SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

COURSE NAME : 23ITT101- PROBLEM SOLVING & C PROGRAMMING

I YEAR /I SEMESTER

Unit II – C PROGRAMMING BASICS

Topic : Managing Input and Output operations

Topics to be covered

- Input / Output in C
- Streams
- Formatted Input/output function
- Unformatted Input/Output function

Input/Output in C

- C has no built-in statements for input or output.

- A library of functions is supplied to perform these operations. The I/O library functions are listed the “header” file <stdio.h>.

Streams

- All input and output is performed with streams.
- A "stream" is a sequence of characters organized into lines.
- Each line consists of zero or more characters and ends with the "newline" character.
- ANSI C standards specify that the system must support lines that are at least 254 characters in length (including the newline character).

Types of Streams in C

- Standard input stream is called "stdin" and is normally connected to the keyboard
- Standard output stream is called "stdout" and is normally connected to the display screen.
- Standard error stream is called "stderr" and is also normally connected to the screen.

Formatted I/O Functions

1. printf - Formatted output
2. scanf - Formatted input
3. fprintf - Formatted output to a file
4. fscanf - Formatted input from a file
5. sprintf - Formatted output to a string
6. sscanf - Formatted input from a string

Unformatted I/O Functions

1. getchar - Read a single character from standard input
2. putchar - Write a single character to standard output
3. gets (deprecated) - Read a string from standard input
4. puts - Write a string to standard output
5. fgetc - Read a single character from a file
6. fputc - Write a single character to a file
7. fgets - Read a string from a file
8. fputs - Write a string to a file
9. fread - Read blocks of data from a file
10. fwrite - Write blocks of data to a file

Formatted Input with scanf

- This function provides for formatted input from the keyboard. The syntax is:

```
scanf (”control string” , &var1, &var2, …) ;
```

- Control string contains field specification which direct the interpretation of input data. It may include:
    - Field(or format)specifications, consisting of conversion character %, a data type character, and an optional number, specifying the field width. Blanks, tabs, or newlines.
- The & in front of each variable name tells the system where to store the value that is input. It provides the address for the variable.
- Example:

```
float a;  int b;
scanf (“%f%d”, &a, &b);
```

Formatted Input with scanf

Inputting integer numbers

The field specification for reading an integer number is

%wd

Example:

scanf(“%2d %5d”,&num1, &num2);

An input field may be skipped by specifying * in the place of field width.

Example:

scanf(“%d %*d %d”,&a, &b);

Formatted Input with scanf

Inputting real numbers

The field width of real numbers is not to be specified and therefore scanf reads real numbers using simple specification %f for both the notations, namely, decimal point notation and exponential notation.

Example:

```
scanf(“%f %f %f”, &x, &y, &z);
```

If the number to be read is of double type, then the specification should be %lf.

Formatted Input with scanf

Inputting Character Strings

Following are the specifications for reading character strings:

%ws or %wc

Some versions of scanf support the following conversion specifications for strings:

%[characters] and %[^characters]

%[characters] - only the characters specified within the brackets are permissible in the input string.

%[^characters] - does exactly the reverse

Mixed type : scanf("%d %c %f %s",&count, &code, &ratio, &name);

Formatted Output with printf

- This function provides for formatted output to the screen. The syntax is:

  printf (“control string”, var1, var2, …) ;

<u>Control string consists of three types of items:</u>

1. Characters that will be printed on the screen as they appear.
2. Format specifications that define the outAput format for display of each item.
3. Escape sequence characters such as \n, \t and \b

Example:

```
float a ;  int b ;
scanf ( “%f%d”, &a, &b ) ;
printf ( “You entered %f and %d \n”, a, b ) ;
```

Formatted Output with printf

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.pf

Integer w indicates the minimum number of positions that are to be used for the display of the value and the integer p indicates the number of digits to be displayed after the decimal point.

%w.pe

display real numbers in exponential notation by using the specification

Formatted Output with printf

Printing of Single Character

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

%wc

The character will be displayed right-justified in the field of w columns. Placing a minus sign before the integer w make the display left-justified. The default value for w is 1.

Printing of Strings

The format specification for outputting strings is of the form

%w.ps

Mixed type: printf(“%d %f %s %c”,a, b, c, d);

Format Codes

Code	Meaning
%c	Read/display a single character
%d	Read/display a decimal integer
%e	Read/display a floating point value in exponential notation
%f	Read/display a floating point value
%g	Read/display a floating point value in either "e" or "f" format
%h	Read/display a short integer
%i	Read/display a decimal, hexadecimal, or octal integer
%o	Read/display an octal integer
%s	Read/display a string
%u	Read/display an unsigned decimal integer
%x	Read/display a hexa decimal integer
%[..]	Read/display a string of word(s)

Input/Output in C

getchar() and putchar():

getchar(): Reads a single character from stdin

putchar(): Writes a single character to stdout.

This function provides for getting exactly one character from the keyboard.

Example:

```
        char ch;
        ch = getchar ( ) ;            /* Accept a character from keyboard*/
        putchar(ch);                  /* display a character on the screen */
```

Input/Output in C

gets() and puts():

gets : Reads a line of text (string) from stdin until a newline (\n) or EOF is encountered.

puts: Writes a string to stdout (standard output) and automatically adds a newline character (\n) at the end.

Example:

```
char str[100];
gets(str);                // Accepts a string from keyboard
puts(str);                //Display a string on the screen
```

Input/Output in C

getc(*file) and putc(char, *file) ;

getc (*file) : similar to getchar( ) except the input can be from the keyboard or a file.

putc (char, *file) : similar to putchar ( ) except the output can be to the screen or a file.

Example:

```
char ch;
ch = getc (stdin) ;       /* input from keyboard */
ch = getc (fileptr) ;     /* input from a file */
putc(ch, stdout) ;        /* output to the screen */
putc(ch, outfileptr) ;    /*output to a file */
```

Example scanf() and printf()

```
#include <stdio.h>

int main() {
    // Declare variables
    int num1, num2, sum;

    // Output prompt to the user
    printf("Enter two integers: ");

    // Input values from the user
    scanf("%d%d", &num1, &num2);

    // Perform addition
    sum = num1 + num2;

    // Output the result
    printf("The sum of %d and %d is %d.\n", num1, num2, sum);

    return 0;
}
```

```
Enter two integers: 5 10
The sum of 5 and 10 is 15.
```

Example getc(),putc(),gets(),puts()

```
#include <stdio.h>

int main() {
    char str[100];  // Array to store input string
    char ch;

    printf("Enter a string: ");
    gets(str);  //Using gets() to take a string input from the user
    printf("\nYou entered: ");
    puts(str);  // puts prints automatically adds a newline after the string

    printf("\nEnter a character: ");
    ch = getc(stdin);  // getc reads a character from stdin
    printf("You entered the character: ");
    putc(ch, stdout);  // putc writes the character to stdout
    printf("\n");

    return 0;
}
```

```
Enter a string: Hello, world!
You entered: Hello, world!

Enter a character: A
You entered the character: A
```

Example getchar(),putchar()

```
#include <stdio.h>

int main() {
    char ch;

    // Prompt the user to enter a character
    printf("Enter a character: ");

    // Using getchar() to read a single character from standard input
    ch = getchar();

    // Using putchar() to print the entered character to standard output
    printf("\nYou entered: ");
    putchar(ch);

    printf("\n");

    return 0;
}
```

```
Enter a character: A

You entered: A
```

Thank You