có thể sử dụng câu lệnh printf() để in một chuỗi vượt quá một dòng 80 ký tự?) You must terminate each line by a \ symbol as shown in the example below (Bạn phải kết thúc mỗi dòng bằng ký hiệu \ như trong ví dụ dưới đây:):

Example 6

```
/* Program demonstrates how to print a long string */
#include <stdio.h>
void main()
{
    printf("aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa\aaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa\aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaa\aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa\aaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa\aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa\aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa");
}
```

6.2.2 scanf()

The scanf() function is used to accept data (Hàm scanf() được sử dụng để chấp nhận dữ liệu. 0. The general format of the scanf() function is as given below (Định dạng chung của hàm scanf() như sau).

```
scanf("control string", argument list);
```

The format used in the printf() statement are used with the same syntax in the scanf() statements too (Định dạng được sử dụng trong câu lệnh printf() cũng được sử dụng với cú pháp tương tự trong câu lệnh scanf())

The format commands, modifiers and argument list discussed for printf() is valid for scanf() also, subject to the following differences (Các lệnh định dạng, trình sửa đổi và danh

sách đối số được thảo luận cho printf() cũng có hiệu lực đối với scanf(), tùy thuộc vào những khác biệt sau:):

- Differences in argument list of between printf() and scanf()

printf() uses variable names, constants, symbolic constants and expressions, but scanf() uses pointers to variables. A pointer to a variable is a data item which contains the address of the location where the variable is stored in memory. Pointers will be discussed in detail later. When us ing scanf() follow these rules, for the argument list:

- If you wish to read in the value of a variable of basic data type, type the variable name with & symbol before it.
- When reading the value of a variable of derived data type, do not use & before the variable name.

- Differences in the format commands of the printf() and scanf()
  - There is no %g option.
  - The %f and %e format codes are in effect the same. Both accept an optional sign, a string of digits with or without a decimal point, and an optional exponent field

How scanf() works?

The scanf() function uses non-printing characters like blanks, tabs, and new lines to determine when an input field ends and when a new input field begins. Nó sử dụng các ký tự không in như khoảng trắng, tab và dòng mới để xác định khi nào một trường nhập liệu kết thúc và khi nào một trường nhập liệu mới bắt đầu. It matches the format commands with the fields in the argument list in the same order in which they are specified, skipping over any

whitespace characters in between. Nó khớp các lệnh định dạng với các trường trong danh sách đối số theo cùng một thứ tự mà chúng được chỉ định, bỏ qua bất kỳ ký tự khoảng trắng nào ở giữa. Therefore, the input can be spread over more than one line, as long as there is at least one tab, space, or newline between each input field. Do đó, dữ liệu đầu vào có thể được trải dài trên nhiều dòng miễn là có ít nhất một tab, khoảng trắng hoặc dòng mới giữa mỗi trường nhập liệu. It effectively skips white spaces and line boundaries to obtain the data. Nó bỏ qua hiệu quả các khoảng trắng và ranh giới dòng để thu thập dữ liệu

The format code %[] means that characters defined within [] can be accepted as valid string characters Mã định dạng %[] có nghĩa là các ký tự được xác định trong [] có thể được chấp nhận là các ký tự chuỗi hợp lệ. If the string BEIJING CITY is entered from the standard input device when the program is executed, the entire string will be assigned to the array line since the string is composed entirely of uppercase letters and blank spaces Nếu chuỗi BEIJING CITY được nhập từ thiết bị đầu vào chuẩn khi chương trình được thực thi, toàn bộ chuỗi sẽ được gán cho mảng line vì chuỗi được tạo thành hoàn toàn từ các chữ cái viết hoa và khoảng trắng. If the string was written as Beijing City however, then only the single letter B would be assigned to line, since the first lowercase letter (in this case, e) would be interpreted as the first character beyond the string Tuy nhiên, nếu chuỗi được viết là Beijing City, thì chỉ có một chữ cái B được gán cho line, vì chữ cái viết thường đầu tiên (trong trường hợp này là e) sẽ được hiểu là ký tự đầu tiên ngoài chuỗi.

To accept any character up to a new line character, we use the format code %[^\n], which implies that the string will accept any character except \n, which is a new line. Để chấp nhận bất kỳ ký tự nào cho đến ký tự xuống dòng, chúng ta sử dụng mã định dạng %[^\n], điều

này có nghĩa là chuỗi sẽ chấp nhận bất kỳ ký tự nào ngoại trừ \n, tức là ký tự xuống dòng. The caret (^) implies that all characters except those following the caret will be accepted as valid input characters. Ký tự mũ (^) chỉ ra rằng tất cả các ký tự ngoại trừ những ký tự đứng sau ký tự mũ sẽ được chấp nhận là ký tự đầu vào hợp lệ.

characters) will be entered from the standard input device and assigned to line. Khi hàm scanf() được thực thi, một chuỗi có độ dài không xác định (nhưng không quá 79 ký tự) sẽ được nhập từ thiết bị đầu vào tiêu chuẩn và được gán cho line. There will be no restriction on the characters that compose the string, except that they all fit on one line. Sẽ không có bất kỳ giới hạn nào về các ký tự tạo thành chuỗi, ngoại trừ việc tất cả chúng đều phải vừa trên một dòng.

The * modifier works differently in scanf() Bộ điều chỉnh * hoạt động khác trong scanf().. The asterisk is used to indicate that a field is to be ignored or skipped Dấu hoa thị được sử dụng để chỉ ra rằng một trường sẽ bị bỏ qua hoặc bỏ qua.

and the input command `scanf("%d, %f, %c", &intgr, &flt, &ch);`

Note that the commas in the conversion string match the commas in the input stream and hence will serve as delimiters. Lưu ý rằng các dấu phẩy trong chuỗi chuyển đổi khớp với các dấu phẩy trong luồng đầu vào và do đó sẽ đóng vai trò là dấu phân cách.

White space characters in the control string are normally ignored, except that it causes problems with the %c format code. Các ký tự khoảng trắng trong chuỗi điều khiển thường bị bỏ qua, ngoại trừ việc chúng gây ra vấn đề với mã định dạng %c. If we use the %c specifier, then a space is considered a valid character. Nếu chúng ta sử dụng định dạng %c, thì một khoảng trắng được coi là một ký tự hợp lệ.

6.3 Buffered I/O

. Buffered I/O – used to read and write ASCII characters (Bộ đệm I/O – được sử dụng để đọc và ghi các ký tự ASCII)

A buffer is a temporary storage area, either in the memory or on the controller card for the device. Một bộ đệm là một khu vực lưu trữ tạm thời, có thể ở trong bộ nhớ hoặc trên thẻ điều khiển cho thiết bị. In buffered I/O, characters typed at the keyboard are collected until the user presses the return or the enter key, when the characters are made available to the program, as a block. Trong I/O được đệm, các ký tự được nhập từ bàn phím được thu thập cho đến khi người dùng nhấn phím return hoặc enter, khi đó các ký tự sẽ được cung cấp cho chương trình, dưới dạng một khối.

Buffered I/O can be further subdivided into:

- Console I/O
- Buffered File I/O

The simplest (đơn giản) Console I/O functions are:

- `getchar()` – which reads (đọc) one (and only one) character (kí tự) from the keyboard.(bàn phím)
- `putchar()` – which outputs (xuất ra) a single character (kí tự duy nhất) on the screen.(ra màn hình)

……,

6.3.1 getchar()

The function getchar() is used to read input data, a character at a time from the keyboard. Hàm getchar() được sử dụng để đọc dữ liệu đầu vào, từng ký tự một từ bàn phím. In most implementations of C, getchar() buffers characters until the user types a

carriage return. Trong hầu hết các triển khai của C, getchar() lưu trữ các ký tự cho đến khi người dùng nhập phím carriage return. Therefore, it waits until the return key is pressed. Vì vậy, nó sẽ đợi cho đến khi phím return được nhấn. The getchar() function has no argument, but the parentheses must still be present. Hàm getchar() không có tham số, nhưng dấu ngoặc đơn vẫn phải có. It simply fetches the next character and makes it available to the program. Nó đơn giản chỉ lấy ký tự tiếp theo và làm cho nó có sẵn cho chương trình. We say that the function returns a value, which is a character. Chúng ta nói rằng hàm trả về một giá trị, đó là một ký tự

6.3.2 putchar()

`putchar()` is the character output function (hàm xuất kí tự) in C, which displays a character on the screen at the cursor position (hiển thị ký tự trên màn hình tại vị trí con trỏ). This function requires an argument (Hàm này yêu cầu một đối số.). The argument of the `putchar()` function can be any one of the following:

- A single character constant
- An escape sequence
- A character variable

If the argument is a constant, it must be enclosed in single quotes (Nếu đối số là hằng số, nó phải được đặt trong dấu ngoặc đơn). Table 6.5 demonstrates some of the options available in putchar() and their effect (trình bày một số tùy chọn có sẵn trong putchar() và tác dụng của chúng)

Argument	Function	Effect
character variable	`putchar(c)`	Displays the contents of character variable c
character constant	`putchar('A')`	Displays the letter A
numeric constant	`putchar('5')`	Displays the digit 5
escape sequence	`putchar('\t')`	Inserts a tab space character at the cursor position
escape sequence	`putchar('\n')`	Inserts a carriage return at the cursor position

Table 6.5: putchar() options and their effects

The difference between getchar() and putchar() is that putchar() requires an argument, while getchar() the earlier does not (Sự khác biệt giữa getchar() và putchar() là putchar() yêu cầu đối số, trong khi getchar() thì không).

Summary

- In C, I/O is performed using functions. Any program in C has access to three standard files. They are standard input file (called stdin), standard output file (called stdout) and the standard error (called stderr). Normally the standard input file is the keyboard, the standard output file is the screen and the standard error file is also the screen.
- The header file <stdio.h> contains the macros for many of the input / output functions used in C.
- Console I/O refers to operations that occur at the keyboard and the screen of your computer. It consists of formatted and unformatted functions (Console I/O đề cập đến các hoạt động diễn ra tại bàn phím và màn hình máy tính của bạn. Nó bao gồm các hàm được định dạng và không được định dạng.).
- The formatted (định dạng) I/O functions are printf() and scanf().
- The unformatted (không định dạng) functions are getchar() and putchar().
- The scanf() function is used to take the formatted input data, whereas the printf() function is used to print data in the specified format.
- The control string of printf() and scanf() must always be present inside " and " (Chuỗi điều khiển của printf() và scanf() phải luôn có bên trong và "). The string consists of a set of format commands (Chuỗi bao gồm một tập hợp các lệnh định dạng). Each format command consists of a %, an optional set of modifiers and a type specifier (Mỗi lệnh định dạng bao gồm một %, một tập hợp các trình sửa đổi tùy chọn và một trình chỉ định kiểu.).
- The major difference between printf() and scanf() is that the scanf() function uses addresses of the variables rather than the variable names themselves.
- The getchar() function reads a character from the keyboard.
- The putchar(ch) function sends the character ch to the screen.
- The difference between getchar() and putchar() is that putchar() takes an argument while getchar() does not (Sự khác biệt giữa getchar() và putchar() là putchar() nhận đối số trong khi getchar() thì không.).