Variables and Declaration

- *Variables* are identifiers whose value changes during the execution of the program. Variables specify the name and type information. The compiler allocates memory for a particular variable based on the type.
- Variables can be modified using the variable name or address of the variable. The variable name must be chosen in a meaningful way. The declaration of the variable must be done before it can be used in the program.
- The general syntax of the variable declaration is given below.

datatype : var1, var2, ….,varn;

where ***datatype : may be any data type***

var1, var2 : variable name separated by a comma

Examples

1. int sum, count;
2. int rollno;
3. float int_rate;
4. double avg, netsal;
5. char char;

Variable declaration with qualifiers

Examples

1. short int number;
2. unsigned int total;
3. long int ser_no;
4. long double volume;

Variable Initialization

- Assigning a relevant value to a variable for the first time in a program is known as *initialization*. Sometimes a variable may be initialized on its declaration itself. Variables can be initialized with a constant value or expression.

Syntax:

datatype variablename = expression;

(or)

datatype variablename = constant;

Example

1. int c = 10, d = c + 5;
2. float rate = 12.5;
3. char ch = 'Y';
4. int count = 0 , sum = 0;
5. float pi = 3.14;

CONSTANT AND VOLATILE VARIABLE

Constant variable

- If we want that the value of a certain variable remain the same or remain unchanged during the execution of a program, then it can be done only by declaring the variable as a constant.
- The keyword constant is then added before the declaration. It tells the compiler that the variable is a constant. Thus, constant declared variables are protected from modification.

  Example

  const int a = 20;

  where, **const** is a keyword, **a** is a variable name and **20** is a constant value. The compiler protects the value of 'a' from modification. The user cannot assign any value to a; by scanf ( ) statement the value can be replaced.

Volatile variable

- The *volatile variables* are those variables that are changed at any time by any other external program or the same program. The syntax is as follows.

  Example

  volatile int b;

  where **volatile** is a keyword and **b** is a variable If the value of a variable in the current program is to be maintained constant and desired not to be changed by any other external operation, then the declaration of the variable will be as follows;

  volatile const b = 20;

OPERATORS IN C

Operator: An operator is a symbol that specifies an operation to be performed on operands. Eg: x= a+b; where + is an operator.

Operands: An operand is an entity on which an operation is to be performed. An operand can be a variable name, a constant, a function call or a macro name.

Eg. x= a+b; where x, a, b are the operands.

Expression: An expression is a sequence of operands and operators that specifies the computations of a value. An expression is made up of one or more operands. Eg. x= a+b.

Types of Operators

1. Arithmetic Operators
2. Relational Operators
3. Logical Operators
4. Assignment Operators
5. Increment and Decrement Operators
6. Conditional Operators (Ternary Operators)
7. Bitwise Operators
8. Special Operators

Arithmetic Operators

- Addition, subtraction, multiplication, division and modulo are the arithmetic operations.
- The arithmetic operators are used for numerical calculations between two Constants

Operators	Explanations	Examples
+	Addition	2 + 2 = 4
-	Subtraction	3 - 2 = 1
*	Multiplication	5 * 4 =20
/	Division	10 / 2 = 5
%	Modular Division	11 % 2 = 1

Example:

```
void main()
{
int a=5, b=4, c;
```

```
c=a-b;
printf("%d", c);
}
```

- The following table show the division operator on various data types.

Operation	Result	Example
int/int	int	2/5=0
real/int	real	5.0/2=2.5
int/real	real	5/2.0=2.5
real/real	real	5.0/2.0=2.5

- Arithmetic operators can be classified as
  - **Unary arithmetic –** it requires only one operand.

    Example: +a, -b
  - **Binary arithmetic** – it requires two operands.

    Example: a+b, a-b, a/b, a%b
  - **Integer arithmetic** – it requires both operands to be integer type for arithmetic operation.

    Example:

    a=4, b=3

    a+b =4+3 =7

    a-b =4-3=1
  - **Floating Point arithmetic** – It requires both operands to be float type for arithmetic operation.

    Example:

    a=6.5, b=3.5

    a+b =6.5+3.5 =10.0

    a-b =6.5-3.5=3.0

Program 1.3

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

```
void main()
{
int b,c;
int sum, sub, mul;
float  div;
clrscr();
printf("enter the value of b,c:");
scanf("%d%d", &b, &c);
sum=b+c;
sub=b-c;
mul=b*c;
div=b/c;
printf("\n sum=%d,sub=%d,mul=%d,div=%f",sum,sub,mul,div);
getch();
}
```

Output:

Enter the value of b,c: 8 4

sum=12,sub=4,mul=32,div=2

Relational Operators

- Relational operators are used to compare two or more operands.
- Operands may be variable, constant or expression

Operators	Descriptions	Example	Return Value
>	Greater than	5>4	1
<	Less than	10<9	0
<=	Less than or equal to	10<=10	1
>=	Greater than or equal to	11>=5	1
==	Equal to	2==3	0
!=	Not equal to	3!=3	0

Syntax

AE1 ***operator*** AE2

where, AE- Arithmetic Expression or Variable or Value.

- These operators provide the relationship between two expressions.
- If the condition is true it returns a value 1, otherwise it returns 0.
- These operators are used in decision making process. They are generally used in conditional or control statement.

Logical Operators

- Logical Operators are used to combine the result of two or more conditions.
- The logical relationship between the two expressions is checked with logical operators.
- After checking the condition, it provides logical true (1) or false (0).

Operators	Descriptions	Example	Return Value
&&	Logical AND	5>3 && 5<10	1
||	Logical OR	8>5 || 8<2	1
!=	Logical NOT	8!=8	0

- **&& -** This operator is usually used in situation where two or more expressions must be true.

  Syntax:

  (exp1) && (exp2)

- **||** – This is used in situation, where at least one expression is true.

  Syntax:

  (exp1) || *(exp2)*

- **!** – This operator reverses the value of the expression it operates on. (i.e.,) it makes a true expression false and false expression true.

  Syntax:

  !(exp1)

Program 1.4

/* Program using Logical operators */

```
#include<stdio.h>
#include<conio.h>
void main( )
{
clrscr( );
printf("\n Condition : Return values ");
printf("\n 5<=8 && 4>2: %5d",5<=8 && 4>2);
printf("\n 5>=3 || 6<8: %5d",5>=3 || 6<8);
printf("\n !(7==7): %5d",!(7==7));
getch( );
}
```

Output

Condition : Return values

5<=8 && 4>2 : 1

5>=3 || 6<8 : 1

!(7==7) : 0

Assignment Operator

- Assignment Operator are used to assign constant or a value of a variable or an expression to another variable.

Syntax

variable =expression (or) value;

Example

```
x=10;
x=a+b;
x=y;
```

Program 1.5

/* Program using Assignment and Short-hand Assignment operators */

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

```
void main( )
{
int a=20,b=10,c=15,d=25,e=34,x=5;
clrscr( );
printf("\n Value of a=%d",a);
printf("\n Value of b=%d",b);
a+=x;
b-=x;
c*=x;
d/=x;
e%=x;
printf("\n Value of a=%d",a);
printf("\n Value of b=%d",b);
printf("\n Value of c=%d",c);
printf("\n Value of d=%d",d);
printf("\n Value of e=%d",e);
getch();
}
```

Output

Value of a = 20
Value of b = 10
Value of a = 25
Value of b = 5
Value of c = 75
Value of d = 5
Value of e = 4

Increment and Decrement Operators (Unary Operators)

- The '++' adds one to the variable and '--'subtracts one from the variable. These operators are called unary operators.

Operator	Meaning

++X	Pre increment
--X	Pre decrement
X++	Post increment
X--	Post decrement

Pre-increment operator

- This operator increment the value of a variable first and then perform other actions.

Program 1.6

```
#include <stdio.h>
void main()
{
int a,b;
a=10;
b=++a;
printf("a=%d",a);
printf("b=%d",b);
}
output: a=11 b=11
#include <stdio.h>
void main()
{
int a,b;
a=10;
b=—a;
printf("a=%d",a);
printf("b=%d",b);
}
```

Output:

a=9 b=9

Post-increment operator

- This operator perform other actions first and then increment the value of a variable.

Program 1.7

```
#include <stdio.h>
void main()
{
int a,b;
a=10;
b=a++;
printf("a=%d",a);
printf("b=%d",b);
}
```

Output:

a=11 b=10

Program 1.8

```
#include <stdio.h>
void main()
{
int a,b;
a=10;
b=a--;
printf("a=%d",a);
printf("b=%d",b);
}
```

Output:

a=9 b=10

Conditional Operator (or) Ternary Operator

- Conditional operator checks the condition itself and executes the statement depending on the condition.

Syntax

condition?exp1:exp2;

Example

```
void main()
{
int a=5,b=3,big;
big=a>b?a:b;
printf("big is…%d",big);
}
```

Output

big is…5

Bitwise Operators

- Bitwise operators are used to manipulate the data at bit level.
- It operates on integers only.
- It may not be applied to float or real.

Operator	Meaning
&	Bitwise AND
|	Bitwise OR
^	Bitwise XOR
<<	Shift left
>>	Shift right
~	One's complement

Bitwise AND (&):

- This operator is represented as '&' and operates on two operands of integer type. If both the operands bit is '1' then the result is '1'.

Bitwise OR (|):

- Bitwise OR (|) operator gives the value '1' if either of the operands bit is '1'

Bitwise Exclusive OR (^)

- Bitwise Exclusive OR(^) gives the value '1' if both operands bit are same.

a	b	a|b	a&b	a^b	~
0	0	0	0	0	1
0	1	1	0	1	1
1	0	1	0	1	0
1	1	1	1	0	0

Program 1.9

```
/* Program using One's complement operator */
#include<stdio.h>
#include<conio.h>
void main( )
{
int a;
clrscr( );
printf("\n Enter the value for a : ");
scanf("%d",&a);
printf("\n The One's complement value for a is : %d", ~a);
getch( );
}
```

Output

Enter the value for a: 3

The One's complement value for a is: -4

The Special Operator

- C language supports some of the special operators given below.

Operator	Meaning
,	Comma operators
sizeof	Size of operators
& and *	Pointer operators
. and — >	Member selection operators

a) Comma operator(,):

- The comma operator is used to separate the statement elements such as variables, constants or expression etc.,
- This operator is used to link the related expression together.
- Such expression can be evaluated from left to right and the value of right most expression is the value of combined expression.

Example:

val=(a=3,b=9,c=77,a+c);

Where,

First assigns the value 3 to a

Second assigns the value 9 to b

Third assigns the value 77 to c

Last assigns the value 80.

b) The sizeof() operator:

- The sizeof() is a unary operator that returns the length in bytes of the specified variable and it is very useful to find the bytes occupied by the specified variable in memory.

Syntax:

sizeof(var);

Example:

```
void main()
{
int a;
printf("size of variable a is…%d", sizeof(a));
}
```

Output:

size of variable a is…….2

c) Pointer operator:

- & : This symbol specifies the address of the variable.
- * : This symbol specifies the value of the variable.

d) Member selection operator:

- . and — >: These symbols are used to access the elements from a structure.