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.).