

Arithmetic Operators and Flowcharts

PRESENTED BY: SATYA PRAKASH PATEL
EMAIL: SATYAPATEL.IND@GMAIL.COM

Operators in C

	Operator	Type
Unary operator →	+ +, - -	Unary operator
Binary operator {	+, -, *, /, %	Arithmetic operator
	<, <=, >, >=, ==, !=	Relational operator
	&&, , !	Logical operator
	&, , <<, >>, ~, ^	Bitwise operator
	=, +=, -=, *=, /=, %=	Assignment operator
Ternary operator →	?:	Ternary or conditional operator



C Programming Operators

C Arithmetic Operators

Operator	Meaning of Operator
+	addition or unary plus
-	subtraction or unary minus
*	multiplication
/	division
%	remainder after division(modulo division)

Increment and decrement operators

Operator	Meaning of Operator
++	Increment
--	Decrement

Pre increment operator is used to increment variable value by 1 before assigning the value to the variable.

Post increment operator is used to increment variable value by 1 after assigning the value to the variable.

Example on pre increment and post increment

```
#include<stdlib.h>
```

```
main()
```

```
{
```

- int a= 4;
- int b= 5;
- a=b++;
- **a=b++;**
- b=a++;
- printf("hello world a= %d, b=%d",a,b);

```
}
```

```
#include<stdlib.h>
```

```
main()
```

```
{
```

```
    int a= 4;  
    int b= 5;  
    a=b++;  
    a=++b;  
    b=a++;  
    printf("hello world a= %d, b=%d",a,b);
```

```
}
```

C Assignment Operators

Operator	Example	Same as
=	a = b	a = b
+=	a += b	a = a+b
-=	a -= b	a = a-b
*=	a *= b	a = a*b
/=	a /= b	a = a/b
%=	a %= b	a = a%b

Relational Operators

Operator	Example
==	(A == B)
!=	(A != B)
>	(A > B)
<	(A < B)
>=	(A >= B)
<=	(A <= B)

Logical Operators

Operator	Description	Example
&&	Called Logical AND operator. If both the operands are non-zero, then the condition becomes true.	(A && B)
	Called Logical OR Operator. If any of the two operands is non-zero, then the condition becomes true.	(A B)
!	Called Logical NOT Operator. It is used to reverse the logical state of its operand. If a condition is true, then Logical NOT operator will make it false.	!(A && B)

Bitwise Operators

Operator	Description	Example
&	Binary AND Operator copies a bit to the result if it exists in both operands.	(A & B) = 12, i.e., 0000 1100
	Binary OR Operator copies a bit if it exists in either operand.	(A B) = 61, i.e., 0011 1101
^	Binary XOR Operator copies the bit if it is set in one operand but not both.	(A ^ B) = 49, i.e., 0011 0001
~	Binary One's Complement Operator is unary and has the effect of 'flipping' bits.	(~A) = ~(60), i.e., -0111101
<<	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.	A << 2 = 240 i.e., 1111 0000
>>	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.	A >> 2 = 15 i.e., 0000 1111

'A' holds 60 and variable 'B' holds 13

Other Operators

Comma Operator : Comma operators are used to link related expressions together. For example:

```
int a, b, c=5;
```

sizeof Operator : The sizeof is an unary operator which returns the size of data (constant, variables, array, structure etc)

Example : sizeof(a), sizeof(int)

Other Operators

Operator	Description	Example
&	Returns the address of a variable.	&a; returns the actual address of the variable.
*	Pointer to a variable.	*a;
? :	Conditional Expression.	If Condition is true ? then value X : otherwise value Y

Category	Operator	Associativity
Postfix	() [] -> . ++ --	Left to right
Unary or prefix	+ - ! ~ ++ -- (type)* & sizeof	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Shift	<< >>	Left to right
Relational	< <= > >=	Left to right
Equality	== !=	Left to right
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right
Bitwise OR		Left to right
Logical AND	&&	Left to right
Logical OR		Left to right
Conditional	?:	Right to left
Assignment	= += -= *= /= %>>= <<= &= ^= =	Right to left
Comma	,	Left to right

Format specifier	Description	Supported data types
%c	Character	char /unsigned char
%d	Signed Integer	short /unsigned short/ int/ long
%e or %E	Scientific notation of float values	float/ double
%f	Floating point	float
%l or %ld or %li	Signed Integer	long
%lf	Floating point	double
%Lf	Floating point	long double
%lu	Unsigned integer	unsigned int / unsigned long
%lli, %lld	Signed Integer	long long
%llu	Unsigned Integer	unsigned long long
%o	Octal representation of Integer.	Short/unsigned short int/unsigned int/long
%p	Address of pointer to void void *	void *
%s	String	char *
%u	Unsigned Integer	unsigned int / unsigned long

Type Conversion, Precedence and Associativity of Operators in C

Type Conversion in C

1. The process of converting one data type into another data type is known as type conversion.
2. implicit type conversion. It is done by the compiler.
3. explicit type conversion.

Arithmetic operation between integer and integer will always result in an integer.

Example

```
int a = 5, b=6;
```

```
int sum=0;
```

```
Sum=a+b
```

Output **Sum= 9**

Arithmetic operation between float and float will always give float number.

Example

Float a = 5, b=6;

Float sum=0;

Sum=a+b;

Output Sum= 9.000000

Arithmetic operation between float and integer will always give float number.

In this case, first integer will be promoted to float after that the arithmetic operation will take place.

Operation	Result	Operation	Result
5 / 2	2	2 / 5	0
5.0 / 2	2.5	2.0 / 5	0.4
5 / 2.0	2.5	2 / 5.0	0.4
5.0 / 2.0	2.5	2.0 / 5.0	0.4

Type Conversion in Assignments

Sometimes variable on the left hand side of assignment operator (=) does not match the type of variable on its right hand side.

Example:

```
float a;
```

```
int b;
```

```
b=4.2;
```

```
a=3;
```

Arithmetic Instruction	Result	Arithmetic Instruction	Result
$k = 2 / 9$	0	$a = 2 / 9$	0.0
$k = 2.0 / 9$	0	$a = 2.0 / 9$	0.2222
$k = 2 / 9.0$	0	$a = 2 / 9.0$	0.2222
$k = 2.0 / 9.0$	0	$a = 2.0 / 9.0$	0.2222
$k = 9 / 2$	4	$a = 9 / 2$	4.0
$k = 9.0 / 2$	4	$a = 9.0 / 2$	4.5
$k = 9 / 2.0$	4	$a = 9 / 2.0$	4.5
$k = 9.0 / 2.0$	4	$a = 9.0 / 2.0$	4.5

K is of `int` type and a is of `float` type

Output and flowchart of the code

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

```
int main()
```

```
{
```

```
    int x, y = 5, z = 5;
```

```
    x = y == z;
```

```
    printf("%d", x);
```

```
    return 0;
```

```
}
```

Output and flowchart of the code

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

```
int main()
```

```
{
```

```
    int y = 0;
```

```
    int x = (~y == 1);
```

```
    printf("%d", x);
```

```
    return 0;
```

```
}
```

Output and flowchart of the code

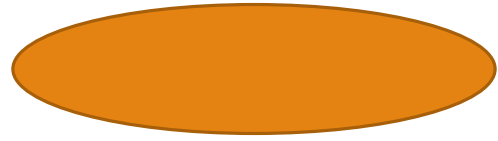
Flowcharts

What is a Flowchart?

1. Flowchart is a graphical representation of an algorithm.
2. it is use as a program-planning tool to solve a problem.
3. It makes use of symbols which are connected among them to indicate the flow of information and processing.

Basic Symbols used in Flowchart Designs

Terminal:



1. The oval symbol indicates Start, Stop and Halt in a program's logic flow.
2. A pause/halt is generally used in a program logic under some error conditions.
3. Terminal is the first and last symbols in the flowchart.

Input/Output:



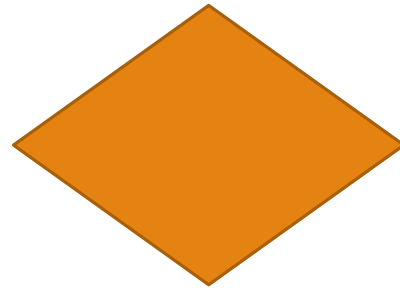
1. A parallelogram denotes any function of input/output type.
2. Program instructions that take input from input devices and display output on output devices are indicated with parallelogram in a flowchart.

Processing



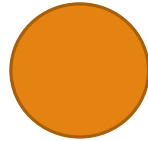
1. A box represents arithmetic instructions.
2. All arithmetic processes such as adding, subtracting, multiplication and division are indicated by action or process symbol.

Decision



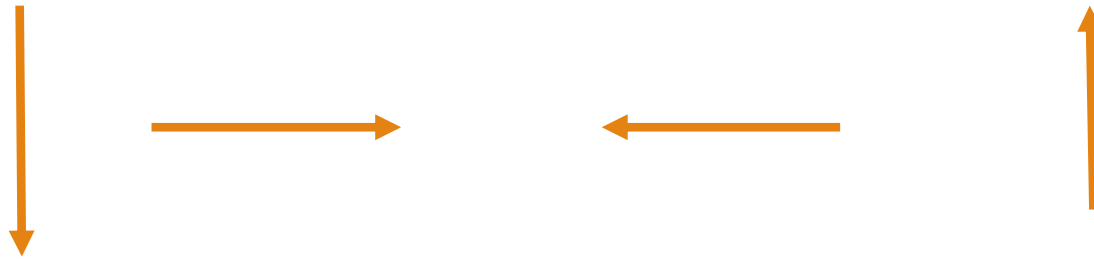
1. Diamond symbol represents a decision point.
2. Decision based operations such as yes/no question or true/false are indicated by diamond in flowchart.

Connectors



1. Whenever flowchart becomes complex or it spreads over more than one page, it is useful to use connectors to avoid any confusions.
2. It is represented by a circle.

Flow lines



1. Flow lines indicate the exact sequence in which instructions are executed.
2. Arrows represent the direction of flow of control and relationship among different symbols of flowchart.

Question 1

Who developed the C programming language?

- A. Bjarne Stroustrup
- B. James Gosling
- C. Dennis Ritchie
- D. Ray Boyce

Question 2

A name having a few letters, numbers and special character _(underscore) is called

- A. keywords
- B. reserved keywords
- C. tokens
- D. identifiers

Question 3

The size of a character variable in C is

- A. 8 bytes
- B. 4 bytes
- C. 2 bytes
- D. 1 byte

Question 4

By default a real number is treated as a

- A. float
- B. double
- C. long double
- D. integer

Question 5 : What is the output of the code

```
#include <stdio.h>
int main()
{
    int i = -3;
    int k = i % 2;
    printf("%d\n", k);
}
```

- A. Compile time error
- B. -1
- C. 1
- D. Implementation defined

Question 1 (11th jan)

Find the value of Z?

```
int Z =(1 > 2 + 3 && 4);
```

Answer

- A. 0
- B. 1
- C. Non zero garbage value
- D. error

Question 2 (11th jan)

Find the value of Z?

```
int A= (1 == 2 != 3);
```

Answer

- A. 0
- B. 1
- C. Non zero garbage value
- D. error

Question 3 (11th jan)

```
Int main(){  
    a=b=0;  
    a= b+++ ++b + ++b;  
    printf("fourth a= %d, b=%d\n\n",a,b);  
    return 0;  
}
```

Answer

A. a=2,b=2

B. A=3,b=3

C. A=5,b=3

D. A=6,b=3

Question 4 (11th jan)

What will be the output of following program ?

```
#include <stdio.h>

void main()

{
    printf("value is = %d", (10++));
}
```

Answer

A. 10

B. 11

C. Error

D. 0

Question 5 (11th jan)

What will be the output of following program ?

```
#include <stdio.h>

void main()
{
    int a=10,b=2,x=0;

    x=a+b*a+10/2*a;

    printf("value is =%d",x);
}
```

Answer

- A. valie is =1250**
- B. value is =80**
- C. value is =125**
- D. ERROR**

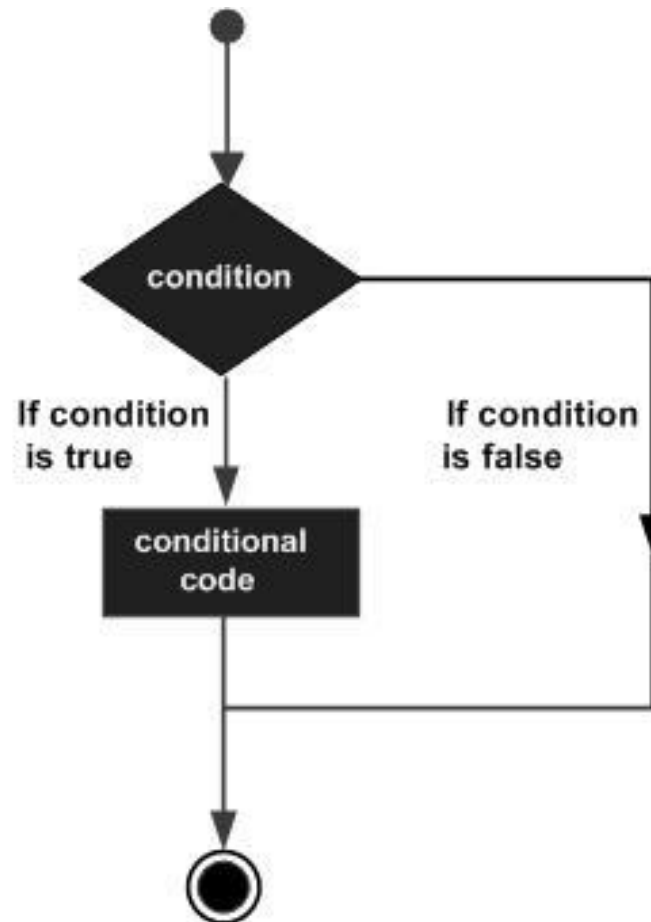
Assignment

Initial value	operation	Result of a and b
a=0,b=0	a= b+++ b+++ b++;	
a=0,b=0	a= b+++ b+++ ++b;	
a=0,b=0	a= b+++ ++b + b++;	
a=0,b=0	a= b+++ ++b + ++b;	
a=0,b=0	a= ++b+ b++ + b++;	
a=0,b=0	a= ++b+ b++ + ++b;	
a=0,b=0	a= ++b+ ++b + b++;	
a=0,b=0	a= ++b+ ++b + ++b;	

Queries and Feedback



```
if(boolean_expression) {  
  /* statement(s) will execute if the boolean expression is true */  
}
```



```
if(boolean_expression) {  
    /* statement(s) will execute if the boolean expression is true */  
} else {  
    /* statement(s) will execute if the boolean expression is false */  
}
```

