ਜ਼ਰੂਰੀ ਸੀ ਪ੍ਰੋਗਰਾਮਿੰਗ
Essential C Programming
(Parrot Book)

Copyright ©
Written By: Bhupinder Singh, B.E., PEC Chandigarh,
President, EraSoft India, Moga, Cheema, Mobile: 94648-79699

ਭੂਮਿਕਾ

ਉਮਰ ਦੇ ਵੱਖੋ-ਵੱਖਰੇ ਪੜਾਵਾਂ ਤੇ ਇਨਸਾਨ ਦੇ ਵੱਖੋ-ਵੱਖਰੇ ਮਕਸਦ ਹੁੰਦੇ ਹਨ। ਪਰ ਮੈਂ ਸਮਝਦਾ ਹਾਂ ਕਿ ਇੱਕ ਕੰਪਿਊਟਰ ਦੀ ਸਿਖਿਆ ਲੈ ਰਹੇ ਵਿਦਿਆਰਥੀ ਦਾ ਇਹ ਮਕਸਦ ਜ਼ਰੂਰ ਹੋਣਾ ਚਾਹੀਦਾ ਹੈ ਕਿ ਡਿਗਰੀ ਦੌਰਾਨ ਉਹ ਕੁਝ ਅਜਿਹਾ ਸਿੱਖੇ ਜਿਸ ਨਾਲ ਉਹ ਕੋਈ ਲਾਭਕਾਰੀ ਸਾਫਟਵੇਅਰ ਬਣਾ ਸਕੇ, ਜੋ ਕਿ ਖਰੀਦਦਾਰ ਲਈ ਉਪਯੋਗੀ ਸਿੱਧ ਹੋਵੇ ਅਤੇ ਉਸਦਾ ਕੰਮ ਸੌਖਾਲਾ ਬਣਾ ਸਕੇ।

ਸੀ ਪ੍ਰੋਗਰਾਮਿੰਗ ਅਜਿਹੀ ਇੱਕ ਕੰਪਿਊਟਰ ਭਾਸ਼ਾ ਹੈ ਜੋ ਵਿਦਿਆਰਥੀ ਦੇ ਇਸ ਮਕਸਦ ਵਿੱਚ ਪਹਿਲਾ ਪੜਾਅ ਹੈ। ਜੇ ਮੇਰੀ ਇਹ ਕਿਤਾਬ ਪੜਕੇ ਵਿਦਿਆਰਥੀ ਵਿੱਚ ਸਾਫਟਵੇਅਰ ਬਣਾਉਣ ਦੀ ਰੁਚੀ ਪੈਦਾ ਹੁੰਦੀ ਹੈ ਅਤੇ ਉਸਦੀ ਪ੍ਰੋਗਰਾਮਿੰਗ ਕਲਪਨਾ ਉਡਾਣ ਭਰਣ ਲਗਦੀ ਹੈ, ਤਾਂ ਮੈਂ ਸਮਝਦਾ ਹਾਂ ਕਿ ਮੇਰਾ ਲਿਖਣਾ ਸਾਰਥਕ ਹੋਇਆ।

ਬਾਕੀ ਮੈਂ ਸਮਝਦਾ ਹਾਂ ਕਿ ਤਕਨੀਕ ਦਾ ਭਾਸ਼ਾ ਨਾਲ ਕੋਈ ਸਰੋਕਾਰ ਨਹੀਂ ਹੋਣਾ ਚਾਹੀਦਾ। ਸਾਡੇ ਦੇਸ਼ ਦੇ ਕਰੋੜਾਂ ਵਿਦਿਆਰਥੀਆਂ ਲਈ ਇਹ ਬਹੁਤ ਲਾਭਕਾਰੀ ਹੋਵੇਗਾ ਜੇ ਤਕਨੀਕੀ ਡਿਗਰੀਆਂ ਸਾਡੀ ਮਾਤ-ਭਾਸ਼ਾ ਵਿੱਚ ਦਿੱਤੀਆਂ ਜਾਣ। ਇਸ ਨੂੰ ਧਿਆਨ ਵਿੱਚ ਰਖਦੇ ਹੋਏ ਇਹ ਕਿਤਾਬ ਅੰਗਰੇਜ਼ੀ ਦੇ ਨਾਲ-ਨਾਲ ਪੰਜਾਬੀ ਵਿੱਚ ਵੀ ਹੈ। ਮੇਰਾ ਮੰਨਣਾ ਹੈ ਕਿ ਜਲਦੀ ਹੀ ਅਸੀਂ ਪੰਜਾਬੀ, ਹਿੰਦੀ ਅਤੇ ਸਾਡੀਆਂ ਹੋਰ ਖੇਤਰੀ ਭਾਸ਼ਾਵਾਂ ਵਿੱਚ ਵਿਕਸਿਤ ਕੀਤੇ ਕੰਪਾਇਲਰ ਵਰਤ ਰਹੇ ਹੋਵਾਂਗੇ।

ਭੁਪਿੰਦਰ ਸਿੰਘ

ਜ਼ਰੂਰੀ ਸੀ ਪ੍ਰੋਗਰਾਮਿੰਗ
Essential C Programming

1. ਮੁਢਲੀ ਜਾਨਕਾਰੀ (Introduction) -4

2. ਸੀ ਦੇ ਡੇਟਾ ਟਾਈਪ ਅਤੇ ਓਪਰੇਟਰ (Data Types and Operators of C) -7

3. ਕੰਟਰੋਲ ਸਟਰਕਚਰਸ (Control Structures) -9

4. ਐਰੇ (Array) -19

5. ਸਟਰਿੰਗਸ (Strings) -24

6. ਫੰਕਸ਼ਨਸ (Functions) -29

7. ਪੋਇੰਟਰਸ (Pointers) -33

8. ਸਟਰਕਚਰਸ (Structures) -36

9. ਫਾਈਲਾਂ (Files) -38

10. ਸਟੋਰੇਜ ਟਾਈਪਸ (Storage Types) -47

11. ਪਰੀਪਰੋਸੈਸਰ ਡਾਈਰੈਕਟਿਵਸ (Pre-Processor Directives) -49

ਜ਼ਰੂਰੀ ਸੀ ਪ੍ਰੋਗਰਾਮਿੰਗ
Essential C Programming

ਮੁਢਲੀ ਜਾਨਕਾਰੀ (Introduction)

C is a programming language of computer. Whatever you see in computer most of it is a magic of C. Word, Excel everything is made using C.

ਸੀ ਕੰਪਿਊਟਰ ਦੀ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾ ਹੈ। ਕੰਪਿਊਟਰ ਵਿਚ ਜੋ ਵੀ ਤੁਸੀਂ ਦੇਖਦੇ ਹੋ, ਜਿਆਦਾਤਰ ਸੀ ਦਾ ਹੀ ਕਮਾਲ ਹੈ। ਵਰਡ, ਐਕਸਲ, ਸਬ ਸੀ ਨਾਲ ਹੀ ਬਣਾਏ ਗਏ ਹਨ।

C was invented by Dennis Ritchie around 1973.

ਸੀ ਦੀ ਕਾਢ 1973 ਦੇ ਆਸ ਪਾਸ ਡੈਨਿਸ ਰਿਚੀ ਨੇ ਕੱਢੀ ਸੀ।

When we write C program it is called source code. We compile it and link with C libraries to get executable file, which has instructions that can be understood by the computer. If we run this file the computer works according to the code in this file.

ਜਦੋਂ ਅਸੀਂ ਪ੍ਰੋਗਰਾਮ ਲਿਖਦੇ ਹਾਂ ਤਾਂ ਉਸ ਨੂੰ ਸੋਰਸ ਕੋਡ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਸ ਨੂੰ ਕੰਪਾਇਲ ਕਰਕੇ ਸੀ ਦੀਆਂ ਲਾਈਬਰੇਰੀਆਂ ਨਾਲ ਲਿੰਕ ਕਰਕੇ ਜੋ ਫਾਈਲ ਬਣਦੀ ਹੈ, ਉਸ ਨੂੰ ਐਕਸੀਕਿਊਟੀਬਲ ਕਿਹਾ ਜਾਂਦਾ ਹੈ, ਜੋ ਕਿ ਕੰਪਿਊਟਰ ਦੇ ਸਮਝਣ ਯੋਗ ਹੁੰਦੀ ਹੈ। ਇਸ ਨੂੰ ਚਲਾਉਣ ਤੇ ਕੰਪਿਊਟਰ ਇਸ ਵਿਚ ਲਿਖੇ ਕੋਡ ਅਨੁਸਾਰ ਚਲਦਾ ਹੈ ।

There are many C compilers. We will use Turbo C to demonstrate all the programs.

ਸੀ ਦੇ ਕਈ ਕੰਪਾਈਲਰ ਹਨ। ਅਸੀਂ ਟਰਬੋ ਸੀ ਕੰਪਾਈਲਰ ਦੁਆਰਾ ਸਾਰੇ ਪ੍ਰੋਗਰਾਮ ਦਰਸ਼ਾਵਾਂਗੇ।

ਸੀ ਦੇ ਪਰੋਗਰਾਮ ਦਾ ਸਟਰਕਚਰ (Structure of a C program)

In C program first of all we tell about those files that we need. Then we write our statements in void main(). The first work is to keep space in memory for our calculations. It is called variable declaration.

ਸੀ ਪ੍ਰੋਗਰਾਮ ਵਿਚ ਸਭ ਤੋਂ ਪਹਿਲਾਂ ਅਸੀਂ ਓਹਨਾਂ ਫਾਈਲਾਂ ਬਾਰੇ ਦਸਦੇ ਹਾਂ ਜੋ ਸਾਨੂੰ ਚਾਹੀਦੀਆਂ ਹਨ। ਫਿਰ ਅਸੀਂ void main() ਵਿਚ ਆਪਨੀਆਂ ਸਟੇਟਮੈਂਟਾਂ ਲਿਖਦੇ ਹਾਂ। ਪਹਿਲਾ ਕੰਮ ਹੁੰਦਾ ਹੈ ਕਿ ਸਾਡੀ ਗਿਣਤੀ-ਮਿਣਤੀ ਲਈ ਮੈਮੋਰੀ ਵਿਚ ਜਗਾ ਬਣਾਈ ਜਾਵੇ। ਇਸ ਨੂੰ ਵੇਰੀਏਬਲ ਡੈਕਲਰੇਸ਼ਨ ਕਿਹਾ ਜਾਂਦਾ ਹੈ।

```
int a;
float b=20, c;
char ch;
```

Here from int a, computer comes to know that it has to keep a space in memory whose name will be a. In this space only int, meaning numbers without decimal point can be kept. Similarly float means that we can keep numbers with decimal points also. By char we mean only one character. In it, apart from 'A', 'B', ……, '1', '2', …… or any other special character like '@', '[', '%' can be kept.

ਇਥੇ int a ਤੋਂ ਕੰਪਿਊਟਰ ਨੂੰ ਪਤਾ ਲਗਦਾ ਹੈ ਕਿ ਉਸਨੇ ਮੈਮੋਰੀ ਵਿਚ ਜਗਾ ਬਨਾਉਣੀ ਹੈ ਜਿਸਦਾ ਨਾਮ a ਰੱਖਣਾ ਹੈ। ਇਸ ਜਗਾ ਵਿਚ int ਜਾਨਿ ਕਿ ਬਿਨਾ ਦਸ਼ਮਲਵ (ਬਿੰਦੂ) ਦੇ ਸੰਖਿਆ (integer) ਹੀ ਰੱਖੀ ਜਾ ਸਕਦੀ ਹੈ। ਇਸ ਤਰਾਂ float ਦਾ ਮਤਲਬ ਹੈ ਕਿ ਸੰਖਿਆ ਦਸ਼ਮਲਵ ਵਾਲੀ ਹੋ ਸਕਦੀ ਹੈ। char ਦਾ ਮਤਲਬ ਹੈ ਕੋਈ ਇਕ ਅਖਰ। ਇਸ ਵਿਚ 'A', 'B', ….. ਤੋਂ ਇਲਾਵਾ '1', '2', …… ਜਾਂ ਕੋਈ ਵੀ ਹੋਰ ਅਖਰ ਜਿਵੇਂ '@', ']', '%', ਆਦਿ ਕੁਝ ਵੀ ਆ ਸਕਦਾ ਹੈ।

Next when we have to show something on screen we have to use printf().

ਫਿਰ ਅਸੀਂ ਜਦ ਸਕਰੀਨ ਤੇ ਕੁਝ ਦਿਖਾਉਣਾ ਹੋਵੇ ਤਾਂ printf() ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ।

printf("Hello World");

The above statement will print Hello World on screen.

ਉਪਰ ਦਿਤੀ ਸਟੇਟਮੈਂਟ ਨਾਲ ਕੰਪਿਊਟਰ ਸਕਰੀਨ ਤੇ Hello World ਦਿਖਾਵੇਗਾ।

If we have to get input (any number from keyboard) from the user, we have to use scanf().

ਜੇ ਅਸੀਂ ਯੂਸਰ (ਕੰਪਿਊਟਰ ਚਲਾਉਣ ਵਾਲਾ) ਤੋਂ ਕੋਈ ਇਨਪੁਟ (ਕੀਬੋਰਡ ਤੋਂ ਕੋਈ ਸ਼ਬਦ ਜਾਂ ਸੰਖਿਆ) ਲੈਣੀ ਹੋਵੇ ਤਾਂ ਅਸੀਂ scanf() ਵਰਤਦੇ ਹਾਂ।

scanf("%d %f", &a, &c);

By the above statement, the first number that user will enter will go into a and the second into b. %d is for telling that the number in int and %f for float. Similarly %c is for char.

ਉਪਰੋਕਤ ਸਟੇਟਮੈਂਟ ਨਾਲ ਯੂਸਰ ਜੋ ਪਹਿਲੀ ਸੰਖਿਆ ਪਾਵੇਗਾ ਉਹ a ਵਿਚ ਜਾਵੇਗੀ ਅਤੇ ਦੂਸਰੀ c ਵਿਚ। %d int ਲਈ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ ਅਤੇ %f float ਲਈ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ, ਕਿਉਂਕਿ a int ਹੈ ਅਤੇ c float ਹੈ। ਇਸੇ ਤਰਾਂ %c char ਲਈ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ।

Whenever we have to do some calculation we do as shown below.

ਜਦ ਅਸੀਂ ਕੋਈ ਗਿਣਤੀ-ਮਿਣਤੀ ਕਰਨੀ ਹੋਵੇ ਤਾਂ ਹੇਠ ਲਿਖੇ ਅਨੁਸਾਰ ਕਰ ਸਕਦੇ ਹੋ।

b=a*c-2;

Here the expression on the right hand side of = will be solved and the answer will go (aqssigned) into b. Remember that on right hand side of =, there can be only one variable,as there is b in the above statement.

ਇਥੇ b ਵਿਚ, = ਦੇ ਸੱਜੇ ਪਾਸੇ ਵਾਲੀ ਅਕਸਪ੍ਰੈਸ਼ਨ ਹੱਲ ਹੋ ਕਿ ਜੋ ਉਤਰ ਆਵੇਗਾ, ਉਹ ਪਵੇਗਾ (ਅਸਾਈਨ ਹੋਵੇਗਾ) । ਯਾਦ ਰੱਖੋ ਕਿ ਖੱਬੇ ਪਾਸੇ ਇਕ ਹੀ ਵੇਰੀਏਬਲ ਹੋ ਸਕਦਾ ਹੈ, ਜਿਵੇਂ ਕਿ ਉਪਰ ਸਟੇਟਮੈਂਟ ਵਿਚ b ਹੈ।

ਸੀ ਦਾ ਪਹਿਲਾ ਪਰੋਗਰਾਮ (First C Program)

You start Turbo C. Use ALT+F, N to create a new file and type the following program in it.

ਤੁਸੀਂ ਟਰਬੋ ਸੀ ਚਲਾਓ। ALT+F, N ਤੋਂ ਨਵੀਂ ਫਾਈਲ ਖੋਲੋ ਅਤੇ ਉਸ ਵਿਚ ਹੇਠ ਲਿਖੇ ਅਨੁਸਾਰ ਪ੍ਰੋਗਰਾਮ ਟਾਈਪ ਕਰੋ।

```
#include<stdio.h>
#include<conio.h>
void main()
{
     int a,b,c;

     clrscr();
     printf("Enter two numbers ");
     scanf("%d %d",&a,&b);

     c=a+b;

     printf("Total is %d",c);

     getch();
}
```

Whenever we have to write a C program, the first thing is to include header files.

ਜਦੋਂ ਵੀ ਅਸੀਂ ਸੀ ਦਾ ਪ੍ਰੋਗਰਾਮ ਲਿਖਣਾ ਹੈ ਤਾਂ ਸਭ ਤੋਂ ਪਹਿਲਾਂ ਹੈਡਰ ਫਾਈਲਾਂ ਇਨਕਲੂਡ ਕਰਨੀਆਂ ਹਨ।

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

We have to write our program with void main(). Start with { and declare your variables first.

ਅਸੀਂ void main() ਨਾਲ ਪ੍ਰੋਗਰਾਮ ਲਿਖਣਾ ਹੈ। { ਸ਼ੁਰੂ ਕਰਕੇ ਸਭ ਤੋਂ ਪਹਿਲਾਂ ਆਪਨੇ ਵੇਰੀਏਬਲ ਲਿਖੋ।

```
int a,b,c;
```

We declare three variables a, b, c. clrscr() is used to to clear the screen. Then using printf() we print on screen, that enter two numbers. Using scanf() we get numbers from keyboard, which go into a and b. %d is for int. Then we add a and b and put the total in c and at the end we print c on screen. getch() is used as wait, so that, after seeing the value of c on screen, we should come to blue screen of editor only after pressing some key.

ਇਥੇ ਅਸੀਂ ਤਿਨ ਵੇਰੀਏਬਲ a, b, c ਡਿਕਲੇਅਰ ਕਰਦੇ ਹਾਂ। clrscr() ਸਕਰੀਨ ਸਾਫ ਕਰਨ ਲਈ ਹੈ। ਫਿਰ ਅਸੀਂ printf() ਨਾਲ ਸਕਰੀਨ ਤੇ ਪਰਿੰਟ ਕਰਦੇ ਹਾਂ ਕਿ ਦੋ ਨੰਬਰ ਭਰੋ। scanf() ਨਾਲ ਅਸੀਂ ਇਹ ਨੰਬਰ ਕੀਬੋਰਡ ਤੋਂ ਲੈਂਦੇ ਹਾਂ ਜੋ ਕਿ a ਅਤੇ b ਵਿਚ ਜਾਂਦੇ ਹਨ। %d ਦਾ ਮਤਲਬ ਹੈ ਕਿ ਨੰਬਰ int ਹਨ। ਫਿਰ ਅਸੀਂ a ਅਤੇ b ਦਾ ਜੋੜ c ਵਿਚ ਪਾਉਂਦੇ ਹਾਂ ਅਤੇ ਅਖੀਰ ਵਿਚ c ਨੂੰ ਪਰਿੰਟ ਕਰਦੇ ਹਾਂ। getch() ਨੂੰ ਵੇਟ ਦੇ ਤੌਰ ਤੇ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ, ਤਾਂਕਿ c ਦੀ ਕੀਮਤ ਦੇਖਣ ਤੋਂ ਬਾਅਦ ਕੋਈ ਕੀ ਦੱਬਣ ਤੇ ਹੀ ਅਸੀਂ ਵਾਪਿਸ ਐਡੀਟਰ ਦੀ ਨੀਲੀ ਸਕਰੀਨ ਤੇ ਆਈਏ।

ਸੀ ਦੇ ਡੇਟਾ ਟਾਈਪ ਅਤੇ ਓਪਰੇਟਰ (Data Types and Operators of C)

When we declare variables, we have to tell its type, like int, float, char etc. We call it data-types. C has the below written data-types.
ਜਦੋਂ ਅਸੀਂ ਵੇਰੀਏਬਲ ਡਿਕਲੇਅਰ ਕਰਦੇ ਹਾਂ ਤਾਂ ਉਸਦੀ ਟਾਈਪ ਦਸਦੇ ਹਾਂ, ਜਿਵੇਂ int, float, char ਆਦਿ। ਇਸ ਨੂੰ ਡੇਟਾ-ਟਾਈਪ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਸੀ ਵਿਚ ਹੇਠ ਲਿਖੇ ਡੇਟਾ-ਟਾਈਪ ਹਨ:

ਡੇਟਾ-ਟਾਈਪ (Data-Type)	**ਮੈਮੋਰੀ (Memory)**	**ਰੇਂਜ (Range)**
int	2 ਬਾਈਟ (Byte)	-32768 ਤੋਂ 32767
float	4 ਬਾਈਟ (Byte)	
long	4 ਬਾਈਟ (Byte)	$-2x10^9$ ਤੋਂ $2x10^9$
double	8 ਬਾਈਟ (Byte)	
char	1 ਬਾਈਟ (Byte)	-128 ਤੋਂ 127

In int and long, we can only keep numbers without decimal points. In float and double we can keep numbers with deimal. In char we can keep only one character. The range of int is less than that of long. Similarly the range of float is less than that of double.

int ਅਤੇ long ਵਿਚ ਬਿਨਾਂ ਦਸ਼ਮਲਵ ਵਾਲੀਆਂ ਸੰਖਿਆਵਾਂ ਹੀ ਰੱਖ ਸਕਦੇ ਹਾਂ। float ਅਤੇ double ਵਿਚ ਦਸ਼ਮਲਵ ਵਾਲੀਆਂ ਸੰਖਿਆਵਾਂ ਰਖਦੇ ਹਾਂ। char ਵਿਚ ਸਿਰਫ ਇਕ ਅਖਰ ਰਖਿਆ ਜਾਂਦਾ ਹੈ। int ਦੀ ਲਿਮਿਟ long ਨਾਲੋਂ ਘੱਟ ਹੁੰਦੀ ਹੈ। ਇਸੇ ਤਰਾਂ float ਦੀ ਲਿਮਿਟ double ਨਾਲੋਂ ਘੱਟ ਹੁੰਦੀ ਹੈ।

ਪਰਿੰਟ ਅਤੇ ਸਕੈਨ ਦੇ ਤਰੀਕੇ

ਟਾਈਪ (Type)	**ਵਰਤੀ ਜਾਂਦੀ ਨੋਟੇਸ਼ਨ (Used Notation)**
Int	%d, %i
Float	%f
Long	%ld
Double	%lf
Char	%c
String (char[])	%s

ਸੀ ਦੇ ਓਪਰੇਟਰ (C Operators)

To make a C program we can use the below written operators.

ਸੀ ਵਿਚ ਪ੍ਰੋਗਰਾਮ ਬਨਾਉਣ ਲਈ ਅਸੀਂ ਹੇਠ ਲਿਖੇ ਓਪਰੇਟਰ ਵਰਤ ਸਕਦੇ ਹਾਂ।

ਅਪਰੇਟਰ (Operator)	**ਕੰਮ (Purpose)**
ਮੈਥੇਮੈਟੀਕਲ (ਗਣਿਤ ਸਬੰਧਿਤ) (Mathematical)	
+	ਜੋੜ ਲਈ (Addition)
-	ਘਟਾਓ ਲਈ (Subtraction)
*	ਗੁਣਾ ਲਈ (Multiplication)
/	ਭਾਗ ਲਈ (Division)

<table>
<tr><td>%</td><td>ਭਾਗ ਕਰਨ ਤੇ ਬਾਕੀ (Remainder) ਲਈ</td></tr>
<tr><td>=</td><td>ਕੀਮਤ ਪਾਉਣ ਲਈ (Assignment)</td></tr>
<tr><td>ਰਿਲੇਸ਼ਨਲ (ਸਬੰਧਿਤ) (Relational)</td><td></td></tr>
<tr><td><</td><td>ਤੋਂ ਛੋਟਾ (Less than)</td></tr>
<tr><td><=</td><td>ਤੋਂ ਛੋਟਾ ਜਾਂ ਬਰਾਬਰ (Less than equal to)</td></tr>
<tr><td>></td><td>ਤੋਂ ਵੱਡਾ (Greater than)</td></tr>
<tr><td>>=</td><td>ਤੋਂ ਵੱਡਾ ਜਾਂ ਬਰਾਬਰ (Greater than equal to)</td></tr>
<tr><td>==</td><td>ਬਰਾਬਰ (Equality)</td></tr>
<tr><td>!=</td><td>ਨਹੀਂ ਬਰਾਬਰ (Not equal to)</td></tr>
<tr><td>ਲੌਜੀਕਲ (ਤਰਕ ਸਬੰਧਿਤ) (Logical)</td><td></td></tr>
<tr><td>&&</td><td>ਅਤੇ (And)</td></tr>
<tr><td>||</td><td>ਜਾਂ (Or)</td></tr>
<tr><td>!</td><td>ਨਹੀਂ (Not)</td></tr>
<tr><td>ਕੁਝ ਹੋਰ ਔਪਰੇਟਰ</td><td></td></tr>
<tr><td>++</td><td>ਇਕ ਵਧਾਓ (Increment one)</td></tr>
<tr><td>--</td><td>ਇਕ ਘਟਾਓ (Decrement one)</td></tr>
</table>

ਕੰਟਰੋਲ ਸਟਰਕਚਰਸ (Control Structures)

ਇਫ ਐਲਸ (if – else)

We use if-else to change the control, based on a result.

if – else ਦੀ ਵਰਤੋਂ ਕਿਸੇ ਨਤੀਜੇ ਦੇ ਅਧਾਰ ਤੇ ਕੰਟਰੋਲ ਬਦਲਣ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ।

```
if(age >= 18)
{
     printf("You can get license.");
}
else
{
     printf("Your age is less.");
}
```

As you see, that if the value in age variable is greater thanor equal to 18 we see on the screen that You can get license. Else it prints that you cannot get license.

ਜਿਵੇਂ ਤੁਸੀਂ ਦੇਖਿਆ ਹੈ ਕਿ ਜੇ age ਵੇਰੀਏਬਲ ਦੀ ਕੀਮਤ 18 ਜਾਂ ਇਸ ਤੋਂ ਵੱਧ ਹੈ ਤਾਂ ਸਕਰੀਨ ਤੇ ਦਿਖੇਗਾ ਕਿ ਤੁਹਾਨੂੰ ਲਾਇਸੰਸ ਮਿਲ ਸਕਦਾ ਹੈ। ਨਹੀਂ ਤਾਂ ਪਰਿੰਟ ਹੋਵੇਗਾ ਕਿ ਤੁਹਾਡੀ ਉਮਰ ਘੱਟ ਹੈ।

The meanings of if-else are as below.

if else ਦੇ ਮਤਲਬ ਹੇਠ ਲਿਖੇ ਅਨੁਸਾਰ ਹਨ।

if - ਜੇ
else - ਨਹੀਂ ਤਾਂ
else if - ਨਹੀਂ ਤਾਂ ਜੇ

ਡਿਵੀਜ਼ਨ ਲੱਭਣ ਦਾ ਪ੍ਰੋਗਰਾਮ (Program to Find Division)

We will now make a program which will tell that in which division the student has passed.

ਅਸੀਂ ਹੁਣ ਇਕ ਪ੍ਰੋਗਰਾਮ ਬਣਾਉਂਦੇ ਹਾਂ ਜੋ ਦੱਸੇਗਾ ਕਿ ਵਿਦਿਆਰਥੀ ਕਿਹੜੇ ਦਰਜੇ ਵਿਚ ਪਾਸ ਹੋਇਆ ਹੈ।

```
#include<stdio.h>
#include<conio.h>
void main()
{
      int marks;

      clrscr();
      printf("Enter marks of student: ");
      scanf("%d",&marks);

      if(marks >= 60)
```

```
    {
         printf("First Division");
    }
    else if(marks >= 50)
    {
         printf("Second Division");
    }
    else if( marks >= 33)
    {
         printf("Third Division");
    }
    else
    {
         printf("Fail");
    }

    getch();
}
```

First of all, we declare marks variable and get the input in it. Then we check that if the value in marks is greater than or equal to 60, we print First Division.

ਅਸੀਂ ਸਭ ਤੋਂ ਪਹਿਲਾਂ marks ਵੇਰੀਏਬਲ ਡਿਕਲੇਅਰ ਕਰਕੇ ਇਸ ਵਿਚ ਨੰਬਰ ਇਨਪੁਟ ਕਰਾਉਂਦੇ ਹਾਂ। ਫਿਰ ਅਸੀਂ if ਨਾਲ ਚੈਕ ਕਰਦੇ ਹਾਂ ਕਿ ਜੇ marks ਦੀ ਕੀਮਤ 60 ਜਾਂ ਇਸ ਤੋਂ ਵੱਧ ਹੈ ਤਾਂ ਅਸੀਂ ਫਰਸਟ ਡਿਵੀਜ਼ਨ ਪਰਿੰਟ ਕਰਦੇ ਹਾਂ।

If the value in marks is not greater than or equal to 60, then the next else checks, if marks are greater than or equal to 50. Then we print Second Division. Similary for 33 or more marks we print Third Division.

ਜੇ marks ਦੀ ਕੀਮਤ 60 ਜਾਂ ਇਸ ਤੋਂ ਵੱਧ ਨਹੀਂ ਹੈ ਤਾਂ ਅਗਲਾ else if ਚੈਕ ਕਰਦਾ ਹੈ ਕਿ ਜੇ marks ਦੀ ਕੀਮਤ 50 ਜਾਂ ਇਸ ਤੋਂ ਵੱਧ ਹੈ ਤਾਂ ਅਸੀਂ ਸੈਕੰਡ ਡਿਵੀਜ਼ਨ ਪਰਿੰਟ ਕਰਦੇ ਹਾਂ। ਇਸ ਤਰਾਂ ਹੀ 33 ਜਾਂ ਇਸ ਤੋਂ ਵੱਧ ਲਈ ਥਰਡ ਡਿਵੀਜ਼ਨ।

Similarly in last case else prints Fail.

ਇਸ ਤਰਾਂ ਹੀ ਅਖੀਰਲੇ ਕੇਸ ਵਿਚ else ਨਾਲ ਅਸੀਂ ਫੇਲ ਪਰਿੰਟ ਕਰਦੇ ਹਾਂ।

ਐਂਡ (&&) ਨੂੰ ਦਰਸ਼ਾਉਂਦਾ ਪ੍ਰੋਗਰਾਮ (Program to Demonstrate &&)

We will write a program which will find out if the student has got A grade or not.

ਅਸੀਂ ਇਕ ਪ੍ਰੋਗਰਾਮ ਲਿਖਦੇ ਹਾਂ ਜੋ ਪਤਾ ਕਰੇਗਾ ਕਿ ਵਿਦਿਆਰਥੀ ਨੇ A ਗਰੇਡ ਪ੍ਰਾਪਤ ਕੀਤਾ ਹੈ ਕਿ ਨਹੀਂ।

If the student has got 60 or more marks in subjects and 70 or more marks in sports, than he has got A grade, else B grade.

ਜੇ ਵਿਦਿਆਰਥੀ ਦੇ ਸਬਜੈਕਟਾਂ ਵਿੱਚ 60 ਜਾਂ ਇਸ ਤੋਂ ਉਪਰ ਅੰਕ ਹਨ ਅਤੇ ਖੇਡਾਂ ਵਿਚ 70 ਜਾਂ ਇਸ ਤੋਂ ਉਪਰ ਅੰਕ ਹਨ ਤਾਂ ਉਸਨੇ A ਗਰੇਡ ਪ੍ਰਾਪਤ ਕੀਤਾ ਹੈ, ਨਹੀਂ ਤਾਂ B ਗਰੇਡ।

Here we use && in if for and.

ਇਥੇ if ਵਿਚ && ਨੂੰ ਅਸੀਂ ਅਤੇ ਲਈ ਵਰਤਦੇ ਹਾਂ।

```
#include<stdio.h>
#include<conio.h>
void main()
{
	float sub, sports;

	clrscr();
	printf("Enter subject marks: ");
	scanf("%f", &sub);
	printf("Enter sports marks: ");
	scanf("%f", &sports);

	if(sub >= 60 && sports >=70)
	{
		printf("You passed in A Grade.");
	}
	else
	{
		printf("You passed in B Grade.");
	}

	getch();
}
```

ਲੂਪ (Loop)

In see if we have to do some work many times (repeat), we use loop. for, while and do-while are the three loops that we use in C.

ਸੀ ਵਿਚ ਕੋਈ ਕੰਮ ਵਾਰ-ਵਾਰ ਕਰਨਾ ਹੋਵੇ ਤਾਂ ਅਸੀਂ ਲੂਪ ਵਰਤਦੇ ਹਾਂ। for, while ਅਤੇ do-while, ਇਹ ਤਿੰਨ ਲੂਪ ਅਸੀਂ ਸੀ ਵਿਚ ਵਰਤਦੇ ਹਾਂ।
We write the below program to understand while loop.

ਅਸੀਂ ਹੇਠ ਲਿਖੇ ਕੋਡ ਨਾਲ while ਲੂਪ ਸਮਝਦੇ ਹਾਂ।

```
int i=1;
while(i<=5)
{
	printf("Bhupinder");
	i++;
}
```

With the above code Bhupinder name will be printed 5 times on the screen.

ਉਪਰ ਦਿਤੇ ਕੋਡ ਨਾਲ Bhupinder ਨਾਮ 5 ਵਾਰ ਸਕਰੀਨ ਤੇ ਪਰਿੰਟ ਹੋਵੇਗਾ।

At the very first, we declare an int variable i and put value 1 in it. Then while loop checks if the value in i is less than or equal to 5. This condition is true. Therefor while works and Bhupinder is printed. Then using i++ we increase the value of i by 1, which will become 2 now. Again the control goes back to while. 2 is again less than 5. Therefore Bhupinder is printed again. This happens 5 times and at the end after printing Bhupinder for the fifth time i becomes 6 and the condition of loop becomes false. So the loop ends.

ਸਭ ਤੋਂ ਪਹਿਲਾਂ ਅਸੀਂ i ਨੂੰ int ਡਿਕਲੇਅਰ ਕਰਕੇ ਇਸ ਵਿਚ 1 ਪਾਉਂਦੇ ਹਾਂ। ਫਿਰ while ਲੂਪ ਚੈਕ ਕਰਦੀ ਹੈ ਕਿ ਕੀ i ਦੀ ਕੀਮਤ 5 ਜਾਂ ਇਸ ਤੋਂ ਘੱਟ ਹੈ। ਇਹ ਕੰਡੀਸ਼ਨ ਠੀਕ (true) ਹੈ। ਇਸ ਲਈ while ਚਲਦੀ ਹੈ ਅਤੇ Bhupinder ਪਰਿੰਟ ਹੁੰਦਾ ਹੈ। ਫਿਰ i++ ਨਾਲ ਅਸੀਂ i ਦੀ ਕੀਮਤ ਨੂੰ 1 ਵਧਾਉਣੇ ਹਾਂ ਜੋ ਕਿ ਹੁਣ 2 ਹੋ ਜਾਵੇਗੀ। ਫਿਰ ਤੋਂ ਕੰਟਰੋਲ while ਤੇ ਜਾਂਦਾ ਹੈ। 2 ਵੀ 5 ਤੋਂ ਘੱਟ ਹੈ। ਇਸ ਲਈ Bhupinder ਫਿਰ ਪਰਿੰਟ ਹੁੰਦਾ ਹੈ। ਅਜਿਹਾ ਪੰਜ ਵਾਰ ਹੁੰਦਾ ਹੇ ਅਤੇ ਅਖੀਰ ਵਿਚ ਪੰਜਵੀਂ ਵਾਰ Bhupinder ਪੱਰਿੰਟ ਕਰਨ ਤੋਂ ਬਾਅਦ i ਦੀ ਕੀਮਤ ਜਦੋਂ 6 ਹੁੰਦੀ ਹੈ ਤਾਂ ਲੂਪ ਦੀ ਕੰਡੀਸ਼ਨ ਗਲਤ (false) ਹੋ ਜਾਂਦੀ ਹੈ ਅਤੇ ਲੂਪ ਖਤਮ ਹੋ ਜਾਂਦੀ ਹੈ।

You can make the above program using for loop also. In for loop i=1, i<=5, i++ can be written at one place itself.

ਓਪਰੋਕਤ ਪ੍ਰੋਗਰਾਮ ਤੁਸੀਂ for ਲੂਪ ਨਾਲ ਵੀ ਕਰ ਸਕਦੇ ਹੋ। for ਲੂਪ ਵਿਚ i=1, i<=5 ਅਤੇ i++ ਇਕ ਜਗਾ ਹੀ ਲਿਖੇ ਜਾ ਸਕਦੇ ਹਨ।

```
int i;
for(i=1;i<=5;i++)
{
      printf("Bhupinder\n");
}
```

We see that we have done the three tasks in for loop itself. We differ them through ;. '\n' is for going to new line. Otherwise Bhupinder will be printed 5 times in the same line.

ਅਸੀਂ ਦੇਖਦੇ ਹਾਂ ਕਿ for ਲੂਪ ਨਾਲ ਇਹ ਤਿੰਨੇ ਕੰਮ ਅਸੀਂ for ਵਿਚ ਹੀ ਕਰ ਦਿਤੇ ਹਨ। ਇਨਾਂ ਨੂੰ ਅਸੀਂ ; ਨਾਲ ਵੱਖਰਿਆਂ ਕਰਦੇ ਹਾਂ। '\n' ਨਵੀਂ ਲਾਈਨ ਵਾਸਤੇ ਹੈ। ਨਹੀਂ ਤਾਂ Bhupinder 5 ਵਾਰ ਇਕ ਲਾਈਨ ਵਿਚ ਹੀ ਪਰਿੰਟ ਹੋ ਜਾਵੇਗਾ।

ਡੂ ਵਾਇਲ (do-while) ਲੂਪ

In do-while loop we check the condition after doing the task. At the end of do-while loop we have to put ;

do-while ਲੂਪ ਵਿਚ ਅਸੀਂ ਕੰਡੀਸ਼ਨ ਕੰਮ ਕਰਨ ਤੋਂ ਬਾਅਦ ਵਿਚ ਚੈਕ ਕਰਦੇ ਹਾਂ। ਇਸ ਦੇ ਅੰਤ ਵਿਚ ; ਵੀ ਲਗਾਇਆ ਜਾਂਦਾ ਹੈ।

```
float price;
do
```

```
{
     printf("Enter price");
     scanf("%f", &price);
}while(price <=0);
```

In the above code we scan the price of a thing. If we enter the price less than or equal to 0 (wrong price), do loop will keep on asking to enter price.

ਉਪਰ ਦਿਤੇ ਕੋਡ ਵਿਚ ਅਸੀਂ ਕਿਸੇ ਚੀਜ਼ ਦਾ ਮੁਲ (price) ਸਕੈਨ ਕਰਦੇ ਹਾਂ। ਜਿੰਨੀ ਵਾਰ ਅਸੀਂ 0 ਜਾਂ ਇਸ ਤੋਂ ਘੱਟ ਮੁਲ ਐਂਟਰ ਕਰਾਂਗੇ, ਇਹ ਲੂਪ ਦੁਬਾਰਾ-ਦੁਬਾਰਾ ਮੁਲ ਐਂਟਰ ਕਰਨ ਲਈ ਕਹੇਗਾ।

do-while is executed at least once. Even if the checking condition is false at the start, then also it is sure that the loop will be executed at least once.

do-while ਘੱਟ ਤੋਂ ਘੱਟ ਇਕ ਵਾਰ ਜ਼ਰੂਰ ਚਲਦੀ ਹੈ। ਜੇ ਕੰਡੀਸ਼ਨ ਸ਼ੁਰੂ ਵਿਚ ਹੀ ਫਾਲਸ ਹੋਵੇ, ਤਾਂ ਵੀ ਇਕ ਵਾਰ ਚਲਨਾ ਯਕੀਨੀ ਹੈ।

ਸਵਿੱਚ (switch)

switch statement is also like if-else. In switch a case is executed based on the value of a variable.

switch ਵੀ if-else ਵਾਂਗ ਹੀ ਚਲਦੀ ਹੈ। ਇਸ ਵਿਚ ਕਿਸੇ ਵੇਰੀਏਬਲ ਦੀ ਕੀਮਤ ਦੇ ਅਨੁਸਾਰ ਕੇਸ ਚਲਦੇ ਹਨ।

```
switch(n)
{
     case 1:
          printf("Hockey");
          break;
     case 2:
          printf("Cricket");
          break;
     case 3:
          printf("Volleyball");
          break;
     default:
          printf("Football");
}
```

If there is 1 in n, Hockey is printed. If it is 2 Cricket and if 3 Volleyball is printed. If n has other than these values, Football is printed. In that case default is executed. n variable should be int or char. We have to always put : after case. It is important to put break after every case, other wise after executing one case the control will go to the next case automatically. break ends the switch statement after the case is done.

n ਵਿਚ ਜੇ ਕੀਮਤ 1 ਹੈ ਤਾਂ Hockey ਪਰਿੰਟ ਹੋਵੇਗਾ। ਜੇ 2 ਹੈ ਤਾਂ Cricket, ਜੇ 3 ਹੈ ਤਾਂ Volleyball । ਜੇ ਇਸ ਤੋਂ ਇਲਾਵਾ ਕੋਈ ਹੋਰ ਕੀਮਤ ਹੋਵੇ ਤਾਂ Football ਪਰਿੰਟ ਹੋਵੇਗਾ। ਇਸ ਲਈ default ਲਗਾਇਆ ਜਾਂਦਾ ਹੈ। n ਵੇਰੀਏਬਲ int ਜਾਂ char ਹੋਣਾ ਚਾਹੀਦਾ ਹੈ। case ਦੇ ਪਿਛੇ : ਲਗਾਉਣਾ ਜ਼ਰੂਰੀ ਹੈ। ਹਰ case ਤੋਂ ਬਾਅਦ break ਲਗਾਉਣਾ ਜ਼ਰੂਰੀ

ਹੈ, ਨਹੀਂ ਤਾਂ ਇਕ case ਚਲਣ ਤੋਂ ਬਾਂਅਦ ਕੰਟਰੋਲ ਆਪਨੇ ਆਪ ਦੂਸਰੇ ਕੇਸ ਵਿਚ ਚਲਿਆ ਜਾਵੇਗਾ। break ਪਰਿੰਟਿੰਗ ਤੋਂ ਬਾਅਦ ਕੰਟਰੋਲ ਖਤਮ ਕਰ ਦਿੰਦਾ ਹੈ।

ਬਰੇਕ ਅਤੇ ਕੰਟੀਨਿਊ (break and continue)

break is used to terminate (end) a loop.

break, ਲੂਪ ਨੂੰ ਤੋੜਨ (ਕੰਟਰੋਲ ਖਤਮ ਕਰਨ) ਲਈ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ।

```
int i=1;
while(i<=10)
{
      if(i==5)
      {
            break;
      }
      printf("Arshdeep");
      i++;
}
```

The above code will print Arshdeep only 4 times. When i is 5, break will terminate the loop.

ਉਪਰ ਦਿਤੀ ਲੂਪ ਸਿਰਫ 4 ਵਾਰ Arshdeep ਪਰਿੰਟ ਕਰੇਗੀ। ਜਦੋਂ i ਦੀ ਕੀਮਤ 5 ਹੋ ਜਾਵੇਗੀ ਤਾਂ break ਲੂਪ ਨੂੰ ਖਤਮ ਕਰ ਦੇਵੇਗਾ।

continue makes the loop to skip the statements, that come after continue, when the condition for continue is true. It does not terminate the loop like break does.

continue ਨਾਲ ਲੂਪ ਉਸ ਜਗਾ ਤੋਂ ਹੀ ਮੁੜ ਜਾਂਦੀ ਹੈ ਅਤੇ continue ਤੋਂ ਬਾਅਦ ਆਉਣ ਵਾਲੀਆਂ ਸਟੇਟਮੈਂਟਾ ਨੂੰ ਸਕਿਪ (ਛੱਡ) ਕਰ ਜਾਂਦੀ ਹੈ। break ਦੀ ਤਰਾਂ ਲੂਪ ਖਤਮ ਨਹੀਂ ਹੁੰਦੀ।

```
int i;
for(i=1;i<=10;i++)
{
      if(i==5)
      {
            continue;
      }
      printf("Akashdeep");
}
```

When i is 5, the statements that come after continue will be skipped. Means Akashdeep will not be printed. So Akashdeep will be printed only 9 times.

ਜਦੋਂ i ਦੀ ਕੀਮਤ 5 ਹੋਵੇਗੀ ਤਾਂ continue ਤੋਂ ਬਾਅਦ ਵਾਲੀਆਂ ਸਟੇਟਮੈਂਟਾਂ ਸਕਿਪ ਹੋ ਜਾਣਗੀਆਂ। ਮਤਲਬ Akashdeep ਪਰਿੰਟ ਨਹੀਂ ਹੋਵੇਗਾ। ਇਸ ਤਰਾਂ Akashdeep ਸਿਰਫ 9 ਵਾਰ ਪਰਿੰਟ ਹੋਵੇਗਾ।

ਪਰਾਈਮ ਨੰਬਰ ਦਾ ਪ੍ਰੋਗਰਾਮ (Program to Find Prime Number)

A prime number is a number that cannot be divided by any number. It can be divided only by 1 and itself.

ਪਰਾਈਮ ਨੰਬਰ ਓਹ ਸੰਖਿਆ ਹੁੰਦੀ ਹੈ ਜੋ ਕਿਸੇ ਨਾਲ ਵੰਡੀ ਨਹੀਂ ਜਾਂਦੀ। ਸਿਰਫ 1 ਜਾਂ ਆਪਨੇ ਆਪ ਨਾਲ ਹੀ ਵੰਡੀ ਜਾਂਦੀ ਹੈ।

Now we make a program that will tell, if the entered number is prime or not.

ਹੁਣ ਅਸੀਂ ਪ੍ਰੋਗਰਾਮ ਬਣਾਉਂਦੇ ਹਾਂ ਜੋ ਦੱਸੇਗਾ ਕਿ ਐਂਟਰ ਕੀਤੀ ਗਈ ਸੰਖਿਆ ਪ੍ਰਾਈਮ ਹੈ ਜਾਂ ਨਹੀਂ।

```
#include<stdio.h>
#include<conio.h>
void main()
{
      int n, i, p=1;

      clrscr();
      printf("Enter a number: ");
      scanf("%d", &n);

      for(i=2;i<n;i++)
      {
            if(n%i==0)
            {
                  p=0;
                  break;
            }
      }
      if(p==1)
      {
            printf("Prime");
      }
      else
      {
            printf("Not Prime");
      }

      getch();
}
```

We scan the entered number in n. The starting value of p is kept 1. The loop starts from 2 and goes 1 less than n. If n is completely dividable by i, we make p as 1 and break the loop.

ਅਸੀਂ ਐਂਟਰ ਕੀਤੀ ਗਈ ਸੰਖਿਆ n ਵਿਚ ਸਕੈਨ ਕਰਦੇ ਹਾਂ। p ਨੂੰ ਸ਼ੁਰੂਆਤੀ ਕੀਮਤ 1 ਦਿੱਤੀ ਜਾਂਦੀ ਹੈ। ਲੂਪ 2 ਤੋਂ ਸ਼ੁਰੂ ਹੋ ਕਿ n ਤੋਂ ਘੱਟ ਤੱਕ ਚਲਦੀ ਹੈ। ਜੇ n, i ਨਾਲ ਪੂਰਾ-ਪੂਰਾ ਵੰਡਿਆ ਜਾਂਦਾ ਹੈ ਤਾਂ p ਨੂੰ 0 ਕਰ ਦਿਤਾ ਜਾਂਦਾ ਹੈ ਅਤੇ break ਨਾਲ ਲੂਪ ਖਤਮ ਕਰ ਦਿੱਤੀ ਜਾਂਦੀ ਹੈ।

After the loop ends we check the value of p. If it is 1, it means that n was never divided by any value of i. Otherwise the value of p would have become 0. So if the value of p is 1, the number is prime, otherwise not.

ਲੂਪ ਖਤਮ ਹੋਣ ਤੋਂ ਬਾਅਦ ਅਸੀਂ p ਦੀ ਕੀਮਤ ਚੈਕ ਕਰਦੇ ਹਾਂ। ਜੇ ਕੀਮਤ 1 ਹੈ ਤਾਂ ਮਤਲਬ ਹੈ ਕਿ n ਕਦੇ ਵੀ i ਨਾਲ ਵੰਡਿਆ ਨਹੀਂ ਗਿਆ। ਨਹੀਂ ਤਾਂ p ਦੀ ਕੀਮਤ 0 ਹੋ ਜਾਣੀ ਸੀ। ਸੋ ਜੇ p ਦੀ ਕੀਮਤ 1 ਹੈ ਤਾਂ n ਪਰਾਈਮ ਨੰਬਰ ਹੈ। ਨਹੀਂ ਤਾਂ n ਪਰਾਈਮ ਨਹੀਂ ਹੈ।

ਫੈਕਟੋਰੀਅਲ ਦਾ ਪ੍ਰੋਗਰਾਮ (Program of Factorial)

Factorial means starting form 2, multiply by every number upto the given number. As you see the factorial of 5 is as given below.

ਫੈਕਟੋਰੀਅਲ ਦਾ ਮਤਲਬ ਹੈ ਕਿ 2 ਤੋਂ ਲੈ ਕੇ ਉਸ ਸੰਖਿਆ ਤੱਕ ਸਾਰੇ ਨੰਬਰਾਂ ਤੱਕ ਗੁਣਾ। ਜਿਵੇਂ 5 ਦਾ ਫੈਕਟੋਰੀਅਲ ਹੈ
5 x 4 x 3 x 2 = 120

Now we will enter a number and find its factorial.

ਅਸੀਂ ਹੁਣ ਕੋਈ ਸੰਖਿਆ ਐਂਟਰ ਕਰਾਂਗੇ ਅਤੇ ਉਸ ਦਾ ਫੈਕਟੋਰੀਅਲ ਲਭਾਂਗੇ।

```
#include<stdio.h>
#include<conio.h>
void main()
{
	int n, i;
	long r=1;

	clrscr();
	printf("Enter a number: ");
	scanf("%d", &n);

	for(i=n;i>=2;i--)
	{
		r=r*i;
	}

	printf("Factorial is %ld",r);

	getch();
}
```

The factorial answer is kept in r. The factorial can be easily greater than the limit of an int which is 32767. Therefor we keep r as long. The starting value of r is kept 1.

ਫੈਕਟੋਰੀਅਲ ਉੱਤਰ r ਵਿਚ ਰੱਖਿਆ ਜਾਵੇਗਾ। ਇਹ ਉੱਤਰ int ਦੀ ਲਿਮਿਟ 32767 ਤੋਂ ਵਧ ਵੀ ਸਕਦਾ ਹੈ। ਇਸ ਲਈ r ਨੂੰ long ਰੱਖਿਆ ਗਿਆ ਹੈ। r ਦੀ ਸ਼ੁਰੂਆਤੀ ਕੀਮਤ 1 ਰੱਖੀ ਜਾਂਦੀ ਹੈ।

We keep the initial value of i equal to the value of n (entered number) and decrease it one by one (i--) to 2 for the loop. Every time we multiply the value of i with r to get the new vaue of r. Therefore

when the loop works for the first time, the value of r will be 5 (if the entered number was 5). Next time r will be 5x4=20, then 20x3=60 and at last 60x2=120, which is the factorial of 5. After the loop ends we print the factorial.

i ਨੂੰ n ਦੀ ਕੀਮਤ ਤੋਂ ਸ਼ੁਰੂ ਕਰਕੇ ਇਕ-ਇਕ ਘਟਾ ਕੇ (i--), 2 ਤੱਕ ਲੂਪ ਚਲਾਈ ਜਾਂਦੀ ਹੈ। ਹਰ ਵਾਰ i ਦੀ ਕੀਮਤ ਨਾਲ ਗੁਣਾ ਕਰਕੇ r ਦੀ ਨਵੀਂ ਕੀਮਤ ਬਣਾਈ ਜਾਂਦੀ ਹੈ। ਇਸ ਤਰਾਂ r ਦੀ ਪਹਿਲੀ ਵਾਰ ਲੂਪ ਚਲਣ ਤੇ ਕੀਮਤ 5 ਹੋ ਜਾਵੇਗੀ (ਜੇ n 5 ਐਂਟਰ ਕੀਤਾ ਸੀ)। ਫਿਰ 5 x 4 = 20, ਫਿਰ 20 x 3 = 60 ਅਤੇ ਅਖੀਰ ਵਿਚ 60 x 2 = 120 ਹੋ ਜਾਵੇਗੀ ਜੋ ਕਿ 5 ਦਾ ਫੈਕਟੋਰੀਅਲ ਹੋਵੇਗਾ। ਲੂਪ ਖਤਮ ਹੋਣ ਤੋਂ ਬਾਅਦ ਅਸੀਂ ਇਸ ਨੂੰ ਪਰਿੰਟ ਕਰਦੇ ਹਾਂ।

ਕਿਸੇ ਸੰਖਿਆ ਦੇ ਅੰਕਾਂ ਨੂੰ ਜੋੜਨ ਦਾ ਪ੍ਰੋਗਰਾਮ (Program to Find Sum of Digits of a Number)

Suppose we enter 5346. We have to make a program that will add its digits, meaning
5 +4 +3 +6 = 18

ਜਿਵੇਂ ਮਨ ਲਵੋ ਅਸੀਂ ਨੰਬਰ 5346 ਐਂਟਰ ਕਰਦੇ ਹਾਂ। ਅਸੀਂ ਪ੍ਰੋਗਰਾਮ ਬਨਾਉਣਾ ਹੈ ਜੋ ਇਸਦੇ ਅੰਕਾ ਦਾ ਜੋੜ ਕਰੇ, ਮਤਲਬ 5 + 3 + 4 + 6 = 18

```
#include<stdio.h>
#include<conio.h>
void main()
{
      long n;
      int d, sum=0;

      clrscr();
      printf("Enter a number: ");
      scanf("%ld", &n);

      while(n>0)
      {
            d=n%10;
            n=n/10;
            sum=sum+d;
      }

      printf("Total of digits is %d", sum);

      getch();

}
```

We scan the number in n. Suppose we have entered 5346. The starting value of sum is kept 0.

ਸੰਖਿਆ n ਵਿਚ ਸਕੈਨ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਮੰਨ ਲਵੋ ਅਸੀਂ 5346 ਐਂਟਰ ਕੀਤਾ ਹੈ। sum ਦੀ ਸ਼ੁਰੂਆਤੀ ਕੀਮਤ 0 ਰੱਖੀ ਜਾਂਦੀ ਹੈ।

In loop we check if the value of n is greater than 0. Because the value of n is 5346, which is greater than 0, the condition is true. After dividing n by 10, the remainder is the value of d.

ਲੂਪ ਵਿਚ ਅਸੀਂ ਚੈਕ ਕਰਦੇ ਹਾਂ ਕਿ ਕੀ n ਦੀ ਕੀਮਤ 0 ਤੋਂ ਵੱਧ ਹੈ। ਕਿਉਂਕਿ n ਦੀ ਕੀਮਤ 5346 ਹੈ ਜੋ ਕਿ 0 ਤੋਂ ਵੱਧ ਹੈ, ਇਸ ਲਈ ਕੰਡੀਸ਼ਨ ਟਰੂ ਹੈ। n ਨੂੰ 10 ਨਾਲ ਭਾਗ ਕਰਕੇ ਜੋ ਬਾਕੀ ਬਚਦਾ ਹੈ ਓਹ d ਦੀ ਕੀਮਤ ਹੋਵੇਗੀ।

```
d = n % 10 ;
```

So the value of d will be 6. then we divide n by 10 and the quotient is the new value of n.

ਸੋ d ਦੀ ਕੀਮਤ 6 ਹੋਵੇਗੀ। ਫਿਰ n ਦੀ ਨਵੀਂ ਕੀਮਤ, n ਨੂੰ 10 ਨਾਲ ਭਾਗ ਕਰਕੇ ਜੋ ਉਤਰ ਆਉਂਦਾ ਹੈ, ਉਹ ਹੋਵੇਗੀ।

```
n = n / 10 ;
```

So it will be 534.

ਸੋ ਇਹ 534 ਹੋਵੇਗੀ।

The value of sum will be sum= sum + d which is 0 + 6 = 6.

sum ਦੀ ਕੀਮਤ sum = sum + d ਜੋ ਕਿ 0 + 6=6 ਬਣਦੀ ਹੈ।

The control again goes to while. The new value of n was 534, which is again greater than 0. Therefore the work explained above will be done again and d will be 4, n will be 53 and sum will become 6 + 4 = 10.

ਕੰਟਰੋਲ ਦੁਬਾਰਾ ਤੋਂ while ਤੇ ਜਾਵੇਗਾ। n ਦੀ ਨਵੀਂ ਕੀਮਤ (534) ਚੈਕ ਕੀਤੀ ਜਾਂਦੀ ਹੈ ਜੋ ਕਿ 0 ਤੋਂ ਵੱਧ ਹੈ। ਇਸ ਲਈ ਉਪਰ ਸਮਝਾਇਆ ਗਿਆ ਕਮ ਫਿਰ ਹੁੰਦਾ ਹੈ ਅਤੇ d ਦੀ ਕੀਮਤ 4, n ਦੀ ਕੀਮਤ 53 ਅਤੇ sum ਦੀ ਕੀਮਤ 6 + 4 = 10 ਹੋ ਜਾਵੇਗੀ।

This way loop will keep on continuing and at the end sum will become 6 + 4 + 3 + 5 = 18. Loop will break at the end when n becomes 0.

ਇਸ ਤਰਾਂ ਲੂਪ ਚਲਦੀ ਜਾਵੇਗੀ ਅਤੇ ਅਖੀਰ ਤੇ sum ਦੀ ਕੀਮਤ 6 + 4 + 3 + 5 = 18 ਹੋ ਜਾਵੇਗੀ ਅਤੇ n ਦੀ ਕੀਮਤ 0 ਹੋ ਕੇ ਲੂਪ ਖਤਮ ਹੋ ਜਾਵੇਗੀ।

At the end we print the value of sum.

sum ਦੀ ਕੀਮਤ ਅਖੀਰ ਵਿਚ ਪਰਿੰਟ ਕੀਤੀ ਜਾਂਦੀ ਹੈ।

ਐਰੇ (Array)

When we have to make a number of variables of same type, we use array. In array all the variables of the array have same name. They are identified by index.

ਜਦੋਂ ਅਸੀਂ ਇੱਕ ਹੀ ਟਾਈਪ ਦੇ ਕਿੰਨ੍ਹੇ ਹੀ ਵੇਰੀਏਬਲ ਬਨਾਉਣੇ ਹੋਣ ਤਾਂ ਅਸੀਂ ਐਰੇ ਵਰਤਦੇ ਹਾਂ। ਐਰੇ ਵਿਚ ਸਾਰੇ ਵੇਰੀਏਬਲਾਂ ਦਾ ਨਾਮ ਇੱਕ ਹੀ ਹੁੰਦਾ ਹੈ। ਇੰਡਕਸ ਨਾਲ ਅਸੀਂ ਇੱਕ ਵੇਰੀਏਬਲ ਦਾ ਦੂਸਰੇ ਨਾਲੋਂ ਫਰਕ ਕਰਦੇ ਹਾਂ।

int num[10];

Above we make an array of 10 int variables. The very first variable of array is num[0] and the last is num[9]. We see that the index of arrays start from 0 and goes on less than the size of array, meaning in the above example it goes until 9.

ਉਪਰ ਅਸੀਂ 10 int ਸੰਖਿਆਵਾਂ ਦਾ ਐਰੇ ਬਨਾਇਆ ਹੈ। ਸਭ ਤੋਂ ਪਹਿਲੀ ਸੰਖਿਆ num[0] ਹੋਵੇਗੀ ਅਤੇ ਸਭ ਤੋਂ ਅਖੀਰੀ ਸੰਖਿਆ num[9] ਹੋਵੇਗੀ। ਅਸੀਂ ਦੇਖਦੇ ਹਾਂ ਕਿ ਐਰੇ ਦਾ ਇੰਡਕਸ 0 ਤੋਂ ਸ਼ੁਰੂ ਹੰਦਾ ਹੈ ਅਤੇ ਸਾਈਜ਼ ਤੋਂ ਇੱਕ ਘੱਟ, ਮਤਲਬ ਉਪਰ ਦਿੱਤੀ ਉਦਾਹਰਣ ਲਈ 9 ਤੱਕ ਹੁੰਦਾ ਹੈ।

ਖੋਜ (search)

Now we make a program that will find out if the given number is in the list of entered numbers.

ਹੁਣ ਅਸੀਂ ਇੱਕ ਪ੍ਰੋਗਰਾਮ ਬਨਾਉਂਦੇ ਹਾਂ ਜੋ ਦੱਸੇਗਾ ਕਿ ਪੁਛਿਆ ਗਿਆ ਨੰਬਰ ਲਿਸਟ ਵਿੱਚ ਹੈ ਕਿ ਨਹੀਂ।

```
#include<stdio.h>
#include<conio.h>
void main()
{
	int num[10], i, t;
	int flag=0;

	clrscr();
	printf("Enter 10 numbers \n");
	for(i=0;i<=9;i++)
	{
		scanf("%d", &num[i]);
	}
	printf("Enter the number to find : ");
	scanf("%d", &t);

	for(i=0;i<=9;i++)
	{
		if(t==num[i])
		{
			flag=1;
```

```
                break;
            }
        }
        if(flag==1)
        {
            printf("Number found.");
        }
        else
        {
            printf("Number not found.");
        }
        getch();
}
```

We scan 10 numbers using for loop in num array. Then we scan the number to find in t. The starting value of flag is kept 0. Then in the second for loop, we compare the value of num[i] with the value of t. If they are equal it means, the number is in the array. We then make the value of flag as 1 and break the loop. At the end we check the value of flag. If it is 1, the number is in the list array. Otherwise, it means the value of flag remained 0, because the if statement in the for loop never got executed. So the number is not in the list array.

ਅਸੀਂ 10 ਸੰਖਿਆਵਾਂ num ਐਰੇ ਵਿਚ for ਲੂਪ ਲਗਾ ਕੇ ਸਕੈਨ ਕਰਦੇ ਹਾਂ। ਫਿਰ ਅਸੀਂ ਜੋ ਸੰਕਿਆ ਲੱਭਣੀ ਹੈ, t ਵਿੱਚ ਸਕੈਨ ਕਰਦੇ ਹਾਂ। flag ਦੀ ਸ਼ੁਰੂਆਤੀ ਕੀਮਤ 0 ਰੱਖੀ ਜਾਂਦੀ ਹੈ। ਫਿਰ ਦੂਸਰੀ for ਲੂਪ ਵਿੱਚ if ਲਗਾ ਕੇ ਅਸੀਂ ਦੇਖਦੇ ਹਾਂ ਕਿ ਕੀ num[i] ਦੀ ਕੀਮਤ t ਵਿਚਲੀ ਕੀਮਤ ਦੇ ਬਰਾਬਰ ਹੈ। ਜੇ ਹੈ ਤਾਂ ਮਤਲਬ ਨੰਬਰ ਲਿਸਟ ਵਿੱਚ ਹੈ। ਅਸੀਂ ਫਿਰ flag ਨੂੰ 1 ਕਰਕੇ ਲੂਪ ਤੋੜ ਦਿੰਦੇ ਹਾਂ। ਅਖੀਰ ਵਿੱਚ ਅਸੀਂ flag ਦੀ ਕੀਮਤ ਚੈਕ ਕਰਦੇ ਹਾਂ। ਜੇ ਇਹ 1 ਹੈ ਤਾਂ ਸੰਖਿਆ, ਐਂਟਰ ਕੀਤੀ ਲਿਸਟ ਵਿੱਚ ਹੈ। ਜੇ ਨਹੀਂ ਤਾਂ ਇਸ ਦਾ ਮਤਲਬ ਹੈ ਕਿ flag ਦੀ ਕੀਮਤ 0 ਹੀ ਰਹਿ ਗਈ ਹੈ ਕਿਉਂਕਿ for ਵਿੱਚ ਲੱਗੀ if ਨਹੀਂ ਚੱਲੀ। ਇਸ ਲਈ ਨੰਬਰ ਲਿਸਟ ਵਿੱਚ ਨਹੀਂ ਹੈ।

ਸੌਰਟ (Sort) – ਤਰਤੀਬਵਾਰ ਕਰਨਾ

We sort the 10 numbers entered in an array in ascending order.

ਅਸੀਂ ਐਰੇ ਵਿੱਚ ਲਏ ਗਏ 10 ਨੰਬਰਾਂ ਨੂੰ ਵਧਦੇ ਕਰਮ ਵਿੱਚ ਤਰਤੀਬ ਦਿੰਦੇ ਹਾਂ।

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int num[10], i, j, temp;

        clrscr();
        printf("Enter 10 numbers \n");
        for(i=0;i<=9;i++)
        {
            scanf("%d", &num[i]);
        }

        for(i=0;i<=8;i++)
        {
```

```
        for(j=i+1;j<=9;j++)
        {
            if(num[i]>num[j])
            {
                temp=num[i];
                num[i]=num[j];
                num[j]=temp;
            }
        }
    }

    printf("Sorted list \n");
    for(i=0;i<=9;i++)
    {
        printf("%d\n",num[i]);
    }

    getch();
}
```

First of all we scan 10 numbers in an array.
The second for loop is to sort the numbers. Our method is that we take the first number and compare it with remaining 9 numbers. If it is greater than any of the numbers we interchange it with that number using temp variable. Therefore at the very beginning when the value of i is 0, the nested for loop with j variable will get executed and we will get the smallest value in num[0].

ਸਭ ਤੋਂ ਪਹਿਲਾਂ ਅਸੀਂ 10 ਸੰਖਿਆਵਾਂ num ਐਰੇ ਵਿਚ ਸਕੈਨ ਕਰਦੇ ਹਾਂ।
ਦੂਸਰੀ for ਲੂਪ ਤਰਤੀਬ ਦਿੰਦੀ ਹੈ। ਸਾਡਾ ਤਰੀਕਾ ਹੈ ਕਿ ਪਹਿਲੀ ਸੰਖਿਆ ਲਈਏ ਅਤੇ ਉਸਦੀ ਬਾਕੀ ਦੀਆਂ 9 ਸੰਖਿਆਵਾਂ ਨਾਲ ਤੁਲਨਾ ਕਰੀਏ। ਜੇ ਉਹ ਵੱਡੀ ਹੈ ਤਾਂ temp ਵੇਰੀਏਬਲ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਉਸਨੂੰ ਪਿੱਛੇ ਕਰ ਦੇਈਏ ਅਤੇ ਦੂਸਰੀ ਸੰਖਿਆ ਨੂੰ ਅੱਗੇ ਕਰ ਦੇਈਏ। ਇਸ ਤਰਾਂ ਸਭ ਤੋਂ ਪਹਿਲੀ ਵਾਰ ਜਦ i ਦੀ ਕੀਮਤ 0 ਹੋਵੇਗੀ ਤਾਂ j ਵਾਲੀ ਲੂਪ ਚਲ ਕੇ num[0] ਵਿਚ ਸਭ ਤੋਂ ਛੋਟੀ ਸੰਖਿਆ ਆ ਜਾਵੇਗੀ।

Next the value of i will become 1 and j loop goes from 1 to 9. So the second smallest value will come in num[1].

ਫਿਰ i ਦੀ ਕੀਮਤ 1 ਹੋ ਜਾਵੇਗੀ ਅਤੇ j ਦੀ ਲੂਪ 1 ਤੋਂ 9 ਤੱਕ ਚਲ ਕੇ ਵਿਚ ਦੂਸਰੀ ਛੋਟੀ ਸੰਖਿਆ num[1] ਲੈ ਆਵੇਗੀ।

This will keep on continuing and at the last num[9] will contain the highest value.

ਇਸ ਤਰਾਂ ਚਲਦਾ ਰਹੇਗਾ ਅਤੇ ਅਖੀਰ ਵਿੱਚ num[9] ਵਿੱਚ ਸਭ ਤੋਂ ਵੱਡੀ ਸੰਖਿਆ ਹੋਵੇਗੀ।

The last loop prints the numbers of array, which are sorted now.

ਅਖੀਰ ਵਿੱਚ ਅਸੀਂ for ਲੂਪ ਲਗਾ ਕੇ ਸਾਰੇ ਨੰਬਰ ਪਰਿੰਟ ਕਰਦੇ ਹਾਂ ਜੋ ਕਿ ਹੁਣ ਤਰਤੀਬਵਾਰ (sorted) ਹੋਣਗੇ।

ਇਕ ਤੋਂ ਵਧ ਡਾਈਮੈਂਸ਼ਨ ਦਾ ਐਰੇ (Multi Dimesnsion Array)

Until now we have made only single dimension array. But we can make any dimension array.

ਅਸੀਂ ਹੁਣ ਤੱਕ ਇਕ ਡਾਈਮੈਂਸ਼ਨ ਦਾ ਹੀ ਐਰੇ ਬਣਾਇਆ ਹੈ। ਪਰ ਅਸੀਂ ਜਿਨੀਆਂ ਮਰਜੀ ਡਾਈਮੈਂਸ਼ਨ ਦਾ ਐਰੇ ਬਣਾ ਸਕਦੇ ਹਾਂ।

Suppose we have to sum two 3 x 3 matrices. In matrix we have rows and columns. So it is a two dimensional. Therefore we make 2 dimension array.

ਮੰਨ ਲਵੋ ਅਸੀਂ 2 3 x 3 ਦੇ ਮੈਟਰੀਸਾਂ ਦਾ ਜੋੜ ਕਰਨਾ ਹੈ। ਮੈਟਰਿਕਸ ਵਿਚ ਰੋ ਅਤੇ ਕਾਲਮ ਹੁੰਦੇ ਹਨ। ਇਸ ਲਈ ਇਹ 2 ਡਾਈਮੈਂਸ਼ਨ ਹੈ। ਅਸੀਂ 2 ਡਾਈਮੈਂਸ਼ਨ ਦਾ ਐਰੇ ਲਵਾਂਗੇ।

```
#include<stdio.h>
#include<conio.h>
void main()
{
     int m[3][3], n[3][3], p[3][3];
     int i,j;

     clrscr();
     printf("Enter first matrix data\n");
     for(i=0;i<=2;i++)
     {
          for(j=0;j<=2;j++)
          {
               scanf("%d", &m[i][j]);
          }
     }

     printf("Enter second matrix data\n");
     for(i=0;i<=2;i++)
     {
          for(j=0;j<=2;j++)
          {
               scanf("%d", &n[i][j]);
          }
     }


     for(i=0;i<=2;i++)
     {
          for(j=0;j<=2;j++)
          {
               p[i][j]=m[i][j]+n[i][j];
          }
     }

     printf("Total Matrix\n");
     for(i=0;i<=2;i++)
```

```
    {
        for(j=0;j<=2;j++)
        {
            printf("%4d", p[i][j]);
        }
        printf("\n");
    }

    getch();
}
```

First we scan the numbers of first matrix in m array and the numbers of second in n array. Then we add them and put in p array.

ਅਸੀਂ ਪਹਿਲਾਂ m ਐਰੇ ਵਿਚ ਪਹਿਲੇ ਮੈਟਰਿਕਸ ਦੀਆਂ ਸੰਖਿਆਵਾਂ ਲੈਂਦੇ ਹਾਂ ਅਤੇ n ਐਰੇ ਵਿੱਚ ਦੂਸਰੇ ਮੈਟਰਿਕਸ ਦੀਆਂ। ਫਿਰ ਅਸੀਂ p ਐਰੇ ਵਿਚ ਇਹਨਾਂ ਦੋਨਾਂ ਮੈਟਰਿਸਾਂ ਦਾ ਜੋੜ ਪਾਉਣੇ ਹਾਂ।

At the end we print p matrix. We provide 4 spaces for each number of p matrix for printing. After each row is ended we go to new line using '\n'. This is done so that the matrix is displayed in proper readable format.

ਅਖੀਰ ਵਿਚ ਅਸੀਂ p ਮੈਟਰਿਕਸ ਨੂੰ ਪਰਿੰਟ ਕਰਦੇ ਹਾਂ। ਅਸੀਂ p ਮੈਟਰਿਕਸ ਦੀ ਹਰ ਸੰਖਿਆ ਨੂੰ 4 ਥਾਵਾਂ ਪਰਿੰਟ ਲਈ ਦਿੰਦੇ ਹਾਂ। ਹਰ ਰੋ ਖਤਮ ਹੋਣ ਤੋਂ ਬਾਅਦ '\n' ਪਰਿੰਟ ਕਰਕੇ ਨਵੀਂ ਲਾਈਨ ਬਣਾਉਂਦੇ ਹਾਂ ਤਾਂਕਿ ਮੈਟਰਿਕਸ ਠੀਕ ਤਰਾਂ ਸਕਰੀਨ ਉਪਰ ਦਿਖੇ।

ਸਟਰਿੰਗਸ (Strings)

String is basically a line. As we know single character (char), if we have an array of number of characters we call it a String. For knowing the end of String we use NULL. NULL can also be written as '\0'.

ਸਟਰਿੰਗ ਇੱਕ ਲੜੀ ਨੂੰ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਜਿਵੇਂ ਕਿ ਕਰੈਕਟਰ (char) ਇੱਕ ਅੱਖਰ ਨੂੰ ਕਿਹਾ ਜਾਂਦਾ ਹੈ, ਜੇ ਅੱਖਰਾਂ ਦੀ ਲੜੀ ਇੱਕ ਐਰੇ ਵਿੱਚ ਰੱਖੀਏ ਤਾਂ ਉਸ ਨੂੰ ਸਟਰਿੰਗ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਲੜੀ ਦਾ ਅੰਤ ਦੱਸਣ ਲਈ ਅਸੀਂ ਨੱਲ (NULL) ਵਰਤਦੇ ਹਾਂ। ਨੱਲ ਨੂੰ ਅਸੀਂ '\0' ਵੀ ਲਿੱਖ ਸਕਦੇ ਹਾਂ।

```
char name[]={'H','A','P','P','Y','\0'};
printf("%s",name);
```

The above statements result will be Happy printed on screen. we see that name is an array. We have kept Happy string in it. You can see that we have kept null ('\0') at the end of Happy. We need not tell even the size of array, the compiler will count the characters in Happy and automatically make array of that size. In the above case the size of array will be kept 6. We can also make a string in an easy way.

ਉਪਰੋਕਤ ਸਟੇਟਮੈਂਟਾਂ ਨਾਲ ਸਕਰੀਨ ਤੇ Happy ਪਰਿੰਟ ਹੋਵੇਗਾ। ਅਸੀਂ ਦੇਖਦੇ ਹਾਂ ਕਿ name ਇੱਕ ਐਰੇ ਹੈ। ਇਸ ਵਿਚ Happy ਸਟਰਿੰਗ ਰੱਖਿਆ ਜਾਂਦਾ ਹੈ। ਤੁਸੀਂ ਦੇਖ ਸਕਦੇ ਹੋ ਕਿ Happy ਦੇ ਅੰਤ ਵਿੱਚ ਅਸੀਂ ਨੱਲ ('\0') ਰੱਖਿਆ ਹੈ। ਸਾਨੂੰ ਐਰੇ ਦਾ ਸਾਈਜ਼ ਵੀ ਦੱਸਣ ਦੀ ਲੋੜ ਨਹੀਂ ਪਈ, ਇਸ ਕੇਸ ਵਿੱਚ ਕੰਪਾਈਲਰ Happy ਦੇ ਅੱਖਰ ਗਿਣ ਕੇ ਖੁਦ ਹੀ ਐਰੇ ਦਾ ਸਾਈਜ਼ ਅਲਾਟ ਕਰ ਦਿੰਦਾ ਹੈ। ਉਪਰੋਕਤ ਉਦਾਹਰਣ ਵਿੱਚ ਐਰੇ ਦਾ ਸਾਈਜ਼ 6 ਰੱਖਿਆ ਜਾਵੇਗਾ। ਅਸੀਂ ਸਟਰਿੰਗ ਹੋਰ ਵੀ ਸੌਖੇ ਤਰੀਕੇ ਨਾਲ ਰੱਖ ਸਕਦੇ ਹਾਂ:

```
char name[]="Happy";
```

This is also perfectly correct. It will also make an array of 6 characters and keep Happy in it. At the 6^{th} place it will be null ('\0').

ਇਹ ਵੀ ਬਿਲਕੁਲ ਠੀਕ ਹੈ। ਇਸ ਨਾਲ ਵੀ 6 ਕਰੈਕਟਰਾਂ ਦਾ ਐਰੇ ਬਣੇਗਾ ਅਤੇ ਉਸ ਵਿੱਚ Happy ਜੋਵੇਗਾ। ਛੇਵੇਂ ਸਥਾਨ ਤੇ ਆਪੇ ਹੀ ਨੱਲ ('\0') ਪੈ ਜਾਵੇਗਾ।

ਸਟਰਿੰਗ ਫੰਕਸ਼ਨਸ (String Functions)

To work on strings C has given many functions.

ਸਟਰਿੰਗਸ ਤੇ ਕੰਮ ਕਰਨ ਲਈ ਸੀ ਨੇ ਕਈ ਬਣੇ-ਬਣਾਏ ਫੰਕਸ਼ਨ ਦਿੱਤੇ ਹਨ।

strlen()

This function gives us the length of the string.

ਇਹ ਫੰਕਸ਼ਨ ਸਾਨੂੰ ਸਟਰਿੰਗ ਦੀ ਲੰਬਾਈ (length) ਦਸਦਾ ਹੈ।

```
int l;
char name[20];

printf("Enter a name: ");
gets(name);

l=strlen(name);
printf("Length of name is %d",l);
```

We first keep an array of 20 characters. Then using gets() we scan a string from the user. gets() function can scan string with spaces also. Suppose you have entered Bhupinder Singh. scanf() will only scan Bhupinder. But gets() will scan full Bhupinder Singh. Then strlen() finds the length of Bhupinder Singh entered in name array, which is 15 and puts it in l variable. Remember that the space between Bhupinder and Singh will also be counted. Then we print the value of l.

ਅਸੀਂ ਪਹਿਲਾਂ 20 ਕਰੈਕਟਰਾਂ ਦਾ ਐਰੇ ਰੱਖਦੇ ਹਾਂ। ਫਿਰ ਅਸੀਂ gets() ਫੰਕਸ਼ਨ ਨਾਲ ਯੂਸਰ (ਵਰਤੋਂਕਰਤਾ) ਤੋਂ ਸਟਰਿੰਗ ਸਕੈਨ ਕਰਦੇ ਹਾਂ। gets() ਫੰਕਸ਼ਨ ਸਪੇਸ (ਖਾਲੀ ਥਾਂ) ਵੀ ਸਕੈਨ ਕਰ ਲੈਂਦਾ ਹੈ। ਮੰਨ ਲਵੋ ਤੁਸੀਂ Bhupinder Singh ਐਂਟਰ ਕੀਤਾ ਹੈ। scanf() ਇਕੱਲਾ Bhupinder ਹੀ ਲਵੇਗਾ। ਪਰ gets() ਪੂਰਾ Bhupinder Singh ਲਵੇਗਾ। ਫਿਰ strlen(), name ਵਿੱਚ ਪਏ Bhupinder Singh ਦੀ ਲੰਬਾਈ ਲੱਭ ਕੇ l ਵਿੱਚ ਪਾਉਂਦਾ ਹੈ, ਜੋ ਕਿ 15 ਹੈ। ਯਾਦ ਰੱਖੋ Bhupinder ਅਤੇ Singh ਵਿਚਕਾਰ ਜੋ ਸਪੇਸ ਹੈ, ਉਹ ਵੀ ਗਿਣੀ ਜਾਵੇਗੀ। ਫਿਰ ਅਸੀਂ l ਨੂੰ ਪਰਿੰਟ ਕਰਦੇ ਹਾਂ।

strcpy()

With this function we copy a string into another.

ਇਸ ਫੰਕਸ਼ਨ ਨਾਲ ਅਸੀਂ ਇੱਕ ਸਟਰਿੰਗ ਨੂੰ ਦੂਸਰੇ ਸਟਰਿੰਗ ਵਿੱਚ ਕਾਪੀ (ਨਕਲ) ਕਰਦੇ ਹਾਂ।

```
char sm[]="Arshdeep";
char sn[20];
strcpy(sn, sm);
```

By the above statements the string of sm that is Arshdeep will be copied into sn. Now both sm and sn will have Arshdeep.

ਉਪਰੋਕਤ ਸਟੇਟਮੈਂਟਾ ਨਾਲ sm ਦਾ ਸਟਰਿੰਗ, ਜੋ ਕਿ Arshdeep ਹੈ, sn ਵਿੱਚ ਕਾਪੀ ਹੋ ਜਾਵੇਗਾ। ਹੁਣ sm ਅਤੇ sn ਦੋਨਾਂ ਵਿੱਚ Arshdeep ਹੋਵੇਗਾ।

Remember, if there is any previous string in sm, it will be erased.

ਯਾਦ ਰੱਖੋ ਵਿੱਚ ਜੇ ਪਹਿਲਾਂ ਕੋਈ ਸਟਰਿੰਗ sm ਹੈ ਤਾਂ ਉਹ ਮਿਟ ਜਾਵੇਗਾ।

strcat()

This function concatenates (join) one string at the end of another.

ਇਸ ਫੰਕਸ਼ਨ ਨਾਲ ਇੱਕ ਸਟਰਿੰਗ ਨੂੰ ਦੂਸਰੇ ਦੇ ਪਿੱਛੇ ਜੋੜਿਆ ਜਾਂਦਾ ਹੈ।

```
char sm[30]="Arshdeep";
char sn[]="Akashdeep";
strcat(sm, sn);
```

Above we see that we have kept the size of sm as 30. If have have not kept 30, the compiler would have kept the size 9, which is one greater than the length of Arshdeep. We have kept the size 30 because we need space after Arshdeep to join Akashdeep. strcat() joins the string of sn at the end of sm. strcat() does not erase any existing string while joining. Now sm will have ArshdeepAkashdeep.

ਉਪਰ ਅਸੀਂ ਦੇਖਦੇ ਹਾਂ ਕਿ sm ਸਾਈਜ਼ ਅਸੀਂ 30 ਰੱਖਿਆ ਹੈ। ਜੇ ਅਸੀਂ 30 ਨਾ ਰਖਦੇ ਤਾਂ ਕੰਪਾਈਲਰ ਨੇ ਸਾਈਜ਼ 9 ਰੱਖਣਾ ਸੀ ਜੋ ਕਿ Arshdeep ਦੀ ਲੰਬਾਈ ਤੋਂ ਇੱਕ ਜਿਆਦਾ ਹੈ। ਅਸੀਂ ਲੰਬਾਈ 30 ਰੱਖੀ ਹੈ ਕਿਉਂਕਿ Arshdeep ਦੇ ਪਿੱਛੇ Akashdeep ਜੋੜਨ ਲਈ ਜਗਾ ਚਾਹੀਦੀ ਹੈ। strcat(), sm ਦੇ ਪਿੱਛੇ sn ਦਾ ਸਟਰਿੰਗ ਜੋੜਦਾ ਹੈ।strcat() ਕਾਪੀ ਕਰਦੇ ਸਮੇਂ ਪਿਹਲਾਂ ਵਾਲੇ ਸਟਰਿੰਗ ਨੂੰ ਨਹੀਂ ਮਿਟਾਉਂਦਾ। ਸੋ ਹੁਣ sm ਵਿਚ ArshdeepAkashdeep ਹੋਵੇਗਾ।

strcmp()

This is a very useful function which compares two strings and tells us if they are same or not. If both the strings are same, the result returned is 0. If the first string comes before the second string in A, B, C… order, the result will be a negative number like -37 etc. If the first string comes after the second string in A, B, C…. order the result will be any positive number like 37 etc.

ਇਹ ਵੀ ਬਹੁਤ ਉਪਯੋਗੀ ਫੰਕਸ਼ਨ ਹੈ ਜੋ ਸਾਨੂੰ ਦੋ ਸਟਰਿੰਗਾਂ ਦੀ ਤੁਲਨਾ ਕਰਕੇ ਦਸਦਾ ਹੈ ਕਿ ਇਹ ਇੱਕੋ ਜਿਹੇ ਹਨ ਕਿ ਨਹੀਂ। ਜੇ ਦੋਨੋਂ ਇੱਕੋ ਜਿਹੇ ਹੋਣ ਤਾਂ 0 ਰਿਸਲਟ ਦਿੰਦਾ ਹੈ। ਜੇ ਪਹਿਲਾ ਸਟਰਿੰਗ ਤਰਤੀਬ ਵਿੱਚ ਦੂਸਰੇ ਨਾਲੋਂ ਪਹਿਲਾਂ ਆਉਂਦਾ ਹੈ ਤਾਂ ਰਿਸਲਟ ਕੋਈ ਨੈਗਟਿਵ ਨੰਬਰ ਜਿਵੇਂ -37 ਵਗੈਰਾ ਹੋਵੇਗਾ। ਜੇ ਪਹਿਲਾ ਸਟਰਿੰਗ ਤਰਤੀਬ ਵਿੱਚ ਪਿੱਛੋਂ ਆਉਂਦਾ ਹੈ ਤਾਂ ਰਿਸਲਟ ਕੋਈ ਪੌਸਟਿਵ ਨੰਬਰ ਜਿਵੇਂ 37 ਵਗੈਰਾ ਹੋਵੇਗਾ।

```
char s1[]="Raman";
char s2[]="Akash";
char s3[]="Raman";
char s4[]="akash";

int n1, n2, n3;

n1=strcmp(s1, s2);
n2=strcmp(s1, s3);
n3=strcmp(s2, s4);
```

n1 will have a positive number, because Raman comes after Akash in alphabetical order. n2 will have 0 because Raman and Raman are same. n3 will have -32 because the difference between A and a of Akash and akash respectively is 32.

n1 ਵਿੱਚ ਕੋਈ ਪੌਸਟਿਵ ਨੰਬਰ ਹੋਵੇਗਾ, ਕਿਉਂਕਿ Raman, Akash ਤੋਂ ਤਰਤੀਬ ਵਿੱਚ ਪਿੱਛੋਂ ਆਉਂਦਾ ਹੈ। n2 ਵਿੱਚ 0 ਹੋਵੇਗਾ, ਕਿਉਂਕਿ Raman ਅਤੇ Raman ਇੱਕੋ ਜਿਹੇ ਹਨ। n3 ਵਿੱਚ -32 ਹੋਵੇਗਾ, ਕਿਉਂਕਿ Akash ਅਤੇ akash ਵਿੱਚ A ਅਤੇ a ਦੇ ਕੋਡ ਦਾ ਫਰਕ 32 ਹੁੰਦਾ ਹੈ।

strcmpi()

In the last case we have seen that strcmp() does not treat Akash and akash as equal because there is difference between code of A and a. But if we use strcmpi() function, they are treated as equal. In strcmpi(), the meaning of i is to ignore case.

ਅਸੀਂ ਪਿਛਲੀ ਉਦਾਹਰਣ ਵਿੱਚ ਦੇਖਿਆ ਕਿ strcmp() ਫੰਕਸ਼ਨ, Akash ਅਤੇ akash ਨੂੰ ਬਰਾਬਰ ਨਹੀਂ ਸਮਝਦਾ ਕਿਉਂਕਿ A ਅਤੇ a ਦੇ ਕੋਡ ਵਿੱਚ ਫਰਕ ਹੈ। ਪਰ ਜੇ ਅਸੀਂ strcmpi() ਫੰਕਸ਼ਨ ਵਰਤਦੇ ਹਾਂ ਤਾਂ ਇਹਨਾ ਨੂੰ ਬਰਾਬਰ ਮੰਨਿਆ ਜਾਂਦਾ ਹੈ। strcmpi() ਵਿੱਚ i ਦਾ ਮਤਲਬ ਹੈ ਜਾਨਿ ਕਿ ਛੋਟੇ ਜਾਂ ਵੱਡੇ ਅੱਖਰਾਂ ਵਿੱਚ ਫਰਕ ਨਹੀਂ ਕਰਨਾ।

```
char s1[]="Akash";
char s2[]="akash";
int n;
n=strcmpi(s1,s2);
```

Now n will have 0, because according to strcmpi(), Akash and akash are same.

ਹੁਣ n ਵਿੱਚ 0 ਰਿਸੱਲਟ ਆਵੇਗਾ ਕਿਉਂਕਿ strcmpi() ਅਨੁਸਾਰ Akash ਅਤੇ akash ਬਰਾਬਰ ਹਨ।

ਸਟਰਿੰਗ ਨੂੰ ਤਰਤੀਬ ਦੇਣ ਵਾਲਾ ਪ੍ਰੋਗਰਾਮ (Sorting of Strings)

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
      char nm[10][20], temp[20];
      int i, j, r;

      clrscr();
      printf("Enter 10 names \n");
      for(i=0;i<=9;i++)
      {
            gets(nm[i]);
      }

      for(i=0;i<=8;i++)
      {
            for(j=i+1;j<=9;j++)
            {
                  r=strcmpi(nm[i], nm[j]);
                  if(r>0)
                  {
                        strcpy(temp, nm[i]);
                        strcpy(nm[i], nm[j]);
                        strcpy(nm[j], temp);
                  }
            }
      }

      printf("Sorted names\n");
      for(i=0;i<=9;i++)
```

```
    {
        printf("%s\n",nm[i]);
    }

    getch();
}
```

To use string functions like strcmpi() etc, we have to include string.h file at the top. We use the same method to sort strings, which we used for numbers. But now for comparing we have to use strcmpi() and to copy we have to use strcpy(). After the sorting is done we use %s in printf to print strings. We can also use puts() function instead of printf(). puts() functions automatically takes us to new line after printing one string.

ਸਟਰਿੰਗ ਦੇ ਫੰਕਸ਼ਨ ਜਿਵੇਂ ਕਿ strcmpi() ਵਗੈਰਾ ਵਰਤਨ ਲਈ ਸਾਨੂੰ string.h ਫਾਈਲ include ਕਰਨੀ ਪੈਂਦੀ ਹੈ। ਤਰਤੀਬ ਦਾ ਕੰਮ ਅਸੀਂ ਉਸੇ ਤਰਾਂ ਹੀ ਕਰਦੇ ਹਾਂ ਜਿਵੇਂ ਅਸੀਂ ਸੰਖਿਆਵਾਂ ਲਈ ਤਰੀਕਾ ਲਗਾਇਆ ਸੀ। ਪਰ ਹੁਣ ਸਾਨੂੰ ਤੁਲਨਾ ਕਰਨ ਲਈ strcmpi() ਅਤੇ ਕਾਪੀ ਕਰਨ ਲਈ strcpy() ਵਰਤਣਾ ਹੈ। ਤਰਤੀਬ ਹੋਣ ਤੋਂ ਬਾਅਦ ਸਟਰਿੰਗ ਪਰਿੰਟ ਕਰਨ ਲਈ ਅਸੀਂ printf() ਵਿੱਚ %s ਵਰਤਦੇ ਹਾਂ। ਜਾਂ ਅਸੀਂ printf() ਦੀ ਜਗਾ puts() ਵੀ ਵਰਤ ਸਕਦੇ ਹਾਂ, ਜੋ ਕਿ ਸਟਰਿੰਗ ਪਰਿੰਟ ਕਰਨ ਤੋਂ ਬਾਅਦ ਨਵੀਂ ਲਾਈਨ ਤੇ ਆਪੇ ਚਲਾ ਜਾਂਦਾ ਹੈ।

ਫੰਕਸ਼ਨਸ (Functions)

Functions make the work of programming very easy. Functions in itself is a code that has the capability of completing a task. Till now we have used printf() and scanf(). printf() knows its task of printing well, similarly scanf() knows how to get input. These are ready-made functions provided by C. We use these functions in our program. We also call them library functions.

ਫੰਕਸ਼ਨ ਪ੍ਰੋਗਰਾਮਿੰਗ ਦਾ ਕੰਮ ਬਹੁਤ ਸੌਖਾ ਕਰ ਦਿੰਦੇ ਹਨ। ਫੰਕਸ਼ਨ ਆਪਨੇ ਆਪ ਵਿੱਚ ਇੱਕ ਕੋਡ ਹੁੰਦਾ ਹੈ ਜੋ ਕੋਈ ਕੰਮ ਪੂਰਨ ਰੂਪ ਵਿੱਚ ਇਕੱਲੇ ਤੌਰ ਤੇ ਕਰਕੇ ਦਿੰਦਾ ਹੈ। ਅਸੀਂ ਹੁਣ ਤੱਕ printf(), scanf() ਵਗੈਰਾ ਨੂੰ ਵਰਤਿਆ ਹੈ।printf() ਪਰਿੰਟਿੰਗ ਦਾ ਕੰਮ ਕਰਦਾ ਹੈ ਅਤੇ scanf() ਇਨਪੁਟ ਦਾ ਕੰਮ ਕਰਦਾ ਹੈ। ਇਹ ਦੋਨੋਂ ਸੀ ਦੁਆਰਾ ਦਿੱਤੇ ਗਏ ਬਨੇ-ਬਨਾਏ ਫੰਕਸ਼ਨ ਹਨ। ਅਸੀਂ ਇਹਨਾਂ ਨੂੰ ਆਪਣੇ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਵਰਤਦੇ ਹਾਂ। ਅਸੀਂ ਇਹਨਾਂ ਨੂੰ ਲਾਈਬਰੇਰੀ ਫੰਕਸ਼ਨ ਵੀ ਕਹਿੰਦੇ ਹਾਂ।

Now we will make our own functions. We take a very simple example. We have to get the addition of two numbers done by a function. We keep add as the name of the function. We will give it two numbers, which are called its arguments. We have to get the total of these two back from function. This total is called return of the function. So this function will be something like this.

ਹੁਣ ਅਸੀਂ ਖੁਦ ਫੰਕਸ਼ਨ ਬਨਾਵਾਂਗੇ। ਅਸੀਂ ਬਹੁਤ ਸੌਖੀ ਉਦਾਹਰਣ ਲੈਂਦੇ ਹਾਂ। ਅਸੀਂ ਦੋ ਨੰਬਰਾਂ ਦਾ ਜੋੜ ਇੱਕ ਫੰਕਸ਼ਨ ਤੋਂ ਕਰਾਂਉਂਦੇ ਹਾਂ। ਅਸੀਂ ਫੰਕਸ਼ਨ ਦਾ ਨਾਮ add ਰਖਦੇ ਹਾਂ। ਅਸੀਂ ਇਸ ਨੂੰ ਦੋ ਸੰਖਿਆਵਾਂ ਦੇਵਾਂਗੇ ਜਿਨਾਂ ਨੂੰ ਅਸੀਂ ਆਰਗਿਊਮੈਂਟਸ (arguments) ਕਹਿੰਦੇ ਹਾਂ। ਅਸੀਂ ਫੰਕਸ਼ਨ ਤੋਂ ਇਹਨਾਂ ਦੋਨਾਂ ਦਾ ਜੋੜ ਰਿਸੱਲਟ ਲੈਣਾ ਹੈ। ਇਸ ਨੂੰ ਰਿਟੱਰਨ (return) ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਸੋ ਇਹ ਫੰਕਸ਼ਨ ਕੁਝ ਇਸ ਪ੍ਰਕਾਰ ਹੋਵੇਗਾ।

```
int add(int n1, int n2)
{
      int c;
      c=n1+n2;
      return c;
}
```

int before add tells us that add function is going to return us an int value. n1 and n2 are arguments. These numbers have to be passed to add function when we call it. The function adds n1 and n2 and put the result in c. It returns c back to us. Because c is an int, therefore the return type of add is int. The whole code written above is called function definition. By definition, it means the actual full code of the function. The above function can be used in aprogram as shown below.

add ਦੇ ਅੱਗੇ ਜੋ int ਲਿਖਿਆ ਹੈ, ਉਸਦਾ ਮਤਲਬ ਹੈ ਕਿ add ਫੰਕਸ਼ਨ ਸਾਨੂੰ ਇੱਕ int ਰਿਟਰਨ ਕਰੇਗਾ। n1 ਅਤੇ n2 ਆਰਗਿਊਮੈਂਟਸ ਹਨ। ਇਹ ਸੰਖਿਆਵਾਂ ਅਸੀਂ ਫੰਕਸ਼ਨ ਨੂੰ ਵਰਤਦੇ ਸਮੇ (call), ਇਸ ਨੂੰ ਪਾਸ ਕਰਨੀਆਂ ਹਨ। ਫੰਕਸ਼ਨ n1 ਅਤੇ n2 ਨੂੰ ਜੋੜ ਕੇ c ਵਿੱਚ ਪਾਉਂਦਾ ਹੈ ਅਤੇ ਇਸ ਨੂੰ ਸਾਨੂੰ ਰਿਟੱਰਨ (ਵਾਪਿਸ) ਕਰਦਾ ਹੈ। ਕਿਉਂਕਿ c ਇੱਕ int ਹੈ, ਇਸ ਲਈ add ਦੀ ਰਿਟੱਰਨ int ਹੈ। ਜੋ ਅਸੀਂ ਉਪਰ ਕੋਡ ਲਿਖਿਆ ਹੈ ਉਸ ਨੂੰ ਫੰਕਸ਼ਨ ਡੈਫੀਨੇਸ਼ਨ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਡੈਫੀਨੇਸ਼ਨ ਦਾ ਮਤਲਬ ਹੈ ਫੰਕਸ਼ਨ ਦਾ ਪੂਰਾ ਕੋਡ। ਉੱਪਰ ਲਿਖੇ ਫੰਕਸ਼ਨ ਨੂੰ ਅਸੀਂ ਪੂਰੇ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਹੇਠ ਲਿਖੇ ਅਨੁਸਾਰ ਵਰਤਦੇ ਹਾਂ।

```
#include<stdio.h>
#include<conio.h>
void main()
{
	int add(int n1, int n2);
	int a, b, r;

	clrscr();
	printf("Enter 2 numbers ");
	scanf("%d %d",&a, &b);

	r=add(a,b);

	printf("Total %d",r);

	getch();
}
int add(int n1, int n2)
{
	int c;
	c=n1+n2;
	return c;
}
```

r=add(a, b) is call to function add. We give a and b to add() get the result in r. Before making call to add(), main() function should know about add(). For this, at the beginning of main() we write int add(int n1, int n2). This is called function declaration. It is to be noted that the values of a and b go into n1 and n2 respectively and the value of c comes into r.

r=add(a, b) ਫੰਕਸ਼ਨ ਕਾਲ ਹੈ। ਅਸੀਂ a ਅਤੇ b, add() ਨੂੰ ਦਿੰਦੇ ਹਾਂ ਅਤੇ ਉੱਤਰ r ਵਿੱਚ ਲੈਂਦੇ ਹਾਂ। add() ਨੂੰ ਵਰਤਨ (call) ਤੋਂ ਪਹਿਲਾਂ main() ਨੂੰ add() ਬਾਰੇ ਪਤਾ ਹੋਣਾ ਚਾਹੀਦਾ ਹੈ। ਇਸ ਲਈ ਅਸੀਂ main() ਦੇ ਸ਼ੁਰੂ ਵਿੱਚ int add(int n1, n2) ਲਿਖਦੇ ਹਾਂ। ਇਸ ਨੂੰ ਫੰਕਸ਼ਨ ਡੈਕਲਰੇਸ਼ਨ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਧਿਆਨ ਦੇਣ ਯੋਗ ਹੈ ਕਿ a ਅਤੇ b ਦੀ ਕੀਮਤ ਕਰਮਵਾਰ n1 ਅਤੇ n2 ਵਿੱਚ ਜਾਂਦੀ ਹੈ ਅਤੇ c ਦੀ ਕੀਮਤ r ਵਿੱਚ ਮਿਲਦੀ ਹੈ।

ਰਿਕਰਸ਼ਨ (recursion)

If a function calls itself, we call it recursion. We will now make the program of factorial using recursion.

ਜੇ ਫੰਕਸ਼ਨ ਆਪਨੇ ਆਪ ਨੂੰ ਹੀ ਕਾਲ ਕਰੇ ਤਾਂ ਉਸ ਨੂੰ ਰਿਕਰਸ਼ਨ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਅਸੀਂ ਹੁਣ ਰਿਕਰਸ਼ਨ ਨਾਲ ਫੈਕਟੋਰੀਅਲ ਦਾ ਪ੍ਰੋਗਰਾਮ ਬਨਾਉਂਦੇ ਹਾਂ।

```
#include<stdio.h>
#include<conio.h>
void main()
{
	long factorial(int n);
	int n;
```

```
	long r;

	clrscr();
	printf("Enter a number ");
	scanf("%d", &n);

	r=factorial(n);

	printf("Factorial is %ld", r);

	getch();

}

long factorial(int n)
{
	long res;

	if(n==1)
	{
		return 1;
	}
	else
	{
		res=n*factorial(n-1);
		return res;
	}
}
```

We will pass a number n to factorial function. The function will calculate and give us the factorial of this number. For finding the factorial, the function calls itself.

factorial ਫੰਕਸ਼ਨ ਨੂੰ ਅਸੀਂ ਇੱਕ ਨੰਬਰ n ਭੇਜਾਂਗੇ ਅਤੇ ਇਹ ਸਾਨੂੰ ਉਸ ਦਾ ਫੈਕਟੋਰੀਅਲ ਲੱਭ ਕੇ ਦੇਵੇਗਾ। ਫੈਕਟੋਰੀਅਲ ਕੱਢਣ ਲਈ factorial ਫੰਕਸ਼ਨ ਆਪਣੇ ਆਪ ਨੂੰ ਹੀ ਕਾਲ ਕਰਦਾ ਰਹਿੰਦਾ ਹੈ।

Suppose the value of n passed to function is 5. if statement checks whether the value of n is 1. It is not so the value of res in else becomes:

ਮੰਨ ਲਵੋ ਅਸੀਂ n ਦੀ ਕਮਿਤ 5 ਭੇਜਦੇ ਹਾਂ। if ਚੈਕ ਕਰਦੀ ਹੈ, ਕਿ ਕੀ n ਦੀ ਕੀਮਤ 1 ਹੈ, ਜੋ ਕਿ ਨਹੀਂ ਹੈ ਸੋ else ਵਿੱਚ res ਦੀ ਕੀਮਤ

5*factorial(5-1);
=5*factorial(4)

Meaning 5*(call to function itself with 4)

ਹੋ ਜਾਂਦੀ ਹੈ। ਜਾਨਿ ਕਿ 5*(ਆਪਨੇ ਆਪ ਨੂੰ 4 ਨਾਲ ਕਾਲ)।

Next time the value is

ਅਗਲੀ ਵਾਰ ਕੀਮਤ

5*4*factorial(3);

ਹੋਵੇਗੀ।

At last when the value of n decreases to 1 the factorial function stops recursion and returns 1 and we get

ਅੰਤ ਵਿੱਚ ਜਦੋਂ n ਦੀ ਕੀਮਤ ਘਟ ਕੇ 1 ਹੋ ਜਾਂਦੀ ਹੈ ਤਾਂ ਫੈਕਟੋਰੀਅਲ ਫੰਕਸ਼ਨ ਰਿਕੱਰਸ਼ਨ ਬੰਦ ਕਰਕੇ 1 ਰਿਟਰਨ ਕਰਦਾ ਹੈ ਅਤੇ ਸਾਨੂੰ

5*4*3*2*1

which is the factorial of 5. main() gets it in r and prints it.

ਮਿਲਦਾ ਹੈ ਜੋ ਕਿ 5 ਦਾ ਫੈਕਟੋਰੀਅਲ ਹੈ। main() ਨੂੰ ਇਹ r ਵਿੱਚ ਮਿਲਦਾ ਹੈ ਜਿਸਨੂੰ ਅਸੀਂ ਪਰਿੰਟ ਕਰਦੇ ਹਾਂ।

ਪੋਇੰਟਰਸ (Pointers)

Whenever we make a variable, the computer keeps a space for it in the memory (RAM). Sometimes we need to know the address of this memory. For this we have to use pointers. Pointers are variables in which we store the address of a memory location.

ਜਦੋਂ ਵੀ ਅਸੀਂ ਕੋਈ ਵੇਰੀਏਬਲ ਬਨਾਉਂਦੇ ਹਾਂ ਤਾਂ ਕੰਪਿਊਟਰ ਮੈਮੋਰੀ (RAM) ਵਿੱਚ ਉਸ ਲਈ ਜਗਾ ਰਾਖਵੀਂ ਕਰਦਾ ਹੈ। ਕਈ ਵਾਰ ਸਾਨੂੰ ਇਸ ਮੈਮੋਰੀ ਦੇ ਅਡਰੈਸ (ਪਤੇ) ਦੀ ਲੋੜ ਪੈਂਦੀ ਹੈ। ਤਾਂ ਇਸ ਲਈ ਸਾਨੂੰ ਪੋਇੰਟਰ ਵਰਤਣੇ ਪੈਂਦੇ ਹਨ। ਪੋਇੰਟਰ ਇੱਕ ਅਜਿਹਾ ਵੇਰੀਏਬਲ ਹੁੰਦਾ ਹੈ ਜਿਸ ਵਿੱਚ ਮੈਮੋਰੀ ਦਾ ਅਡਰੈਸ ਰੱਖਿਆ ਜਾਂਦਾ ਹੈ।

```
int a=10;
int *p;
p=&a;
printf("Address in p is %u",p);
```

Above we see, that first we make an int variable a and put 10 in it. As a is an int, the computer keeps to bytes for a and the address of first byte gets associated with a. If we have to know this address we have to use &a. Then we make pointer p.

ਉੱਪਰ ਅਸੀਂ ਦੇਖਦੇ ਹਾਂ ਕਿ ਪਹਿਲਾਂ ਅਸੀਂ ਇੱਕ int ਵੇਰੀਏਬਲ a ਬਨਾਉਂਦੇ ਹਾਂ ਅਤੇ ਉਸ ਵਿੱਚ ਕੀਮਤ 10 ਰਖਦੇ ਹਾਂ। a ਲਈ ਕੰਪਿਊਟਰ ਮੈਮੋਰੀ ਦੀਆਂ 2 ਬਾਈਟਸ ਰਾਖਵੀਆਂ ਕਰਦਾ ਹੈ ਅਤੇ ਪਹਿਲੀ ਬਾਈਟ ਦਾ ਅਡਰੈਸ a ਨਾਲ ਜੁੜ ਜਾਂਦਾ ਹੈ। ਜੇ ਅਸੀਂ ਇਸ ਨੂੰ ਪ੍ਰਾਪਤ ਕਰਨਾ ਹੈ ਤਾਂ ਸਾਨੂੰ &a ਲਿਖਣਾ ਪਵੇਗਾ। ਫਿਰ ਅਸੀਂ ਪੋਇੰਟਰ p ਬਨਾਉਂਦੇ ਹਾਂ।

```
int *p;
```

A pointer is always made with star (*). int before *p means that only the address of an int can go in this pointer. Then we put the address of a in p.

ਪੋਇੰਟਰ ਹਮੇਸ਼ਾ * ਨਾਲ ਬਨਦਾ ਹੈ। int ਦਾ ਮਤਲਬ ਹੈ ਇਸ ਪੋਇੰਟਰ ਵਿਚ ਕਿਸੇ int ਵੇਰੀਏਬਲ ਦਾ ਅਡਰੈਸ ਜਾਵੇਗਾ। ਫਿਰ ਅਸੀਂ p ਵਿੱਚ a ਦਾ ਅਡਰੈਸ ਪਾਉਂਦੇ ਹਾਂ।

```
p=&a;
```

Address is printed using %u.

ਅਡਰੈਸ ਨੂੰ %u ਨਾਲ ਪਰਿੰਟ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।

```
printf("Address in p is %u",p);
```

If we have to see the value at the address stored in p, then we have to use *p.

ਜੇ ਅਸੀਂ ਦੇਖਣਾ ਹੈ ਕਿ p ਵਿੱਚ ਜੋ ਅਡਰੈਸ ਹੈ ਉਸ ਵਿੱਚ ਕੀ ਕੀਮਤ ਹੈ ਤਾਂ ਸਾਨੂੰ *p ਲਿਖਣਾ ਪਵੇਗਾ।

```
printf("Value at address in p is %d",*p);
```

ਕਾਲ ਬਾਏ ਵੈਲਿਊ ਅਤੇ ਕਾਲ ਬਾਏ ਰੈਫਰੈਂਸ (Call By Value and Call By Reference)

Whenever we call a function, the values of the variables passed goes into its arguments. This is called call by value. By this it is not possible for the function to change the value of the passed variable.

ਜਦੋਂ ਅਸੀਂ ਕਿਸੇ ਫੰਕਸ਼ਨ ਨੂੰ ਕਾਲ ਕਰਦੇ ਹਾਂ ਤਾਂ ਉਸਦੀਆਂ ਆਰਗਿਊਮੈਂਟਸ ਵਿੱਚ ਪਾਸ ਕੀਤੇ ਗਏ ਵੇਰੀਏਬਲਾਂ ਦੀਆਂ ਕੀਮਤਾਂ ਜਾਂਦੀਆ ਹਨ। ਇਸਨੂੰ ਕਾਲ ਬਾਏ ਵੈਲਿਊ ਕਹਿੰਦੇ ਹਨ। ਇਸ ਨਾਲ ਇਹ ਸੰਭਵ ਨਹੀਂ ਕਿ ਅਸੀਂ ਫੰਕਸ਼ਨ ਦੁਆਰਾ, ਪਾਸ ਕੀਤੇ ਗਏ ਵੇਰੀਏਬਲ ਦੀ ਕੀਮਤ ਬਦਲ ਸਕੀਏ।

```
#include<stdio.h>
#include<conio.h>
void main()
{
     void getvalue(int n);
     int a=10;

     clrscr();
     getvalue(a);
     printf("Value in a is %d",a);

     getch();
}
void getvalue(int n)
{
     n=20;
}
```

We call function getvalue() and pass the value of variable a to its argument n. In getvalue() function, we make n as 20. But this does not affect the value of a. Upon printing, the value of a remain 10, not 20.

ਅਸੀਂ a ਦੀ ਕੀਮਤ getvalue() ਨੂੰ ਕਾਲ ਕਰਕੇ n ਵਿਚ ਪਾਸ ਕਰਦੇ ਹਾਂ। getvalue() ਵਿਚ n ਨੂੰ ਅਸੀਂ 20 ਕਰ ਦਿੰਦੇ ਹਾਂ। ਪਰ ਇਸ a ਨਾਲ ਦੀ ਕੀਮਤ ਤੇ ਕੋਈ ਫਰਕ ਨਹੀਂ ਪੈਂਦਾ। ਪਰਿੰਟ ਕਰਨ ਤੇ a ਦੀ ਕੀਮਤ 10 ਹੀ ਰਹਿੰਦੀ ਹੈ, ਨਾ ਕਿ 20।

If we want that getvalue() should be able to change the value of a to 20, then we have to call by reference.

ਜੇ ਅਸੀਂ ਚਾਹੁੰਦੇ ਹਾਂ ਕਿ getvalue() ਫੰਕਸ਼ਨ a ਵਿੱਚ ਕੀਮਤ 20 ਕਰ ਦੇਵੇ ਤਾਂ ਸਾਨੂੰ ਕਾਲ ਬਾਏ ਰੈਫਰੈਂਸ ਕਰਨੀ ਪਵੇਗੀ।

```
#include<stdio.h>
#include<conio.h>
void main()
{
     void getvalue(int *p);
```

```
      int a=10;

      clrscr();
      getvalue(&a);
      printf("Value in a is %d",a);

      getch();
}
void getvalue(int *p)
{
      *p=20;
}
```

Upon printing, the above program will print the value of a as 20.

ਉਪਰੋਕਤ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਪਰਿੰਟ ਕਰਨ ਤੇ a ਦੀ ਕੀਮਤ 20 ਹੋਵੇਗੀ।

Here we pass the address (&a) of a to getvalue(), not the value of a. the address of a goes into pointer p. When we do the following in function:

ਇਥੇ ਅਸੀਂ a ਦੀ ਕੀਮਤ ਨਹੀਂ, ਬਲਕਿ a ਦਾ ਅਡਰੈਸ (&a), getvalue() ਨੂੰ ਪਾਸ ਕਰਦੇ ਹਾਂ, ਜੋ ਕਿ ਪੋਇੰਟਰ p ਵਿੱਚ ਜਾਂਦਾ ਹੈ। ਫੰਕਸ਼ਨ ਵਿੱਚ ਜਦੋਂ ਅਸੀਂ

```
*p=20;
```

it means, put value 20 in the memory location that is pointed by p. Since p is pointing to the address of p, so the value of a becomes 20.

ਕਰਦੇ ਹਾਂ, ਤਾਂ ਇਸ ਦਾ ਮਤਲਬ ਹੈ ਕਿ p ਜਿਸ ਮੈਮੋਰੀ ਨੂੰ ਪੋਇੰਟ ਕਰਦਾ ਹੈ, ਉਥੇ 20 ਪਾ ਦਿਉ। ਕਿਉਂਕਿ p, a ਦੇ ਅਡਰੈਸ ਨੂੰ ਪੋਇੰਟ ਕਰਦਾ ਹੈ ਇਸ ਲਈ a ਵਿੱਚ 20 ਪੈ ਜਾਂਦਾ ਹੈ।

Now you can yourself understand, why we use & with variables in scanf().

ਹੁਣ ਤੁਸੀਂ ਖੁਦ ਸਮਝ ਸਕਦੇ ਹੋ ਕਿ scanf() ਵਿੱਚ ਅਸੀਂ ਵੇਰੀਏਬਲਾਂ ਅੱਗੇ & ਕਿਉਂ ਵਰਤਦੇ ਹਾਂ।

By using the above way, we we can make the function to put results in as many variables as we want. Though a function can return only one value, but by the above way, whenever we want to get more than one returns from a function, pointers are helpful. The number of returns we want, we pass addresses of that many variables to the function and the function by using pointers, put the results into theses variables. This is what scanf() does.

ਉਪਰੋਕਤ ਤਰੀਕੇ ਨਾਲ ਅਸੀਂ ਇੱਕ ਫੰਕਸ਼ਨ ਤੋਂ ਜਿੰਨੇ ਮਰਜ਼ੀ ਵੇਰੀਏਬਲਾਂ ਵਿੱਚ ਕੀਮਤਾਂ ਪਵਾ ਸਕਦੇ ਹਾਂ। ਇਸ ਤਰਾਂ ਜਦਕਿ ਫੰਕਸ਼ਨ ਦੀ ਰਿਟੱਰਨ ਇੱਕ ਹੀ ਹੋ ਸਕਦੀ ਹੈ ਤਾਂ ਉਪਰੋਕਤ ਤਰੀਕੇ ਨਾਲ ਜਦੋਂ ਅਸੀਂ ਚਾਹੁੰਦੇ ਹਾਂ ਕਿ ਫੰਕਸ਼ਨ ਤੋਂ ਸਾਨੂੰ ਇੱਕ ਤੋਂ ਜਿਆਦਾ ਰਿਟੱਰਨਸ ਮਿਲਣ ਤਾਂ ਪੋਇੰਟਰ ਸਹਾਈ ਹੁੰਦੇ ਹਨ। ਅਸੀਂ ਜਿਨੀਆਂ ਰਿਟੱਰਨਸ ਚਾਹੁੰਦੇ ਹਾਂ, ਉਹਨੇ ਵੇਰੀਏਬਲਾਂ ਦੇ ਅਡਰੈਸ ਫੰਕਸ਼ਨ ਨੂੰ ਭੇਜ ਦਿੰਦੇ ਹਾਂ ਤੇ ਫੰਕਸ਼ਨ ਪੋਇੰਟਰ ਵਰਤ ਕੇ ਇਹਨਾਂ ਵਿੱਚ ਨਤੀਜੇ ਪਾ ਦਿੰਦਾ ਹੈ। ਇਹੀ ਤਰੀਕਾ scanf() ਵਰਤਦਾ ਹੈ।

ਸਟਰਕਚਰਸ (Structures)

When we have to make a group of variables of different type, we make structures.

ਜਦੋਂ ਅਸੀਂ ਕਿਸੇ ਵੇਰੀਏਬਲਾਂ ਦਾ ਸਮੂਹ ਬਨਾਓਣਾ ਹੋਵੇ ਤਾਂ ਅਸੀਂ ਸਟਰਕਚਰ ਵਰਤਦੇ ਹਾਂ।

```
#include<stdio.h>
#include<conio.h>
struct student
{
     int roll;
     float marks;
     char grade;
};
void main()
{
     struct student st;

     clrscr();
     st.roll=101;
     st.marks=75.4;
     st.grade='A';

     printf("Student Information\n");
     printf("%d, %.2f, %c", st.roll, st.marks, st.grade);

     getch();
}
```

We have made a structure whose name is student. It will have three variables, roll, marks and grade.

ਅਸੀਂ ਇੱਕ ਸਟਰਕਚਰ ਬਨਾਇਆ ਹੈ ਜਿਸਦਾ ਨਾਮ student ਹੈ। ਇਸ ਵਿੱਚ 3 ਤਰਾਂ ਦੇ ਵੇਰੀਏਬਲ ਬਣ ਸਕਦੇ ਹਨ: roll, marks ਅਤੇ grade.

In main() we have made variable st of structure student. Then we put the values in all the three variables roll, name and grade of st. For this we have to use dot (.) like st.roll, st.marks, st.grade. Then we print these values of st. %.2f means that there can will be only 2 digits after dot.

main() ਵਿੱਚ ਅਸੀਂ student ਸਟਰਕਚਰ ਦਾ ਇੱਕ ਵੇਰੀਏਬਲ st ਬਨਾਉਂਦੇ ਹਾਂ। ਫਿਰ st ਦੇ ਤਿੰਨੇ ਵੇਰੀਏਬਲਾਂ roll, marks ਅਤੇ grade ਵਿੱਚ ਕੀਮਤਾਂ ਪਾਉਂਦੇ ਹਾਂ। ਇਸ ਲਈ st ਨਾਲ ਸਾਨੂੰ ਬਿੰਦੂ (.) ਦੀ ਵਰਤੋਂ ਕਰਨੀ ਪੈਂਦੀ ਹੈ ਜਿਵੇਂ st.roll, st.marks ਅਤੇ st.grade । ਫਿਰ ਅਸੀਂ st ਦੀਆਂ ਇਹ ਕੀਮਤਾਂ ਪਰਿੰਟ ਕਰਦੇ ਹਾਂ। %.2f ਦਾ ਮਤਲਬ ਹੈ ਕਿ ਦਸ਼ਮਲਵ (ਬਿੰਦੂ) ਤੋਂ ਬਾਅਦ ਦੋ ਹੀ ਅੰਕ ਦਿਖਾਉਣੇ ਹਨ।

If needed, we can also make an array of structure. Like

ਲੋੜ ਪੈਣ ਤੇ ਅਸੀਂ ਸਟਰਕਚਰ ਦਾ ਐਰੇ ਵੀ ਬਨਾ ਸਕਦੇ ਹਾਂ। ਜਿਵੇਂ:

```
struct student studs[10];
```

Here studs is an array of 10 student. Suppose we have to set the data for 5th student, then we have to do something like below:

ਇੱਥੇ studs ਦਸ student ਦਾ ਐਰੇ ਹੈ। ਮੰਨ ਲਵੋ ਅਸੀਂ ਪੰਜਵੇਂ student ਦੀ ਜਾਨਕਾਰੀ ਸੈਟ ਕਰਨੀ ਹੈ ਤਾਂ ਸਾਨੂੰ ਕੁਝ ਇੰਝ ਕਰਨਾ ਪਵੇਗਾ।

```
studs[4].roll=7;
studs[4].marks=68.7;
studs[4].grade='B';
```

We can also put the address of a structure in pointer. Then to get or set the values of its variables we have to use ->

ਸਟਰਕਚਰ ਨੂੰ ਅਸੀਂ ਪੋਇੰਟਰ ਵਿੱਚ ਵੀ ਪਾ ਸਕਦੇ ਹਾਂ। ਫਿਰ ਇਸਦੇ ਵੇਰੀਏਬਲਾਂ ਵਿੱਚ ਕੀਮਤਾਂ ਪਾਉਣ ਲਈ ਜਾਂ ਕੀਮਤਾਂ ਦੇਖਣ ਲਈ ਸਾਨੂੰ -> ਦੀ ਵਰਤੋਂ ਕਰਨੀ ਪਵੇਗੀ।

```
struct student st;
struct student *p;

p=&st;

p->roll=10;
p->marks=88.5;
p->grade='A';
```

We see, that in array we have same type of variables, but in structure we can have variables of different type.

ਅਸੀਂ ਦੇਖਦੇ ਹਾਂ ਕਿ ਜਦੋਂ ਕਿ ਐਰੇ ਵਿੱਚ ਇੱਕ ਹੀ ਤਰਾਂ ਦੇ ਵੇਰੀਏਬਲ ਹੁੰਦੇ ਹਨ, ਸਟਰਕਚਰ ਵਿੱਚ ਅਸੀਂ ਵੱਖ-ਵੱਖ ਤਰਾਂ ਦੇ ਵੇਰੀਏਬਲ ਰੱਖ ਸਕਦੇ ਹਾਂ।

ਫਾਈਲਾਂ (Files)

We can make C programs to create files on hard-disk and write or read data into these files.

ਅਸੀਂ ਸੀ ਦੇ ਪ੍ਰੋਗਰਾਮ ਨਾਲ ਹਾਰਡ-ਡਿਸਕ ਤੇ ਫਾਈਲਾਂ ਬਨਾ ਸਕਦੇ ਹਾਂ ਅਤੇ ਫਾਈਲਾਂ ਵਿੱਚ ਜਾਨਕਾਰੀ ਲਿਖ ਵੀ ਸਕਦੇ ਹਾਂ ਅਤੇ ਪੜ ਵੀ ਸਕਦੇ ਹਾਂ।

Now we will make a program that will get the information from us and save it in a file on hard-disk.

ਹੁਣ ਅਸੀਂ ਪ੍ਰੋਗਰਾਮ ਬਨਾਉਂਦੇ ਹਾਂ ਜੋ ਸਾਥੋਂ ਜਾਨਕਾਰੀ ਲੈ ਕੇ ਹਾਰਡ-ਡਿਸਕ ਤੇ ਫਾਈਲ ਵਿੱਚ ਸੇਵ ਕਰੇਗਾ।

```
#include<stdio.h>
#include<conio.h>
struct person
{
	char name[40];
	int age;
	float sal;
};
void main()
{
	struct person per;
	FILE *fp;
	char ch;

	clrscr();
	fp=fopen("per.txt", "w");
	if(fp==NULL)
	{
		puts("Error opening file.");
		getch();
		exit();
	}

	do
	{
		printf("\nEnter name, age and salary\n");
		scanf("%s %d %f", &per.name, &per.age, &per.sal);

		fprintf(fp, "%s %d %.2f\n", per.name, per.age, per.sal);

		fflush(stdin);
		printf("Continue: ");
		ch=getche();

	}while(ch=='Y' || ch=='y');
```

```
    fclose(fp);
}
```

Our program keeps on getting data from the use through a do-while loop into structure variable per and keeps on writing it to file per.txt using fprintf() function. %.2f means that there can be only 2 digits after the decimal point.

ਸਾਡਾ ਤਰੀਕਾ ਹੈ ਕਿ ਅਸੀਂ ਇੱਕ do-while ਲੂਪ ਨਾਲ ਯੂਸਰ (ਵਰਤੋਂਕਰਤਾ) ਤੋਂ per ਸਟਰਕਚਰ ਵਿੱਚ ਜਾਨਕਾਰੀ ਲਈ ਜਾਵਾਂਗੇ ਅਤੇ ਇਸਨੂੰ fprintf() ਨਾਲ per.txt ਫਾਈਲ ਵਿੱਚ ਲਿਖਦੇ ਜਾਵਾਂਗੇ। %.2f ਦਾ ਮਤਲਬ ਹੈ ਕਿ ਦਸ਼ਮਲਵ (ਬਿੰਦੂ) ਤੋਂ ਬਾਅਦ ਦੋ ਹੀ ਅੰਕ ਦਿਖਾਉਣੇ ਹਨ।

Whenever we have to read or write into a file on hard-disk, we have to use special FILE pointer.

ਜਦੋਂ ਵੀ ਅਸੀਂ ਹਾਰਡ-ਡਿਸਕ ਤੋਂ ਫਾਈਲ ਪੜਨੀ ਹੋਵੇ, ਜਾਂ ਫਾਈਲ ਵਿੱਚ ਲਿਖਣਾ ਹੋਵੇ ਤਾਂ ਸਾਨੂੰ FILE ਪੋਇੰਟਰ ਵਰਤਣਾ ਪੈਂਦਾ ਹੈ।

We make FILE pointer fp as below:

ਅਸੀਂ FILE ਪੋਂਇਟਰ fp

```
FILE *fp;
```

ਬਨਾਉਂਦੇ ਹਾਂ।

Then we make file per.txt on hard-disk using fopen().

ਫਿਰ ਅਸੀਂ fopen() ਨਾਲ ਹਾਰਡ-ਡਿਸਕ ਤੇ per.txt ਨਾਮ ਦੀ ਫਾਈਲ ਬਨਾਉਂਦੇ ਹਾਂ।

```
fp=fopen("per.txt", "w");
```

fopen() opens a file. “w” means write, meaning we have to write in file. With “w” always a new file is created. If per.txt file was there previously, it will be erased.

fopen() ਫੰਕਸ਼ਨ ਫਾਈਲ ਖੋਲਦਾ ਹੈ। “w” ਦਾ ਮਤਲਬ ਹੈ (write) , ਮਤਲਬ ਅਸੀਂ ਫਾਈਲ ਵਿੱਚ ਲਿਖਣਾ ਹੈ।“w” ਨਾਲ ਹਮੇਸ਼ਾ ਨਵੀਂ ਫਾਈਲ ਬਣਦੀ ਹੈ। ਜੇ ਪਹਿਲਾਂ per.txt ਨਾਮ ਦੀ ਫਾਈਲ ਹੋਵੇਗੀ, ਤਾਂ ਉਹ ਮਿਟ ਜਾਵੇਗੀ।

If for some reason per.txt cannot be opened, fp will have NULL. In that case, we print an error message and end the program.

ਜੇ ਕਿਸੇ ਕਾਰਨ per.txt ਨਹੀਂ ਖੁਲਦੀ ਤਾਂ fp ਵਿੱਚ NULL ਹੋਵੇਗਾ। ਇਸ ਕੇਸ ਵਿੱਚ ਅਸੀਂ error ਦਾ ਮੈਸੇਜ ਦੇ ਕੇ ਪ੍ਰੋਗਰਾਮ ਖਤਮ ਕਰ ਦਿੰਦੇ ਹਾਂ।

If the file opens, we start do-while loop. Data is scanned in per structure.

ਜੇ ਫਾਈਲ ਠੀਕ-ਠਾਕ ਖੁੱਲ ਜਾਂਦੀ ਹੈ ਤਾਂ ਅਸੀਂ do-while ਲੂਪ ਸ਼ੁਰੂ ਕਰਦੇ ਹਾਂ। per ਸਟਰਕਚਰ ਵਿੱਚ ਜਾਨਕਾਰੀ ਸਕੈਨ ਕਰਦੇ ਹਾਂ।

fprintf() writes this data in per.txt using fp pointer. '\n' is for new line.

fprintf() ਇਹ ਜਾਨਕਾਰੀ fp ਨਾਲ per.txt ਵਿੱਚ ਲਿਖਦਾ ਹੈ। '\n' ਨਵੀਂ ਲਾਈਨ ਤੇ ਜਾਣ ਲਈ ਹੈ।

Then we ask from the use if he wants to continue entering the data. If user presses Y or y, then the loop keeps on going and again data is scanned and entered into the file.

ਫਿਰ ਅਸੀਂ ਯੂਸਰ (ਵਰਤੋਂਕਰਤਾ) ਤੋਂ ਪੁਛਦੇ ਹਾਂ ਕਿ ਹੋਰ ਜਾਨਕਾਰੀ ਐਂਟਰ ਕਰਨੀ ਹੈ ਕਿ ਨਹੀਂ। ਜੇ ਯੂਸਰ Y ਜਾਂ y ਪਾਉਂਦਾ ਹੈ ਤਾਂ ਲੂਪ ਚਲਦੀ ਰਹਿੰਦੀ ਹੈ ਅਤੇ ਫਿਰ ਤੋਂ ਜਾਨਕਾਰੀ ਐਂਟਰ ਕਰਕੇ ਫਾਈਲ ਵਿੱਚ ਲਿਖੀ ਜਾਂਦੀ ਹੈ।

Whenever we enter data, we press ENTER key at the end. We use fflush(stdin) to remove its code. Otherwise the code of ENTER gets assigned to the next variable, that is name variable of per structure.

ਜਦ ਵੀ ਅਸੀਂ ਜਾਨਕਾਰੀ ਐਂਟਰ ਕਰਦੇ ਹਾਂ ਤਾਂ ਅਖੀਰ ਤੇ ENTER ਕੀ ਦਬਦੇ ਹਾਂ, ਜਿਸਦਾ ਕੋਡ ਕੱਢਣ ਲਈ fflush(stdin) ਵਰਤਦੇ ਹਾਂ। ਨਹੀ ਤਾਂ ਇਹ ਕੋਡ ਅਗਲੇ ਵੇਰੀਏਬਲ (per ਦਾ name)ਨੂੰ ਚਲਾ ਜਾਵੇਗਾ।

After the loop terminates, fclose(fp) saves and closes the file.

ਲੂਪ ਖਤਮ ਹੋਣ ਤੋਂ ਬਾਅਦ fclose(fp) ਫਾਈਲ ਨੂੰ ਸੇਵ ਕਰਕੇ ਬੰਦ ਕਰ ਦਿੰਦਾ ਹੈ।

If you see on hard disk, you will see file per.txt created in C:\TurboC++\Disk\TurboC3\BIN and it will have all the information that you have entered. You can use Notepad to open it. Remember that if you run the program again, new per.txt will be created.

ਤੁਸੀਂ ਹਾਰਡ-ਡਿਸਕ ਤੇ ਦੇਖੋਗੇ ਕਿ C:\TurboC++\Disk\TurboC3\BIN ਫੋਲਡਰ ਵਿੱਚ per.txt ਨਾਮ ਦੀ ਫਾਈਲ ਬਣੀ ਹੋਵੇਗੀ ਅਤੇ ਇਸ ਵਿੱਚ ਸਾਰੀ ਜਾਨਕਾਰੀ ਜੋ ਤੁਸੀਂ ਐਂਟਰ ਕੀਤੀ ਸੀ, ਹੋਵੇਗੀ। ਤੁਸੀਂ ਨੋਟਪੈਡ ਨਾਲ ਇਸ ਨੂੰ ਦੇਖ ਸਕਦੇ ਹੋ। ਯਾਦ ਰੱਖੋ ਕਿ ਦੁਬਾਰਾ ਪ੍ਰੋਗਰਾਮ ਚਲਾਉਣ ਤੇ ਨਵੀਂ per.txt ਬਣੇਗੀ।
If we have to read from a file, then we have to use "r" instead of "w".

ਫਾਈਲ ਵਿੱਚੋਂ ਜਾਨਕਾਰੀ ਪੜਨੀ ਹੋਵੇ ਤਾਂ "w" ਦੀ ਜਗਾ "r" ਵਰਤਣਾ ਪਵੇਗਾ।

```
fp=fopen("per.txt", "r");
```

If we have to read all the records from per.txt and print on screen, then we have to do like below.

ਹੁਣ ਅਸੀਂ ਸਾਰੇ ਰਿਕਾਰਡ per.txt ਚੋਂ ਪੜ ਕੇ ਸਕਰੀਨ ਤੇ ਪਰਿੰਟ ਕਰਨੇ ਹਨ ਤਾਂ ਹੇਠ ਲਿਖੇ ਅਨੁਸਾਰ ਕਰਨਾ ਪਵੇਗਾ।

```
while(fscanf(fp,"%s %d %f", &per.name,&per.age,&per.sal) != EOF)
{
     printf("%s %d %.2f", per.name, per.age, per.sal);
}
```

We use fscanf() to read from a file. Using while loop, fscanf() keeps on reading from file until EOF comes. EOF means End of File.

ਫਾਈਲ ਚੋਂ ਪੜਨ ਲਈ fscanf() ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ। while ਲੂਪ ਨਾਲ fscanf() ਉਦੋਂ ਤੱਕ ਫਾਈਲ ਚੋਂ ਪੜਦਾ ਰਹੇਗਾ ਜਦੋਂ ਤੱਕ EOF ਨਹੀਂ ਆਉਂਦਾ। EOF ਦਾ ਮਤਲਬ ਹੈ End Of File (ਫਾਈਲ ਦਾ ਅੰਤ)।

Side by side you keep on printing the data on screen using printf(). By %.2f we mean that there will be only 2 digits after decimal point.

ਤੁਸੀਂ ਨਾਲ ਦੀ ਨਾਲ ਹੀ ਜਾਨਕਾਰੀ printf() ਨਾਲ ਸਕਰੀਨ ਤੇ ਪਰਿੰਟ ਕਰੀ ਜਾਂਦੇ ਹੋ। %.2f ਦਾ ਮਤਲਬ ਹੈ ਕਿ ਦਸ਼ਮਲਵ (ਬਿੰਦੂ) ਤੋਂ ਬਾਅਦ ਦੋ ਹੀ ਅੰਕ ਦਿਖਾਉਣੇ ਹਨ।

We can also read from a file character by character. Suppose we have to make copy of a file. Then you can do as below.

ਅਸੀਂ ਫਾਈਲ ਚੋਂ ਇੱਕ-ਇੱਕ ਕਰੈਕਟਰ ਵੀ ਪੜ ਸਕਦੇ ਹਾਂ। ਮੰਨ ਲਵੋ ਅਸੀਂ ਇੱਕ ਫਾਈਲ ਦੀ ਕਾਪੀ ਬਨਾਉਣੀ ਹੈ ਤਾਂ ਅਸੀਂ ਹੇਠ ਲਿਖੇ ਅਨੁਸਾਰ ਕਰ ਸਕਦੇ ਹਾਂ।

```
while(1)
{
      ch=fgetc(f1);
      if(ch==EOF)
      {
            break;
      }
      else
      {
            fputc(ch, f2);
      }
}
fclose(f1);
fclose(f2);
```

The above code uses reads a character at a time into ch from file pointed by f1. If EOF comes in ch, it means the file has ended. Therefore we break the loop. Otherwise we write the value of ch into file pointed by f2.

ਉਪਰੋਕਤ ਕੋਡ ਫਾਈਲ ਪੋਇੰਟਰ f1 ਵਰਤ ਕੇ ਇਸ ਵਿੱਚੋਂ ਇੱਕ-ਇੱਕ ਕਰੈਕਟਰ ch ਵਿੱਚ ਲਿਆਉਂਦੇ ਹਾਂ। ਜੇ ch ਵਿੱਚ EOF ਆਉਂਦਾ ਹੈ ਤਾਂ ਸਮਝੋ ਫਾਈਲ ਦਾ ਅਖੀਰ ਹੋ ਗਿਆ। ਇਸ ਲਈ break ਨਾਲ ਲੂਪ ਖਤਮ ਕਰ ਦੇਵੋ। ਨਹੀਂ ਤਾਂ ch ਨੂੰ f2 ਫਾਈਲ ਵਿੱਚ ਲਿਖਦੇ ਜਾਓ।

We can use fputs() to write strings and fgets() to read strings().

ਸਟਰਿੰਗਸ ਲਿਖਣ ਲਈ ਅਸੀਂ fputs() ਅਤੇ ਪੜਨ ਲਈ fgets() ਵਰਤ ਸਕਦੇ ਹਾਂ।

```
char name[]="Bhupinder";
fputs(name, fp);
```

The above code will write Bhupinder in the file pointed by fp.

ਉਪਰੋਕਤ ਕੋਡ fp ਪੋਇੰਟਰ ਵਾਲੀ ਫਾਈਲ ਵਿੱਚ Bhupinder ਲਿਖ ਦੇਵੇਗਾ।

```
while(fgets(line, 79, fp) != NULL)
{
     puts(line);
}
```

The above code reads a string at a time into line. At a time, a maximum of 79 characters can come in string line. At the end, fgets() return NULL and the loop ends. puts() prints line on screen.

ਉਪਰੋਕਤ ਕੋਡ ਫਾਈਲ ਚੋਂ ਇੱਕ-ਇੱਕ ਸਟਰਿੰਗ line ਵਿੱਚ ਲਿਆਉਂਦਾ ਹੈ। ਇੱਕ ਸਮੇ ਤੇ ਵੱਧ ਤੋਂ ਵੱਧ 79 ਕਰੈਕਟਰ line ਸਟਰਿੰਗ ਵਿੱਚ ਆਉਣਗੇ। ਅਖੀਰ ਵਿੱਚ fgets(), NULL ਰਿਟੱਰਨ ਕਰਦਾ ਹੈ ਅਤੇ ਲੂਪ ਖਤਮ ਹੋ ਜਾਵੇਗੀ। puts(), line ਨੂੰ ਸਕਰੀਨ ਤੇ ਪਰਿੰਟ ਕਰਦਾ ਹੈ।

ਰੈਂਡਮ ਅਕਸੈਸ ਫਾਈਲਸ (Random Access Files)

When we have to write records of same size in a file, then it is best to open the file in binary mode and do random access. Random Access means that we can to go any record at any time. Binary mode means that raw data will be read or written. We don't convert data into readable characters as fprintf(), fscanf() does using %d, %f while writing or reading into file.

ਜਦੋਂ ਅਸੀਂ ਫਾਈਲ ਵਿੱਚ ਰਿਕਾਰਡ ਲਿਖਣੇ ਹੋਣ, ਜਿਨਾਂ ਦਾ ਸਾਈਜ਼ ਇੱਕੋ ਜਿਹਾ ਹੋਵੇ ਤਾਂ ਫਾਈਲ ਨੂੰ ਬਾਈਨਰੀ ਮੋੜ ਵਿੱਚ ਖੋਲ ਕੇ ਰੈਂਡਮ ਅਕਸੈਸ ਕਰ ਸਕਣਾ ਠੀਕ ਰਹਿੰਦਾ ਹੈ। ਰੈਂਡਮ ਅਕਸੈਸ ਦਾ ਮਤਲਬ ਹੈ ਕਿ ਕਿਸੇ ਵਕਤ ਵੀ ਕਿਸੇ ਰਿਕਾਰਡ ਤੇ ਜਾ ਸਕਦੇ ਹੋ। ਬਾਈਨਰੀ ਮੋੜ ਦਾ ਮਤਲਬ ਹੈ ਕਿ ਕੱਚਾ ਡੇਟਾ (bytes) ਪੜਿਆ, ਲਿਖਿਆ ਜਾਂਦਾ ਹੈ। ਅਸੀਂ ਡੇਟੇ ਨੂੰ ਪੜਨਯੋਗ ਕਰੈਕਟਰਾਂ ਵਿੱਚ ਨਹੀਂ ਬਦਲਦੇ, ਜੋ ਕਿ fprintf(), fscanf() ਵਗੈਰਾ ਫੰਕਸ਼ਨ %d, %f ਵਗੈਰਾ ਲਾ ਕੇ ਕਰਦੇ ਹਨ।

In binary mode, we use fwrite() for writing and fread() for reading from file. To go forward or backward in records we use fseek(). If we to see the position of FILE pointer, we use ftell. To come to the beginning of file, we use rewind().

ਬਾਈਨਰੀ ਮੋੜ ਵਿੱਚ ਲਿਖਣ ਲਈ fwrite() ਅਤੇ ਪੜਨ ਲਈ fread() ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ। ਰਿਕਾਰਡਾਂ ਵਿੱਚ ਅੱਗੇ ਪਿੱਛੇ ਜਾਣ ਲਈ fseek() ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ। ਦੇਖਣਾ ਹੋਵੇ ਕਿ ਇਸ ਵਕਤ ਪੋਇੰਟਰ ਕਿਸ ਜਗਾ ਤੇ ਹੈ ਤਾਂ ftell() ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ। ਪੋਇੰਟਰ ਨੂੰ ਸ਼ੁਰੂਆਤ ਤੇ ਲਿਆਉਣ ਲਈ rewind() ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ।

To open a file for writing in binary mode.
ਫਾਈਲ ਲਿਖਣ ਲਈ ਖੋਲਣ ਲਈ

```
fp=fopen("per.dat", "wb");
```

To open a file for reading in binary mode.
ਫਾਈਲ ਪੜਨ ਲਈ ਖੋਲਣ ਲਈ

```
fp=fopen("per.dat", "rb");
```

For both writing and reading.
ਲਿਖਣ ਪੜਨ ਦੋਨਾਂ ਲਈ

```
fp=fopen("per.dat", "rb+");
```

We use fwrite to write into a binary file.
ਫਾਈਲ ਵਿੱਚ ਲਿਖਣਾ ਹੋਵੇ ਤਾਂ fwrite() ਵਰਤਦੇ ਹਾਂ।

```
fwrite(&per, sizeof(per), 1, fp);
```

The above code write the data of per structure into file pointed by fp. sizeof() tells the size of per structure in bytes. 1 means that at a time only one per structure has to be written. We can also write an array of structures at a time. Then we have to give the size of array instead of 1.

ਉਪਰੋਕਤ ਕੋਡ per ਸਟਰਕਚਰ ਦਾ ਡੇਟਾ fp ਵਿੱਚ ਲਿਖਦਾ ਹੈ। sizeof() ਨਾਲ per ਦਾ ਸਾਈਜ਼ (ਕੰਨੇ ਬਾਈਟਸ ਮੈਮੋਰੀ ਲਈ ਹੈ) ਪਤਾ ਲਗਦੀ ਹੈ। 1 ਦਾ ਮਤਲਬ ਹੈ ਕਿ ਇੱਕ ਵਕਤ 1 ਹੀ per ਸਟਰਕਚਰ fp ਵਿੱਚ ਲਿਖਣਾ ਹੈ। ਅਸੀਂ ਇੱਕੋ ਸਮੇਂ ਸਟਰਕਚਰਾਂ ਦਾ ਐਰੇ ਵੀ ਲਿੱਖ ਸਕਦੇ ਹਾਂ। ਉਸ ਕੇਸ ਵਿੱਚ 1 ਦੀ ਥਾਂ ਐਰੇ ਦਾ ਸਾਈਜ਼ ਲਿਖਿਆ ਜਾਵੇਗਾ।

We use fread() to read from a binary file.
ਫਾਈਲ ਵਿੱਚੋਂ ਪੜਨਾ ਹੋਵੇ ਤਾਂ fread() ਵਰਤਦੇ ਹਾਂ।

```
fread(&per, sizeof(per), 1, fp);
```

The above code reads 1 record at a time from file fp into per structure. sizeof() tells the size of per structure in bytes. 1 means that at a time only one record has to be read into per structure.

ਉਪਰੋਕਤ ਕੋਡ fp ਵਿਚੋਂ 1 ਰਿਕਾਰਡ ਪੜ ਕੇ per ਸਟਰਕਚਰ ਵਿੱਚ ਪਾਉਂਦਾ ਹੈ। sizeof() ਨਾਲ per ਦਾ ਸਾਈਜ਼ (ਕੰਨੇ ਬਾਈਟਸ ਮੈਮੋਰੀ ਲਈ ਹੈ) ਪਤਾ ਲਗਦੀ ਹੈ। 1 ਦਾ ਮਤਲਬ ਹੈ ਕਿ ਇੱਕ ਵਕਤ 1 ਹੀ per ਸਟਰਕਚਰ fp ਵਿੱਚੋਂ ਪੜਨਾ ਹੈ।

If we have to read and print all the records, we do as below.

ਜੇ ਅਸੀਂ ਸਾਰੇ ਰਿਕਾਰਡ ਪੜ ਕੇ ਪਰਿੰਟ ਕਰਨੇ ਹੋਣ ਤਾਂ ਅਸੀਂ ਹੇਠ ਲਿਖੇ ਅਨੁਸਾਰ ਕਰਦੇ ਹਾਂ।

```
while(fwrite(&per, sizeof(per), 1, fp)==1)
{
     printf("%s %d %.2f\n", per.name, per.age, per.sal);
}
```

We have asked fwrite to read 1 record at a time. Therefore till records are being read, 1 is returned after each read. At the end 0 is returned and the loop ends.

ਕਿਉਂਕਿ ਅਸੀਂ ਇੱਕ ਸਮੇਂ fwrite() ਨੂੰ 1 ਹੀ ਸਟਰਕਚਰ ਪੜਨ ਲਈ ਕਿਹਾ ਹੈ, ਇਸ ਲਈ ਜਦ ਤੱਕ ਰਿਕਾਰਡ ਪੜੇ ਜਾਣਗੇ, 1 ਰਿਟੱਰਨ ਹੁੰਦਾ ਰਹੇਗਾ। ਅਖੀਰ ਵਿੱਚ 0 ਰਿਟੱਰਨ ਹੋਵੇਗਾ ਅਤੇ ਲੂਪ ਖਤਮ ਹੋ ਜਾਵੇਗੀ।

We can also use “a” to open a file, which means append, meaning we have to add data at the end of previous data. If the file exists, then no new file is to be created, instead the new data has to be appended at its end.

ਅਸੀਂ ਫਾਈਲ ਖੋਲਣ ਸਮੇਂ “a” ਮੋੜ ਵੀ ਵਰਤ ਸਕਦੇ ਹਾਂ, ਜਿਸਦਾ ਮਤਲਬ ਹੈ append, ਮਤਲਬ ਫਾਈਲ ਦੇ ਪੁਰਾਣੇ ਡੇਟੇ ਦੇ ਪਿੱਛੇ ਜੋੜਨਾ। ਜੇ ਫਾਈਲ ਹੈ ਤਾਂ ਨਵੀਂ ਫਾਈਲ ਨਹੀਂ ਬਨਾਉਣੀ, ਬਲਕਿ ਉਸਦੇ ਪਿੱਛੇ ਹੀ ਡੇਟਾ ਜੋੜ ਦੇਣਾ ਹੈ।

Remember that “r” always opens an existing file, “w” always opens a new file and “a” will only open a new file, if the file does not exist.

ਯਾਦ ਰੱਖੋ ਕਿ “r” ਹਮੇਸ਼ਾ ਪੁਰਾਣੀ ਫਾਈਲ ਹੀ ਖੋਲੇਗਾ, “w” ਹਮੇਸ਼ਾ ਨਵੀਂ ਫਾਈਲ ਹੀ ਬਣਾਵੇਗਾ ਅਤੇ “a” ਜੇ ਫਾਈਲ ਨਹੀਂ ਹੈ, ਤਾਂ ਹੀ ਨਵੀਂ ਬਨਾਵੇਗਾ।

Below is a simple database program through which you can enter, list, modify and delete records. This program has been modified for simplicity from Let Us C Book by Shri Yashwant Kanetkar.

ਹੇਠਾਂ ਅਸੀਂ ਡੇਟਾਬੇਸ ਬਣਾਉਣ ਦਾ ਪਰੋਗਰਾਮ ਦੇ ਰਹੇ ਹਾਂ ਜਿਸ ਨਾਲ ਕਿ ਤੁਸੀਂ ਕੋਈ ਰਿਕਾਰਡ ਸੇਵ ਕਰ ਸਕਦੇ ਹੋ, ਸਾਰੇ ਰਿਕਾਰਡ ਦਿਖਾ ਸਕਦੇ ਹੋ, ਕਿਸੇ ਰਿਕਾਰਡ ਨੂੰ ਲੱਭ ਕੇ ਬਦਲ ਸਕਦੇ ਹੋ ਅਤੇ ਕਿਸੇ ਰਿਕਾਰਡ ਨੂੰ ਹਟਾ ਸਕਦੇ ਹੋ। ਇਹ ਪਰੋਗਰਾਮ ਸ਼੍ਰੀ ਯਸ਼ਵੰਤ ਕਨੇਤਕਰ ਦੀ ਕਿਤਾਬ “ਲੈਟ ਅਸ ਸੀ” ਤੋਂ ਧੰਨਵਾਦ ਸਹਿਤ ਬਦਲਾਵ ਸਹਿਤ ਲਿਆ ਗਿਆ ਹੈ।

```
#include <stdio.h>
void main()
{
    FILE* fp, *ft;
    int choice;
    struct emp
    {
        char name[40];
        int age;
        float bs;
    };
    struct emp e;
    char empname[40];
    long int recsize;

    fp=fopen("EMP.DAT","rb+");
    if(fp==NULL)
    {
        fp=fopen("EMP.DAT","wb+");
        if(fp==NULL)
        {
            puts("Cannot open file");
            exit(-1);
        }
    }
    recsize=sizeof(e);
    while(1)
    {
        system("cls");
        printf("1. Add Records\n");
        printf("2. List Records\n");
        printf("3. Modify Records\n");
        printf("4. Delete Records\n");
        printf("5. Exit\n");
        printf("Your choice: ");
        fflush(stdin);
        scanf("%d",&choice);
        fflush(stdin);
        if(choice==1)
        {
```

```
        fseek(fp,0,SEEK_END);
        printf("Enter Name, age and basic salary : ");
        scanf("%s %d %f",e.name, &e.age, &e.bs);
        fwrite(&e, recsize, 1, fp);
        printf("Record added...");
        getche();
}
else if(choice==2)
{
    rewind(fp);
    while(fread(&e,recsize,1,fp)==1)
    {
        printf("\n%s, %d, %f",e.name,e.age,e.bs);
    }
    getche();
}
else if(choice==3)
{
    printf("\nEnter name of employee to modify: ");
    scanf("%s",empname);
    rewind(fp);
    while(fread(&e,recsize,1,fp)==1)
    {
        if(strcmp(e.name, empname)==0)
        {
            printf("\nEnter new name, age, bs");
            scanf("%s %d %f",e.name,&e.age,&e.bs);
            fseek(fp, -recsize, SEEK_CUR);
            fwrite(&e, recsize, 1, fp);
            printf("Record modified...");
            getche();
            break;
        }
    }
}
else if(choice==4)
{
    printf("\nEnter name of employee to delete: ");
    scanf("%s",empname);
    ft=fopen("TEMP.DAT","wb");
    rewind(fp);
    while(fread(&e,recsize,1,fp)==1)
    {
        if(strcmp(e.name,empname)!=0)
        {
            fwrite(&e, recsize, 1, ft);
        }
    }
    fclose(fp);
    fclose(ft);
```

```
            remove("EMP.DAT");
            rename("TEMP.DAT","EMP.DAT");
            fp=fopen("EMP.DAT","rb+");
            printf("Record deleted...");
            getche();
        }
        else if(choice==5)
        {
            fclose(fp);
            exit(0);
        }
    }
}
```

While(1) means while the condition is true. This loop keeps on running because 1 always means true. We use choice 5 to end this loop.
While(1) ਦਾ ਮਤਲਬ ਹੈ ਕਿ ਇਹ ਲੂਪ ਹਮੇਸ਼ਾ ਚਲਦੀ ਰਹੇਗੀ ਕਿਉਂਕਿ 1 ਦਾ ਮਤਲਬ ਹਮੇਸ਼ਾ ਟਰੂ ਹੁੰਦਾ ਹੈ। ਅਸੀਂ ਚੌਇਸ 5 ਤੇ ਇਸ ਲੂਪ ਨੂੰ ਖਤਮ ਕਰਦੇ ਹਾਂ।

ਵਰਤੇ ਗਏ ਫੰਕਸ਼ਨ (Functions Used)

rewind() -> We use this function to take the file pointer to beginning of file.
ਇਸ ਫੰਕਸ਼ਨ ਨਾਲ ਅਸੀਂ ਫਾਈਲ ਪੋਆਂਇਟਰ ਫਾਈਲ ਦੇ ਸ਼ੁਰੂਆਤ ਤੇ ਲੈ ਜਾ ਸਕਦੇ ਹਾਂ।

fseek() -> We use this function to take file pointer to a particular position
SEEK_END means that the position given is from end of file.
SEEK_CUR means that the position given is from current position of file pointer.

fseek() ਫੰਕਸ਼ਨ ਨਾਲ ਅਸੀਂ ਫਾਈਲ ਪੋਆਂਇਟਰ ਨੂੰ ਕਿਸੇ ਵੀ ਜਗਾ ਤੇ ਲੈ ਜਾ ਸਕਦੇ ਹਾਂ।
SEEK_END ਦਾ ਮਤਲਬ ਹੈ ਕਿ ਅਸੀਂ ਫਾਈਲ ਦੇ ਅਖੀਰ ਦੇ ਹਿਸਾਬ ਨਾਲ ਜਗਾ ਦੱਸ ਰਹੇ ਹਾਂ।
SEEK_CUR ਦਾ ਮਤਲਬ ਹੈ ਕਿ ਅਸੀਂ ਫਾਈਲ ਪੋਇੰਟਰ ਦੀ ਮੌਜੂਦਾ ਜਗਾ ਦੇ ਅਨੁਸਾਰ ਜਗਾ ਦੱਸ ਰਹੇ ਹਾਂ।

remove() -> We use this function to delete a file.
ਇਹ ਫੰਕਸ਼ਨ ਫਾਈਲ ਨੂੰ ਡਲੀਟ ਕਰਨ ਲਈ ਹੈ।

rename()-> We use this function to rename a file.
ਇਹ ਫੰਕਸ਼ਨ ਫਾਈਲ ਦਾ ਨਾਮ ਬਦਲਣ ਲਈ ਹੈ।

ਸਟੋਰੇਜ ਟਾਈਪਸ (Storage Types)

When we make variables, they can be created as any storage type. In C we can have below written storage types.

ਜਦੋਂ ਅਸੀਂ ਵੇਰੀਏਬਲ ਬਨਾਉਂਦੇ ਹਾਂ ਤਾਂ ਇਹਨਾਂ ਨੂੰ ਕਈ ਤਰਾਂ ਦੀ ਸਟੋਰੇਜ਼ ਵਿੱਚ ਰੱਖਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਸੀ ਵਿੱਚ ਹੇਠ ਲਿਖੇ ਸਟੋਰੇਜ ਹੋ ਸਕਦੇ ਹਨ।

auto (private or local)

The variables that we have created till now are auto variables. They are created in the memory of computer and their initial value is some garbage. You can use them only inside the function. When the control ends in the function, they are lost.

ਜਿਹੜੇ ਅਸੀਂ ਹੁਣ ਤੱਕ ਵੇਰੀਏਬਲ ਬਨਾਏ ਹਨ, ਇਹ ਸਾਰੇ auto ਵੇਰੀਏਬਲ ਬਨਦੇ ਹਨ। ਇਹ ਮੈਮੋਰੀ ਵਿੱਚ ਬਨਦੇ ਹਨ ਅਤੇ ਇਹਨਾਂ ਦੀ ਸ਼ੁਰੂਆਤੀ ਕੀਮਤ ਕੋਈ ਵੀ ਕੂੜਾ-ਕਰਕਟ ਹੋ ਸਕਦੀ ਹੈ। ਇਹਨਾਂ ਨੂੰ ਫੰਕਸ਼ਨ ਦੇ ਅੰਦਰ ਹੀ ਵਰਤਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਜਦੋਂ ਫੰਕਸ਼ਨ ਚੋਂ ਕੰਟਰੋਲ ਖਤਮ ਹੁੰਦਾ ਹੈ ਤਾਂ ਇਹ ਵੇਰੀਏਬਲ ਖਤਮ ਹੋ ਜਾਂਦੇ ਹਨ।

register

If we write register before the variables, they get created in CPU registers. Because registers are in processor, they are nearer than memory, so the work gets done very fast. But we cannot make many register variables because processor registers are limited. They can be used in the function itself and gets deleted once the control ends from the function.

ਜੇ ਅਸੀਂ ਵੇਰੀਏਬਲ ਦੇ ਅੱਗੇ register ਲਿਖਦੇ ਹਾਂ, ਤਾਂ ਇਹ ਵੇਰੀਏਬਲ ਪ੍ਰੋਸੈਸਰ ਦੇ ਰਜਿਸਟਰਾਂ ਵਿੱਚ ਬਣਦੇ ਹਨ। ਕਿਉਂਕਿ ਰਜਿਸਟਰ ਮੈਮੋਰੀ ਨਾਲੋਂ ਵੀ ਪ੍ਰੋਸੈਸਰ ਦੇ ਨੇੜੇ ਹੁੰਦੇ ਹਨ, ਇਸ ਲਈ ਇਹਨਾਂ ਵਿੱਚ ਕੰਮ ਬਹੁਤ ਤੇਜ ਹੁੰਦਾ ਹੈ। ਪਰ ਅਸੀਂ ਜਿਆਦਾ ਵੇਰੀਏਬਲ register ਨਹੀਂ ਰੱਖ ਸਕਦੇ ਕਿਉਂਕਿ ਪ੍ਰੋਸੈਸਰ ਦੇ ਰਜਿਸਟਰ ਸੀਮਿਤ ਹੁੰਦੇ ਹਨ। ਇਹ ਵੀ ਫੰਕਸ਼ਨ ਦੇ ਵਿੱਚ ਹੀ ਵਰਤੇ ਜਾ ਸਕਦੇ ਹਨ ਅਤੇ ਕੰਟਰੋਲ ਫੰਕਸ਼ਨ ਚੋਂ ਖਤਮ ਹੋਣ ਤੇ ਖਤਮ ਹੋ ਜਾਂਦੇ ਹਨ।

public (global)

These variables are created outside the functions. Mostly, they are created just below include statements. These variables get created in the memory. They can be used by any function. They remain in memory, until the whole program ends. There starting value is 0.

ਇਹ ਵੇਰੀਏਬਲ ਫੰਕਸ਼ਨਾਂ ਤੋਂ ਬਾਹਰ ਬਣਾਏ ਜਾਂਦੇ ਹਨ। ਆਮ ਤੌਰ ਤੇ ਇਹ include ਦੀਆਂ ਸਟੇਟਮੈਂਟਾਂ ਦੇ ਬਿਲਕੁਲ ਥੱਲੇ ਬਣਾਏ ਜਾਂਦੇ ਹਨ। ਇਹ ਵੇਰੀਏਬਲ ਵੀ ਮੈਮੋਰੀ ਵਿੱਚ ਬਣਦੇ ਹਨ। ਇਹਨਾਂ ਨੂੰ ਕੋਈ ਵੀ ਫੰਕਸ਼ਨ ਵਰਤ ਸਕਦਾ ਹੈ। ਇਹ ਓਦੋਂ ਤੱਕ ਰਹਿੰਦੇ ਹਨ, ਜਦ ਤੱਕ ਸਾਰਾ ਪ੍ਰੋਗਰਾਮ ਖਤਮ ਨਹੀਂ ਹੁੰਦਾ। ਇਹਨਾਂ ਦੀ ਸ਼ੁਰੂਆਤੀ ਕੀਮਤ 0 ਹੁੰਦੀ ਹੈ।

static

These variables are created inside the function. They get created in memory. They can be used only inside the function. There starting value is 0. They end only when the whole program ends. during first function call, they get created and then remain in memory for the life of whole program.

ਇਹ ਵੇਰੀਏਬਲ ਫੰਕਸ਼ਨ ਦੇ ਅੰਦਰ ਬਣਾਏ ਜਾਂਦੇ ਹਨ। ਇਹ ਮੈਮੋਰੀ ਵਿੱਚ ਬਣਦੇ ਹਨ। ਇਹਨਾਂ ਨੂੰ ਫੰਕਸ਼ਨ ਦੇ ਅੰਦਰ ਹੀ ਵਰਤਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਇਹਨਾਂ ਦੀ ਸ਼ੁਰੂਆਤੀ ਕੀਮਤ 0 ਹੁੰਦੀ ਹੈ। ਇਹ ਸਾਰਾ ਪ੍ਰੋਗਰਾਮ ਖਤਮ ਹੋਣ ਤੇ ਹੀ ਖਤਮ ਹੁੰਦੇ

ਹਨ। ਜਦ ਪਹਿਲੀ ਵਾਰ ਫੰਕਸ਼ਨ ਨੂੰ ਕਾਲ ਹੁੰਦੀ ਹੈ, ਤਾਂ ਇਹ ਬਣ ਜਾਂਦੇ ਹਨ ਅਤੇ ਬਾਅਦ ਵਿੱਚ ਸਾਰੇ ਪ੍ਰੋਗਰਾਮ ਤੱਕ ਰਹਿੰਦੇ ਹਨ।

ਪਰੀਪਰੋਸੈਸਰ ਡਾਈਰੈਕਟਿਵਸ (Pre-Processor Directives)

Their best use is in creating constants. Constant means whole value is not changed again in the program.

ਇਸ ਦੀ ਸਭ ਤੋਂ ਮਹੱਤਵਪੂਰਣ ਵਰਤੋਂ ਸਥਿਰ (Constant) ਬਨਾਉਣ ਵਿੱਚ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। Constant ਦਾ ਮਤਲਬ ਹੈ ਜਿਸ ਦੀ ਕੀਮਤ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਦੁਬਾਰਾ ਬਦਲੀ ਨਾ ਜਾ ਸਕੇ।

```
#define PI 3.14
```

Above, we have created a constant PI whose value is 3.14. In the program, wherever we write PI, it will be substituted by 3.14. In the program, by mistake also we cannot change the value of PI. Therefore PI is a constant.

ਉਪਰ ਅਸੀਂ PI ਇੱਕ ਕਾਨਸਟੈਂਟ ਬਨਾਇਆ ਹੈ, ਜਿਸ ਦੀ ਕੀਮਤ 3.14 ਰੱਖੀ ਹੈ। ਅਸੀਂ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਜਿੱਸ ਜਗਾ ਵੀ PI ਲਿਖਾਂਗੇ ਉਸ ਜਗਾ 3.14 ਪੈ ਜਾਵੇਗਾ। ਅਸੀਂ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਗਲਤੀ ਨਾਲ ਵੀ PI ਦੀ ਕੀਮਤ ਵਧਾ-ਘਟਾ ਨਹੀਂ ਸਕਦੇ। ਇਸ ਲਈ PI ਨੂੰ Constant ਕਹਿੰਦੇ ਹਾਂ।

The benefit is that tomorrow, if want to keep the value of PI as 3.1428, then we have to change only in define.

ਇਸ ਦਾ ਫਾਇਦਾ ਇਹ ਹੈ ਕਿ ਕੱਲ ਨੂੰ ਜੇ ਅਸੀਂ PI ਨੂੰ 3.1428 ਰੱਖਣਾ ਹੈ ਤਾਂ ਸਿਰਫ define ਵਿੱਚ ਹੀ ਬਦਲਣਾ ਪਵੇਗਾ।

```
#define PI 3.1428
```

The value will change in whole program.

ਸਾਰੇ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਕੀਮਤ ਬਦਲ ਜਾਵੇਗੀ।