LECTURE NOTE-5

VARIABLES

Variables are names that are used to store values. It can take different values but one at a time. A data type is associated with each variable & it decides what values the variable can take. When you decide your program needs another variable, you simply declare (or define) a new variable and C makes sure you get it. You declare all C variables at the top of whatever blocks of code need them. Variable declaration requires that you inform C of the variable's name and data type. Syntax – datatype variablename;

Eg:

int page_no;

char grade;

float salary;

long y;

➢ **Declaring Variables:**

There are two places where you can declare a variable:

- After the opening brace of a block of code (usually at the top of a function)
- Before a function name (such as before main() in the program) Consider various examples:

  Suppose you had to keep track of a person's first, middle, and last initials. Because an initial is obviously a character, it would be prudent to declare three character variables to hold the three initials. In C, you could do that with the following statement:

```
1. main()
   {
   char first, middle, last;
    // Rest of program follows
   }
```

2. *main()*

   { char first;

   char middle;

   char last;

   // Rest of program follows

   }

- **Initialization of Variables**

  When a variable is declared, it contains undefined value commonly known as garbage value. If we want we can assign some initial value to the variables during the declaration itself. This is called *initialization of the variable.*

  Eg- int pageno=10;

  char grade='A';

  float salary= 20000.50;

Expressions

An expression consists of a combination of operators, operands, variables & function calls. An expression can be arithmetic, logical or relational. Here are some expressions:

a+b – arithmetic operation

a>b- relational operation a

== b - logical operation

func (a,b) – function call

4+21

a*(b + c/d)/20

q = 5*2 x =

++q % 3

q > 3

As you can see, the operands can be constants, variables, or combinations of the two. Some expressions are combinations of smaller expressions, called subexpressions. For example, c/d is a subexpression of the sixth example.

An important property of C is that every C expression has a value. To find the value, you perform the operations in the order dictated by operator precedence.

Statements

Statements are the primary building blocks of a program. A program is a series of statements with some necessary punctuation. A statement is a complete instruction to the computer. In C, statements are indicated by a semicolon at the end. Therefore

legs = 4

is just an expression (which could be part of a larger expression), but

legs = 4;

is a statement.

What makes a complete instruction? First, C considers any expression to be a statement if you append a semicolon. (These are called expression statements.) Therefore, C won't object to lines such as the following:

8;

3 + 4;

However, these statements do nothing for your program and can't really be considered sensible statements. More typically, statements change values and call functions:

x = 25;

++x;

y = sqrt(x);

Although a statement (or, at least, a sensible statement) is a complete instruction, not all complete instructions are statements. Consider the following statement:

x = 6 + (y = 5);

In it, the subexpression y = 5 is a complete instruction, but it is only part of the statement. Because a complete instruction is not necessarily a statement, a semicolon is needed to identify instructions that truly are statements.

Compound Statements (Blocks)

A compound statement is two or more statements grouped together by enclosing them in braces; it is also called a block. The following while statement contains an example:

```
while (years < 100)

{

   wisdom = wisdom * 1.05;

printf("%d %d\n", years, wisdom);

years = years + 1;


}
```

If any variable is declared inside the block then it can be declared only at the beginning of the block. The variables that are declared inside a block can be used only within the block.