INSTRUCTION AND OPERATION IN

Topics covered:

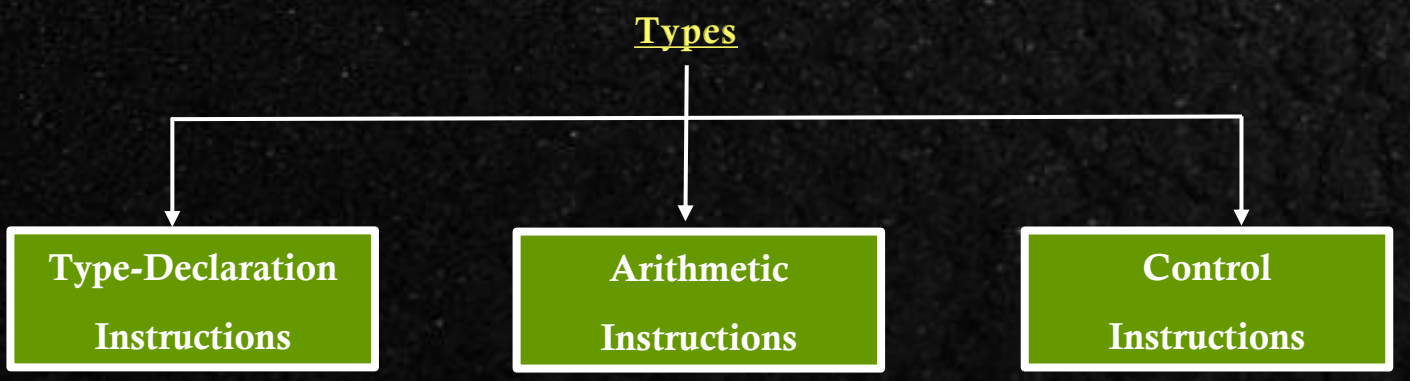
- Instructions
- Types of Instructions
    1) Type declaration Instructions
    2) Arithmetic instructions
    3) Control Instructions
- Operators
    1) Arithmetic Operators
    2) Relational Operators
    3) Logical Operators
    4) Bitwise Operators
    5) Assignment Operators
    6) Ternary Operators

CodeEncode

INSTRUCTION AND OPERATION IN C

Instruction

- ⇨ C instructions are the commands in the program that instructs the compiler to do certain action.
- ⇨ A statement written in C language is known as instruction.



Type Declaration Instructions :

- ⇨ Inform the compiler about the type of variable used.
- ⇨ Whenever a variable is used we need to specify the type of data it can hold.
- ⇨ (Declare var before using it).

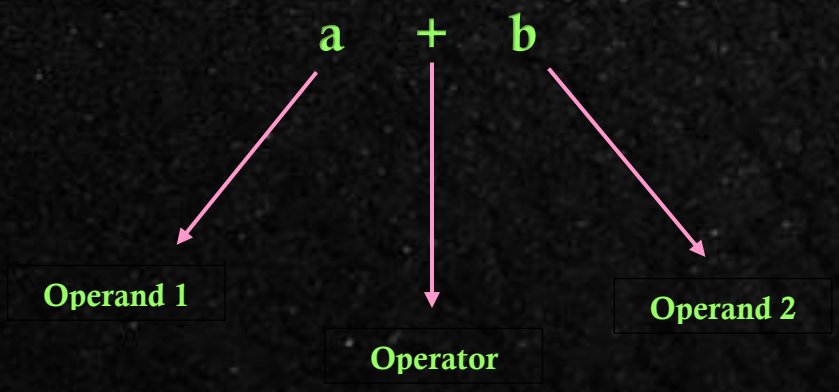
Valid Operations

```
int a = 48;
int b = a;
int c = b + 1;
int d = 1, e;
int a,b,c;
a = b = c = 1;
```

wrong way of initialization → `int a=b=c = 1`

Arithmetic Instructions :

- ⇨ Used to perform some arithmetic operations.
- ⇨ Uses arithmetic operators like +, - , * , / , %



NOTE : single variable on the LHS

Valid Operations

```
a = b + c
a = b * c
a = b / c
```

`int a,b = 2 * a`

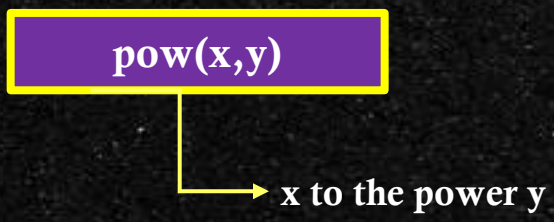
Invalid Operations

b + c = a

a = bc

a = b^c

power operation are performed using power function included in math library.



Modular Operator

⇨ Returns remainder for int

3 % 2 = 1

-3 % 2 = -1

NOTE : float data types are not allowed.

Type Conversions :

⇨ Conversion of one type of data to another.

int OP int → int
int OP float → float
float OP float → int

Types

Implicit Type Conversion

also known as →

Automatic Type Conversion

- ⇨ Done by compiler on its own
- ⇨ Data types are upgraded to the greater data type.

Ques. Write an example of Implicit type conversion.

Explicit Type Conversion

Higher Data type → Lower Data type

Syntax :

→ (type) Expression

Ques. Write an example of Explicit type conversion.

Operator Precedence

Operators are evaluated based on precedence.

* , / , %

+, -

=

x = 4 + 9 * 10

x = 4 * 3/ 6 * 2

Associativity :

(for same precedence)

⇨ Tells us the direction of execution of operators.

/ * → L to R

+ - → L to R

Control Instructions :

⇨ Used to determine the flow of control in the program

⇨ It enable us to specify the order within which the various instructions in each program are to be executed by the computer.

Types of Control Instruction :

a) Sequence Control Instruction
b) Decision Control Instruction
c) Case Control Instruction

Set of conditions to execute the instructions.

d) Loop Control Instruction → Executing a set of instructions for certain number of times.

Operators

Operator is a symbol given to an operation that operates on some value. It tells the computer to perform some mathematical or logical manipulations.

a) Arithmetic Operators
b) Relational Operators
c) Logical Operators
d) Bitwise Operators
e) Assignment Operators
f) Ternary Operators

Let us suppose **a = 10** **b = 5**

a) Arithmetic Operators

⇨ Arithmetic operator are used to perform basic arithmetic operations.

Operators	Description	Example
+	Add two integer or real type.	a + b gives 15
-	Subtracts two integer or real type.	a - b gives 5
*	Multiply two integer or real types.	a * b gives 50
/	Divide two integer or real types.	a / b gives 2
%	Modulus operator divide first operand from second and returns remainder.	a % b gives 0 (As 10/5 will have 0 remainder)

b) Relational Operators

➪ Relational operators are used to check relation between any two operands.

Operators	Description	Example
>	If value of left operand is greater than right, returns true else returns false	(a > b) returns true
<	If value of right operand is greater than left, returns true else returns false	(a < b) returns false
==	If both operands are equal returns true else false	(a == b) returns false
!=	If both operands are not equal returns true else false.	(a != b) returns true
>=	If value of left operand is greater or equal to right operand, returns true else false	(a >= b) returns true
<=	If value of right operand is greater or equal to left operand, returns true else false	(a <= b) will return false

c) Logical Operators

➪ Logical operators are used to combine two boolean expression together and results a single boolean value according to the operand and operator used.

Operators	Description	Example
&&	Used to combine two expressions. If both operands are true or Non-Zero, returns true else false	((a>=1) && (a<=10)) returns true since (a>=1) is true and also (a<=10) is true.
||	If any of the operand is true or Non-zero, returns true else false	((a>1) || (a<5)) will return true. As (a>1) is true. Since first operand is true hence there is no need to check for second operand.
!	Logical NOT operator is a unary operator. Returns the complement of the boolean value.	!(a>1) will return false. Since (a>1) is true hence its complement is false.

d) Bitwise Operators

➪ Bitwise operator performs operations on bit(Binary level).

Lets suppose a = 10, b = 5
a = 0000 1010 (8-bit binary representation of 10)
b = 0000 0101 (8-bit binary representation of 5)

Operators	Description	Example
&	*Bitwise AND* performs and operation on two binary bits value. If both are 1 then will result is 1 otherwise 0.	0000 1010 & 0000 0101 _____ 0000 0000
|	*Bitwise OR* returns 1 if any of the two binary bits are 1 otherwise 0.	0000 1010 | 0000 0101 _____ 0000 1111
^	*Bitwise XOR* returns 1 if both the binary bits are different else returns 0.	0000 1010 ^ 0000 0101 _____ 0000 1111
~	*Bitwise COMPLEMENT* is a unary operator. It returns the complement of the binary value i.e. if the binary bit is 0 returns 1 else returns 0.	~ 0000 1010 _____ 1111 0101
<<	*Bitwise LEFT SHIFT* operator is unary operator. It shift the binary bits to the left. It inserts a 0 bit value to the extreme right of the binary value.	0000 1010 << 2 = 0010 1000
>>	*Bitwise RIGHT SHIFT* operator is unary operator. It shifts the binary bits to the right. It inserts a 0 bit value to the extreme left of the binary value.	0000 1010 << 2 = 0000 0010

e) Assignment Operator

⇨ Assignment operator is used to assign value to a variable.

Operators	Description	Example
=	Assign value from right operand to left operand.	a = 10 will assign 10 in a
+=	Add AND assignment operator adds the current value of the variable on left to the value on the right and then assigns the result to the variable on the left.	(a += b) can be written as (a = a + b)
-=	Subtract AND assignment operator subtracts the current value of the variable on left from the value on the right and then assigns the result to the variable on the left.	(a -= b) can be written as (a = a - b)
*=	Multiply AND assignment operator multiplies the current value of the variable on left to the value on the right and then assigns the result to the variable on the left.	(a *= b) can be written as (a = a * b)
/=	Divide AND assignment operator divides the current value of the variable on left by the value on the right and then assigns the result to the variable on the left.	(a /= b) can be written as (a = a / b)
%=	Modulus AND assignment operator takes modulus using two operands and assigns the result to the left operand.	(a %= b) can be written as (a = a % b)

f) Ternary Operators

- Ternary operator as a conditional operator and is similar to simple if-else.
- It takes three operand.

Operators	Description	Example
?:	It is used as conditional operator. Syntax of using ternary operator: (condition) ? (true part) : (false part)	b = (a>1) ? a : b; will store the value 10 in b as (a>1) is true hence true part will execute, assigning the value of a in b.

Credits :

Team CodeEncode