MANAGING INPUT AND OUTPUT OPERATIONS

Staff Incharge: S. Sasirekha
MANAGING INPUT AND OUTPUT OPERATIONS

- One of the essential operations performed in a C language programs is to provide input values to the program and output the data produced by the program to a standard output device.

- We can assign values to variable through assignment statements such as \( x = 5 \) \( a = 0 \); and so on.

- Another method is to use the Input then `scanf` which can be used to read data from a keyboard.

- For outputting results we have used extensively the function `printf` which sends results out to a terminal.

- There exists several functions in ‘C’ language that can carry out input output operations.

- These functions are collectively known as standard Input/Output Library.

- Each program that uses standard input / output function must contain the statement.

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

  at the beginning
1. Formatted Input For Scanf:

- The formatted input refers to input data that has been arranged in a particular format.

- Input values are generally taken by using the scanf function.

**The scanf function has the general form.**

```
Scanf ("control string", arg1, arg2, arg3 ..........argn);
```

- The format field is specified by the control string
- The arguments arg1, arg2, .................argn specifies the address of location where address is to be stored.

- The control string specifies the field format which includes
  - Format specifications
  - optional number specifying field width
  - The conversion character % and also blanks, tabs and newlines.

- The Blanks tabs and newlines are ignored by compiler.

- The conversion character % is followed by the type of data that is to be assigned to variable of the assignment.
The general format for reading a integer number is

\[ \% x d \]

- Here percent sign (\%) denotes that a specifier for conversion follows.
- \( x \) is an integer number which specifies the width of the field of the number that is being read.
- The data type character \( d \) indicates that the number should be read in integer mode.

**Example:**

\[
\text{scanf}(\"\%3d \%4d\", \&\text{sum1}, \&\text{sum2});
\]

- If the values input are 175 and 1342 here value 175 is assigned to \text{sum1} and 1342 to \text{sum2}.
- Suppose the input data was follows 1342 and 175. The number 134 will be assigned to \text{sum1} and \text{sum2} has the value 2.
- If floating point numbers are assigned then the decimal or fractional part is skipped by the computer.
- To read the long integer data type we can use conversion specifier \% ld & \% hd for short integer.
Input specifications for real number:

- Field specifications are not to be use while representing a real number therefore real numbers are specified in a straight forward manner using %f specifier.

The general format of specifying a real number input is

\texttt{Scanf \( \% f \) \", \&variable\);}

\textbf{Example:}

\texttt{Scanf \("\%f \%f \%f\", \&a, \&b, \&c\);}

With the input data

321.76, 4.321, 678 The values 321.76 is assigned to a, 4.321 to b & 678 to C.

If the number input is a double data type then the format specifier should be \texttt{\%lf} instead of \texttt{\%f}. 
Input specifications for a character:

- Single character or strings can be input by using the character specifiers.
  **The general format is**
  \[% xc or %xs\]

- Where C and S represents character and string respectively and x represents the field width.

- The address operator need not be specified while we input strings.

**Example:**
- `scanf("%C %15C", &ch, nname):`

Here suppose the input given is

- a, Robert

  - a is assigned to ch
  - name will be assigned to Robert.
Points To Remember while using scanf

- All function arguments, except the control string, must be pointers to variables.

- Format specifications contained in the control string should match the arguments in order.

- Input data items must be separated by spaces and must match the variables receiving the input in the same order.

- The reading will be terminated, when scanf encounters an ‘invalid mismatch’ of data or a character that is not valid for the value being read.

- When searching for a value, scanf ignores line boundaries and simply looks for the next appropriate character.

- Any unread data items in a line will be considered as a part of the data input line to the next scanf call.

- When the field width specifier \( w \) is used, it should be large enough to contain the input data size.
Program

main()
{
    int a, b, c, x, y, z;
    int p, q, r;
    printf("Enter three integer numbers \n");
    scanf("%d %*d %d", &a, &b, &c);
    printf("%d %d %d \n \n", a, b, c);
    printf("Enter two 4-digit numbers \n");
    scanf("%2d %4d ", &x, &y);
    printf("%d %d \n \n", p, q, r);
    printf("Enter two integer numbers \n");
    scanf("%d %d", &a, &x);
    printf("%d %d \n \n", a, x);
    printf("Enter a nine digit numbers \n");
    scanf("%3d %4d %3d", &p, &q, &r);
    printf("%d %d %d \n \n", p, q, r);
    printf("Enter two three digit numbers \n");
    scanf("%d %d", &x, &y);
    printf("%d %d \n \n", x, y);
}
2. Formatted output for printf:

- `printf();`

  The most simple output statement can be produced in C’ Language by using printf statement.

- It allows you to display information required to the user and also prints the variables we can also format the output and provide text labels.

- The simple statement such as

  `Printf (“Enter 2 numbers”);`

- Prompts the message enclosed in the quotation to be displayed.
A simple program to illustrate the use of printf statement:-

```c
#include <stdio.h>
main () {
    printf ("Hello!");
    printf ("Welcome to the world of Engineering!");
}
```

**Output:**

Hello! Welcome to the world of Engineering.

- Both the messages appear in the output as if a single statement.
- If you wish to print the second message to the beginning of next line, a new line character must be placed inside the quotation marks.

**Example :**

```c
printf ("Hello!\n");
OR
printf ("\n Welcome to the world of Engineering");
```
Conversion Strings and Specifiers:

- The printf( ) function is quite flexible.

- It allows a variable number of arguments, labels and sophisticated formatting of output.

- The general form of the printf( ) function is

  Syntax
  \[
  \text{Printf ("conversion string", variable list);
  }\]

- Conversion string consists of three types of items:

  1. Characters that will be printed on the screen as they appear.

  2. Format specifications that define the output format for display of each item.

  3. Escape sequence characters such as \n, \t and \b

- The variable includes all the variable to be printed in order they are to be printed.
Output of Integer Numbers

- The format specification for printing an integer number is `%wd`

Output of Real Numbers

- The output of real numbers may be displayed in decimal notation using the following format specification:
  `%w.p f`

- The integer \( w \) indicates the minimum number of positions that are to be used for the display of the value.

- The integer \( p \) indicates the number of digits to be displayed after the decimal point.

- We can also display real numbers in exponential notation by using the specification
  `%w.p e`
Printing of Single Character

- A single character can be displayed in a desired position using the format

- The character will be displayed right-justified in the field of $w$ columns. We can make the display left-justified by placing a minus sign before the integer $w$.

- The default value for $w$ is 1.

Printing of Strings

- The format specification for outputting strings is of the form

  \%
  \%w
  \%w.ps
Specifier Meaning

%c – Print a character
%d – Print a Integer
%i – Print a Integer
%e – Print float value in exponential form.
%f – Print float value
%g – Print using %e or %f whichever is smaller
%o – Print actual value
%s – Print a string
%x – Print a hexadecimal integer (Unsigned) using lower case a – F
%X – Print a hexadecimal integer (Unsigned) using upper case A – F
%a – Print a unsigned integer.
%p – Print a pointer value
%hx – hex short
%lo – octal long
3. a. Reading A Character

• Reading a single character can be done by using the function `getchar`.

• The `getchar` takes the following form:
  `variable_name = getchar();`

Example:

• `char name;`
• `name=getchar();`
/* Reading a character from terminal */
#include<stdio.h>
main() {
  char ans;
  printf("Would you like to know my name? \n");
  printf("Type Y for yes and N for no\n");
  ans=getchar();
  if(ans == 'Y' || ans == 'y')
    printf("\n\n My name is India \n");
  else
    printf("\n\n You are good for nothing \n");
}

OUTPUT
Would you like to know my name?
Type Y for yes and N for no: Y
My name is India
Would you like to know my name?
Type Y for yes and N for no: n
You are good for nothing
3.b. Writing A Character

- Like `getchar`, there is an analogous function `putchar` for writing characters one at a time to the terminal.

- It takes the form as shown below:
  ```c
  putchar(variable_name);
  ```

**Example**

```c
  answer=’y’;
  putchar(answer);
```

- will display the character y on the screen.

The statement

```c
  putchar(‘\n’);
```

- Would cause the cursor on the screen to move to the beginning of the next line.
Program

```c
#include<stdio.h>
#include<ctype.h>
main() {
    char alphabet;
    printf("Enter an alphabet");
    putchar(\n");
    alphabet = getchar();
    if(islower(alphabet))
        putchar(toupper(alphabet));
    else
        putchar(tolower(alphabet));
}
```

OUTPUT
Enter An alphabet
a
A
Enter An alphabet
Q
q
Enter An alphabet
z
Z
4. String input and output:

- The **gets** function relieves the string from standard input device.

- **puts** outputs the string to the standard output device.

- A string is an array or set of characters.

- The function gets accepts the name of the string as a parameter, and fills the string with characters that are input from the keyboard till newline character is encountered.

- All the end function gets appends a null terminator as must be done to any string and returns.

- The puts function displays the contents stored in its parameter on the standard screen.

The standard form of the gets function is

**gets (str)**
- Here str is a string variable.

The standard form for the puts character is

**puts (str)**
- Where str is a string variable.
Example program (Involving both gets and puts)

```c
#include <stdio.h>

Void main ( )
{
  char s[80];
  printf(“Type a string less than 80 characters:”);
  gets (s);
  printf(“The string types is:”);
  puts(s);
}
```