CADT
បណ្ឌិត្យសភាបច្ចេកវិទ្យាឌីជីថលកម្ពុជា
Cambodia Academy of Digital Technology

IDT
វិទ្យាស្ថានបច្ចេកវិទ្យាឌីជីថល
Institute of Digital Technology

Week 04

Input and Output

Prepared by: **Leangsiv HAN**

06th March 2024

Last Week

You have learned about:

- The overview of programming in C.
- The various data types used in programming.
- The concept of tokens in programming.
- The definition and use of variable with respective rule sets.
- The various operators used in programming.

Learning Objectives

By the end of this lesson, you will be to:

- ❑ Define input and output.
- ❑ Apply Non-formatted input and output.
- ❑ Apply Formatted input and output.

Introduction

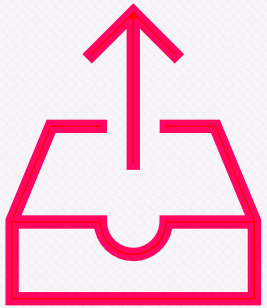
Definition:

- When we say **Input**, it means to *feed some data into a program*. An input can be given in the form of a file or from the command line. C programming provides a set of built-in functions to read the given input and feed it to the program as per requirement.

Introduction

Definition:

- When we say **Output**, it means to *display some data* on a screen, printer, or in any file. C programming provides a set of built-in functions to output the data on the computer screen as well as to save it in text or binary files.



Non-Formatted Input and Output

- Non-formatted input and output can be carried out by standard input–output library functions in C.
- These can handle *one character* at a time.
- For the **input functions**, it does (not) require <Enter> to be pressed after the entry of the character.
- For **output functions**, it prints a single character on the console.

Single character Input and output

A number of functions provide for character-oriented input and output. The declaration formats of two of these are given as follows:

```
int getchar(void); // function for character input
int putchar(int c); // function of character output
```

Single character Input and output

Check out the following example:

```c
#include <stdio.h>
int main( ) {

   int c;

   printf( "Enter a value :");
   c = getchar( );

   printf( "\nYou entered: ");
   putchar( c );

   return 0;
```

Self-study

- Research about The gets() and puts() Functions.

1. Gets
2. Puts

Formatted Input and Output

- When input and output is required in *a specified format*, the standard library functions scanf() and printf() are used.
- The **scanf() function** allows the user to *input data in a specified format*. It can accept data of *different data types*.
- The **printf() function** allows the user to *output data of different data types* on the console in a *specified format*.

Input function scanf( )

The syntax to form input function scanf()

scanf(“control_string”,variable_address); // inputting single variable

or

scanf(“control_string”,variable1_address, variable2_

address,...); // inputting multiple variable

where the **control string**, also known as the format string is a list of *format specifiers* indicating the format and type of data to be read from the standard input device, which is the keyboard, and stored in the corresponding address of variables

Format specifiers for *scanf*

Conversion code	Usual variable type	Action
%c	char	Reads a single character.
%d(%i)	int	Reads a signed decimal integer.
%e(%E)	float or double	Reads signed decimal.
%f	float or double	Reads signed decimal.
%g(%G)	float or double	Reads signed decimal.
%o	int	Reads octal value.
%p	pointer	Reads in hex address stored in pointer.
%s	array of char	Reads sequence of characters (string).
%u	int	Reads unsigned decimal integer.
%x(%X)	int	Reads unsigned hex value.
%%	none	A single % character in the input stream is expected. There is no corresponding argument.
%n	pointer to int	No characters in the input stream are matched. The corresponding argument is a pointer to an integer into which the number of characters read is placed.
[...]	array of char	Reads a string of matching characters.

When using Scanf() function

- In scanf(), the control string or format string that consists of a list of format specifiers indicates the format and type of data to be read in from the standard input device, which is the keyboard, for storing in the corresponding address of variables specified.
- There *must be the same number of format specifiers and addresses as there are input variables*.
- The format string in scanf() is enclosed in a set of quotation marks and it may contain the following:

  (a) White space

  (b) Conversion specifier field

  (c) Ordinary character string

Output function printf( )

The syntax to form output function printf()

printf("%X", variableOfXType); // outputting single variable

or

printf("%X1 %X2", variableOfXType1, variableOfXType2,...);

// outputting multiple variable

where **%X** is the *format specifier* in C. It is a way to tell the compiler what type of data is in a variable and & is the address operator in C, which tells the compiler to change the real value of this variable, stored at this address in the memory.

Format specifiers for *printf*

Conversion code	Usual variable type	Display
%c	`char`	single character
%d (%i)	`int`	signed integer
%e (%E)	`float` or `double`	exponential format
%f	`float` or `double`	signed decimal
%g (%G)	`float` or `double`	use `%f` or `%e`, whichever is shorter
%o	`int`	unsigned octal value
%p	pointer	address stored in pointer
%s	array of `char`	sequence of characters (string)
%u	`int`	unsigned decimal integer
%x (%X)	`int`	unsigned hex value
%%	none	no corresponding argument is converted, prints only a `%`.
%n	pointer to `int`	the corresponding argument is a pointer to an integer into which the number of characters displayed is placed.

List of commonly used control codes

Control code	Action
\b	Backspace
\f	Form feed
\n	New line
\r	Carriage return
\t	Horizontal tab
\'	Single quote
\0	Null

When using printf() function

- A control string, also termed as format string, and variable names are specified for the printf() output function to display the values in the variables in the desired form on the monitor screen.
- The format string in printf(), enclosed in *quotation marks*, has three types of objects: (i) Character string (ii) Conversion specifier (iii) Control code, with the programmer's option of changing the order of these three objects within the format string.
- Except for the % conversion specifier field, the other two objects, that is, the character string and the control code, are optional when the list of variables is present in printf().
- The control code and conversion specifier may be embedded within the character string.

Example: scanf() and printf()

Add two integer numbers and print the input numbers and result.

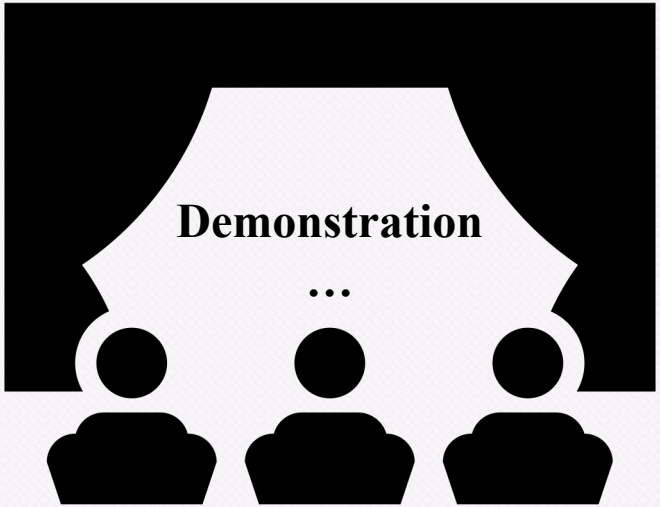
Solution

```
#include <stdio.h>
int main()
{
  int a,b,c;
  printf("\nThe first number is ");
  scanf("%d",&a);
  printf("\nThe second number is ");
  scanf("%d",&b);
  c=a+b;
  printf("The answer is %d \n",c);
  return 0;
}
```

Output

```
The first number is 5
The second number is 9
The answer is 14
```

Let’s look at the code in IDE together!



Key Takeaways

You are now able to:

- ✓ Define input and output.
- ✓ Apply Non-formatted input and output.
- ✓ Apply Formatted input and output.

References

- Dey, P., & Ghosh, M. (2013). *Computer fundamentals and programming in C*.
- *C - Input and output*. (n.d.). Online Tutorials, Courses, and eBooks Library | Tutorialspoint. https://www.tutorialspoint.com/cprogramming/c_input_output.htm

Thank you !

Questions or Feedbacks?

Contact Me via:

leangsiv.han@cadt.edu.kh

@leangsiv

leangsivhan