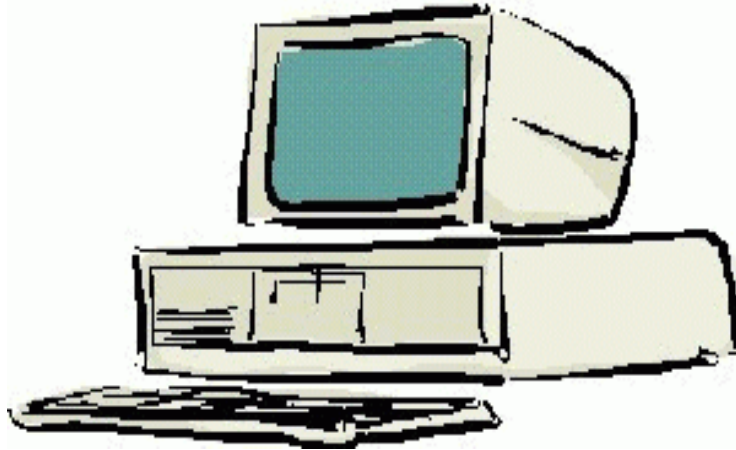
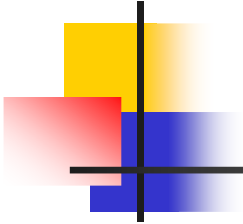


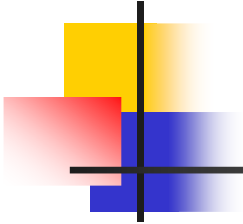
Programming in C



Session 2

Seema Sirpal
Delhi University Computer Centre

Input & Output



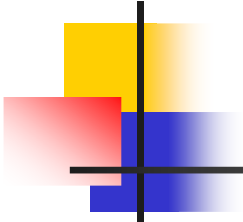
Input and Output form an important part of any program.

To do anything useful your program needs to be able to accept input data and report back your results.

In C, the standard library provides routines for input and output. The standard library has functions for i/o that handle input, output, and character and string manipulation.

Presently we consider, all the input functions described read from standard input and all the output functions described write to standard output. Standard input is usually the keyboard. Standard output is usually the monitor.

Formatted Output



The standard library function printf is used for formatted output.

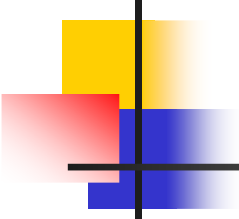
It takes as arguments a format string and an optional list of variables or literals to output.

The variables and literals are output according to the specifications in the format string.

Here is the prototype for printf.

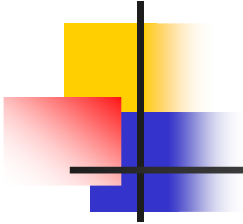
```
int printf(const char *format, arg1, arg2, arg3, .....);
```

Example



```
#include <stdio.h>
int main()
{
    int luckyNumber = 5;
    float radius = 2;
    char myName[15] = "John";
    char initial = 'J';
    printf("Hello World\n"); /* The format string contains only ordinary
                             characters. Ordinary characters are output unmodified.
                             A character string in C is of type "char *". */
    printf("My lucky number is %d\n", luckyNumber); /* The "%d" specifies
                                                     that an integer value will be output. */
    printf("My name is %s\n",myName); /* The %s specifies that a character
                                       string will be output. */
    printf("My first initial is %c\n",initial); /* The %c specifies that a
                                                character will be output. */
    printf("The area of a circle with radius %f is %f\n", radius, 3.14*radius*radius);
        /* %f specifies that a float will be output. */
    printf("Hello %s or should I say %c\n",myName,initial);
        /* Multiple arguments of different types may be output. */
    return(0);
}
```

Formatted Output



Here are the more common conversion specifiers.

Specifier	Argument Type
%d	int
%f	float or double
%e	float or double, output in scientific notation.
%c	character
%s	character string (char *)

Formatted Input



The standard library function scanf is used for formatted input.

It takes as its arguments a format string and a list of pointers to variables to store the input values.

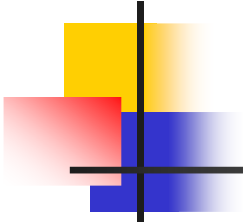
Similar to its use in printf, the format string can contain literals and conversion specifiers.

In C, to return a value from a function to a calling routine, a pointer to a variable is passed into the function. A pointer stores the memory address of another variable.

Here is the prototype of scanf and a program illustrating its use.

```
int scanf(const char *format, arg1, arg2, ....);
```

Formatted Input



Scanf returns an integer, either the number of values read in, or EOF if an end of file is reached.

EOF is a special termination character, specified in stdio.h, which designates the end of a file.

If no values are successfully read, scanf returns 0.

To use scanf in a program, the file `stdio.h` must be included.

Formatted

Input



```
#include <stdio.h>
int main()
{
    float radius;
    char yourName[15];
    int number;
    printf("Enter the radius of circle: ");
    scanf("%f",&radius); /* & is the "address of" operator.
        The address of radius is passed into scanf. Passing
        the address of a variable is equivalent to passing
        a pointer containing the address of the variable. */
    printf("A circle of radius %f has area
%f\n",radius,3.14*radius*radius);
    printf("Enter a number from 1 to 1000: ");
    scanf("%d",&number);
        /* The address of number is passed to scanf */
    printf("Your number is %d\n",number);
    printf("Enter your name: ");
    scanf("%s",yourName); /* yourName is a character array.
        yourName[0] specifies the first element of the array.
        &yourName[0] specifies the address of the first element
        of the array. In C, an array name by itself is shorthand
        for the address of the first element. So, yourName is
        equivalent to &yourName[0], which is what must be
        passed into scanf. This will be made clearer in the
        lesson covering arrays. */
    printf("Hello %s\n",yourName);
    return(0);
}
```


Other Useful Standard Library Functions

for Input and Output



getchar

getchar reads a single character from standard input. Its prototype is:

```
int getchar();
```

It returns int rather than char because the "end of file", EOF, character must be handled. EOF is an int and is too large to fit in a char variable.

A newline character in C is '\n'. This designates the end of a line.

putchar

putchar writes a single character to standard output. Its prototype is:

```
int putchar(int value);
```

Other Useful Standard Library Functions

for Input and Output



A simple program which echoes back everything that is typed in. Depending on which operating system you are using, EOF is entered by typing control-z or control-d. So, try running this code, type control-z or control-d to end execution

```
#include <stdio.h>

int main()
{
    int c;

    while ((c = getchar()) != EOF)
    {
        putchar(c);
    }

    return(0);
}
```

Other Useful Standard Library Functions for Input and Output

gets

gets reads a line of input into a character array. Its prototype is:

```
char *gets(char *buffer);
```

It returns a pointer to the character string if successful, or NULL if end of file is reached or if an error has occurred.

The string is also read into the character array specified as an argument. The character string will be terminated by a "\0", which is standard for C.

Other Useful Standard Library Functions for Input and Output

puts

puts writes a line of output to standard output. Its prototype is:

```
int puts(const char *buffer);
```

It terminates the line with a newline, '\n'. It will return EOF if an error occurred. It will return a positive number on success.

Here is the "echo" program implemented using the functions gets and puts. Type control-z or control-d to end execution.

Other Useful Standard Library Functions

for Input and Output

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

```
int main()
```

```
{
```

```
    char buffer[120]; /* Holds input and output strings */
```

```
    char *pt; /* A pointer to datatype char */
```

```
    int returnCode;
```

```
    while ((pt = gets(buffer)) != NULL)
```

```
    {
```

```
        returnCode = puts(buffer);
```

```
        if (returnCode == EOF)
```

```
        {
```

```
            printf("Error on output\n");
```

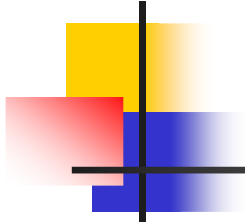
```
        }
```

```
    }
```

```
    return(0);
```

```
}
```

Practice



- 1) Write a program to read in the name and address of the user and echo back that information.
- 2) Write a program to prompt a user for a nickname, the year they were born as an integer, and their weight in pounds as a float. Calculate their age. Calculate their weight in kilograms. Write this information to standard output. Note there are 2.2046 pounds per kilogram.