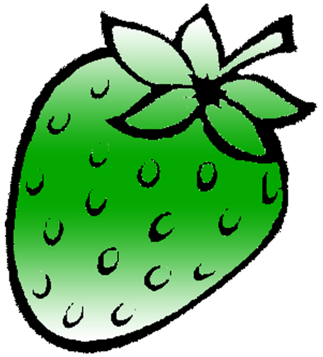
STRAWBERRY



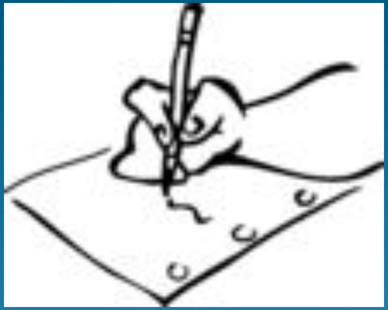
/strawberrydevelopers

/strawberry_app

For more visit:

Strawberrydevelopers.weebly.com

Operators & Expressions



Department of Computer Science

Developed by Strawberry

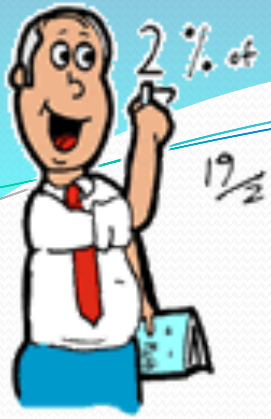
Definition

“An operator is a symbol (+,-,*,/) that directs the computer to perform certain mathematical or logical manipulations and is usually used to manipulate data and variables”

Ex: a+b

Operators in C

1. Arithmetic operators
2. Relational operators
3. Logical operators
4. Assignment operators
5. Increment and decrement operators
6. Conditional operators
7. Bitwise operators
8. Special operators



Arithmetic operators

Operator	example	Meaning
+	a + b	Addition –unary
-	a – b	Subtraction- unary
*	a * b	Multiplication
/	a / b	Division
%	a % b	Modulo division- remainder

Arithmetic operators

```
//Program to convert a given number of days into months and days
#include <stdio.h>
#include<conioh>
void main()
{
        int months, days;
        printf(“\n Enter days:”);
        scanf(“%d”,&days);
        months=days/30;
        days=days%30;
        printf(“\n Months = %d  Days = %d”,months,days);
        getch();
}
```

Relational Operators

Operator	Meaning
<	Is less than
<=	Is less than or equal to
>	Is greater than
>=	Is greater than or equal to
==	Equal to
!=	Not equal to

Relational Operators

- Used to compare two quantities.

- For example: to compare the age of two persons, or the price of two items so on..

- The value of relational expression is either true (1) or false (0).

- 10 < 20 is true

- 20<10 is false

Logical Operators

Operator	Meaning
&&	Logical AND
||	Logical OR
!	Logical NOT

Logical expression or a compound relational expression-

An expression that combines two or more relational expressions

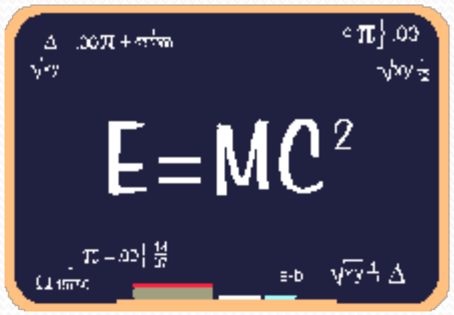
Ex: if (a==b && b==c)

Truth Table

a	b	Value of the expression	
		a && b	a || b
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

```
#include<stdio.h>
void main()
{
int num1 = 30;
int num2 = 40;

if(num1>=40 || num2>=40)

printf("Or If Block Gets Executed");

if(num1>=20 && num2>=20)

printf("And If Block Gets Executed");

if( !(num1>=40))

printf("Not If Block Gets Executed"); }
```

Assignment operators



Syntax:

v op = exp;

Where v = variable,

op = shorthand assignment operator

exp = expression

Ex: x=x+3

x+=3

Shorthand Assignment operators

Simple assignment operator	Shorthand operator
a = a+1	a + =1
a = a-1	a - =1
a = a* (m+n)	a * = m+n
a = a / (m+n)	a / = m+n
a = a %b	a %=b

Program to print a sequence of squar o numbers

```
#include<stdio.h>
#define N 100
#define A 2
void main()
{
        int a;
        a=A;
        while(a<N)
        {
        a*=a;
        printf("%d\n",a);
                }
}
```

Increment & Decrement Operators

C supports 2 useful operators namely

1. Increment ++
2. Decrement – operators

The ++ operator adds a value 1 to the operand

The – operator subtracts 1 from the operand

++a or a++

--a or a--

Rules for ++ & -- operators

1. These require variables as their operands
2. When postfix either ++ or – is used with the variable in a given expression, the expression is evaluated first and then it is incremented or decremented by one
3. When prefix either ++ or – is used with the variable in a given expression, it is incremented or decremented by one first and then the expression is evaluated with the new value

Examples for ++ & -- operators

Let the value of a =5 and b=++a then

a = b =6

Let the value of a = 5 and b=a++ then

b=5 but a=6

i.e.:

1. a prefix operator first adds 1 to the operand and then the result is assigned to the variable on the left
2. a postfix operator first assigns the value to the variable on left and then increments the operand.

Conditional operators

Syntax:

exp1 ? exp2 : exp3

Where exp1,exp2 and exp3 are expressions

Working of the ? Operator:

Exp1 is evaluated first, if it is nonzero(1/true) then the expression2 is evaluated and this becomes the value of the expression,

If exp1 is false(0/zero) exp3 is evaluated and its value becomes the value of the expression

Ex: m=2; if(m>n) then r=m else r=n

n=3

r=(m>n) ? m : n;

Bitwise operators

These operators allow manipulation of data at the bit level

Operator	Meaning
&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR
<<	Shift left
>>	Shift right

Special operators

1. Comma operator (,)
2. sizeof operator – sizeof( )
3. Pointer operators – (& and *)
4. Member selection operators – (. and ->)

Sizeof Operator

- The sizeof operator is a compile time operator and when used with an operand, it returns the number of bytes the operand occupies. The operand may be a variable, a constant or a data type qualifier.

- M=sizeof(num);
- N=sizeof(sum);

Sample Program

```
main(){
   int a,b,c,d;
   a=15; b=10;
   c=++a-b;
   printf("a=%d b=%d c=%d\n",a,b,c);
   d=b++ +a;
   printf("a=%d b=%d  d=%d\n",a,b,d);
   printf("a/b = %d", a/b);
   printf("a%b = %d",a%b);
   printf("%d\n", (c>d) ? 1:0);
Printf("%d\n",(c<d)?1:0);
}
```

Arithmetic Expressions

Algebraic expression	C expression
axb-c	a*b-c
(m+n)(x+y)	(m+n)*(x+y)
$\left[\frac{ab}{c}\right]$	a*b/c
$3x^2+2x+1$	3*x*x+2*x+1
$\frac{a}{b}$	a/b
S= $\frac{a+b+c}{2}$	S=(a+b+c)/2

Arithmetic Expressions

Algebraic expression	C expression
area= $\sqrt{s(s-a)(s-b)(s-c)}$	area=sqrt(s*(s-a)*(s-b)*(s-c))
Sin $\left(\frac{b}{\sqrt{a^2+b^2}}\right)$	sin(b/sqrt(a*a+b*b))
$\tau_1 = \sqrt{\left\{\frac{\sigma_x - \sigma_y}{2}\right\} + \tau xy^2}$	tow1=sqrt((rowx-rowy)/2+tow*x*y*y)
$\tau_1 = \sqrt{\left\{\frac{\sigma_x - \sigma_y}{2}\right\}^2 + \tau xy^2}$	**tow1=sqrt(pow((rowx-rowy)/2,2)+tow*x*y*y)**
$y = \frac{\alpha + \beta}{\sin \theta} + \|x\|$	**y=(alpha+beta)/sin(theta*3.1416/180)+abs(x)**

Precedence of operators

BODMAS RULE-

- Brackets of Division Multiplication Addition Subtraction

- Brackets will have the highest precedence and have to be evaluated first, then comes division, multiplication, addition and finally subtraction.

- C language uses some rules in evaluating the expressions and they r called as precedence rules, with some operators with highest precedence and some with least.

- The 2 distinct priority levels of arithmetic operators in c are-

- Highest priority : * / % Lowest priority : + -

Rules for evaluation of expression

1. First parenthesized sub expression from left to right are evaluated.
2. If parentheses are nested, the evaluation begins with the innermost sub expression
3. The precedence rule is applied in determining the order of application of operators in evaluating sub expressions
4. The associatively rule is applied when 2 or more operators of the same precedence level appear in a sub expression.
5. Arithmetic expressions are evaluated from left to right using the rules of precedence
6. When parentheses are used, the expressions within parentheses assume highest priority

Precedence of Arithmetic Operators

High Priority * / %

Low priority + -

X= a-b/3+c*2-1   where a=9,b=12 and c=3

X=9-12/3+3*2-1

Step 1: 9-4+3*2-1

Step 2: 9-4+6-1

Step 3: 5+6-1

Step 4: 11-1

Step 5: 10

X= 9-12/(3+3)*(2-1)

X=9-12/6*(2-1)

X=9-12/6*1

X=9-2*1

X=9-2

X=7

Precedence of Arithmetic Operators

X= 9-(12/(3+3)*2)-1

X=9-(12/6*2)-1

X=9-(2*2)-1

X=9-4-1

X=5-1

X=4

Example 1

Evaluate x1=(-b+ sqrt (b*b-4*a*c))/(2*a) @ a=1, b=-5, c=6

=(-(-5)+sqrt((-5)(-5)-4*1*6))/(2*1)

=(5 + sqrt((-5)(-5)-4*1*6))/(2*1)

=(5 + sqrt(25 -4*1*6))/(2*1)

=(5 + sqrt(25 -4*6))/(2*1)

=(5 + sqrt(25 -24))/(2*1)

=(5 + sqrt(1))/(2*1)

=(5 + 1.0)/(2*1)

=(6.0)/(2*1)

=6.0/2 = 3.0

Example 2

Evaluate the expression when a=4

b=a- ++a

=a – 5

=5-5

=0

Type Conversion in Expressions

- Implicit Type Conversion
- Explicit Type Conversion

1. Implicit type conversion:

   C permits mixing of constants and variables of different types in an expression. C automatically converts any intermediate values to the proper type so that the expression can be evaluated without loosing any significance. This automatic type conversion is known as implicit type conversion.

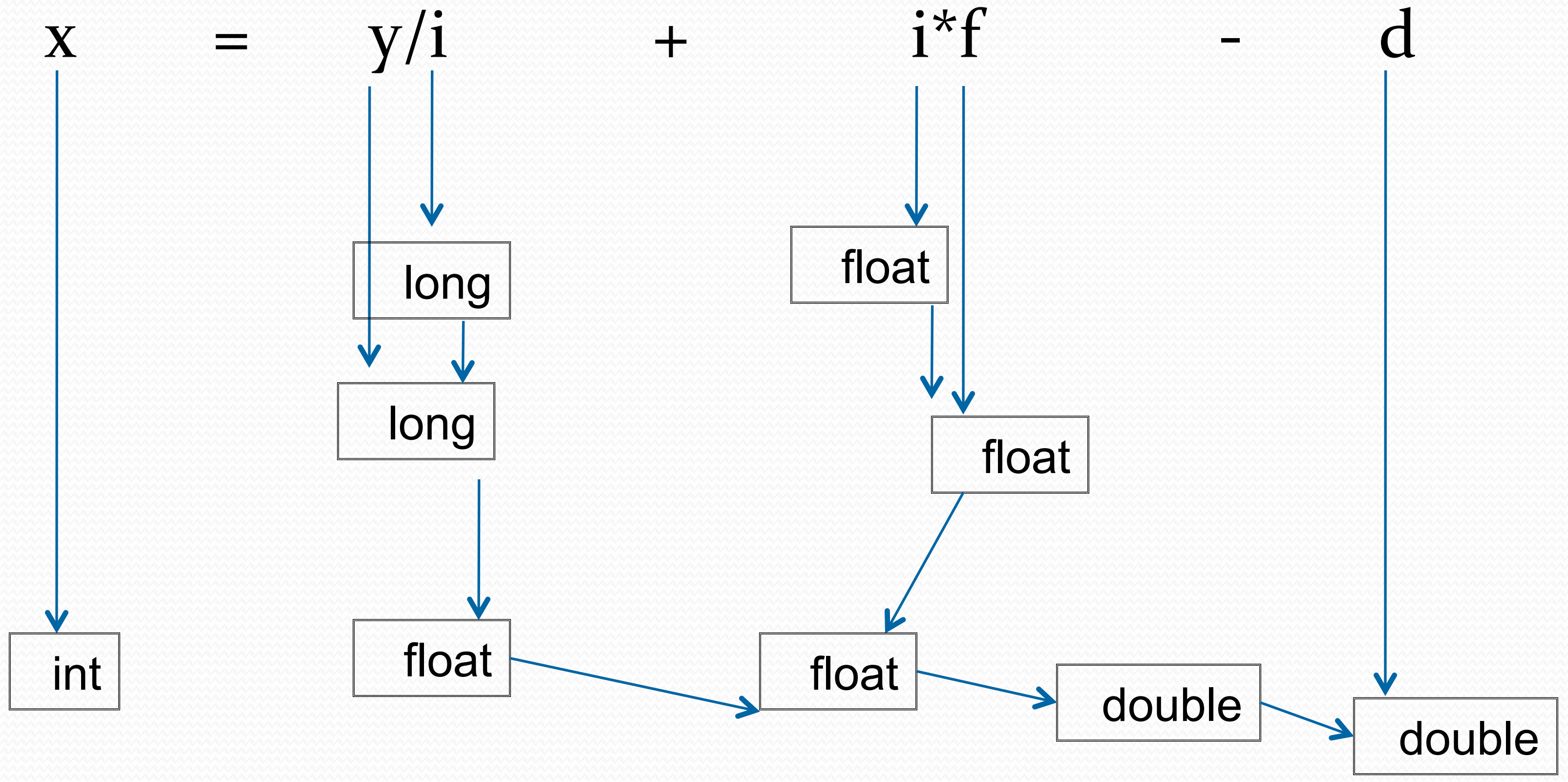
Implicit Type Conversion

If the operands are of different types, the ‘lower’ type is automatically converted to higher type before the operation precedes.

1. float to int causes truncation of the fractional part.
2. double to float causes rounding of digits.
3. long int to int causes dropping of the excess higher order bits.

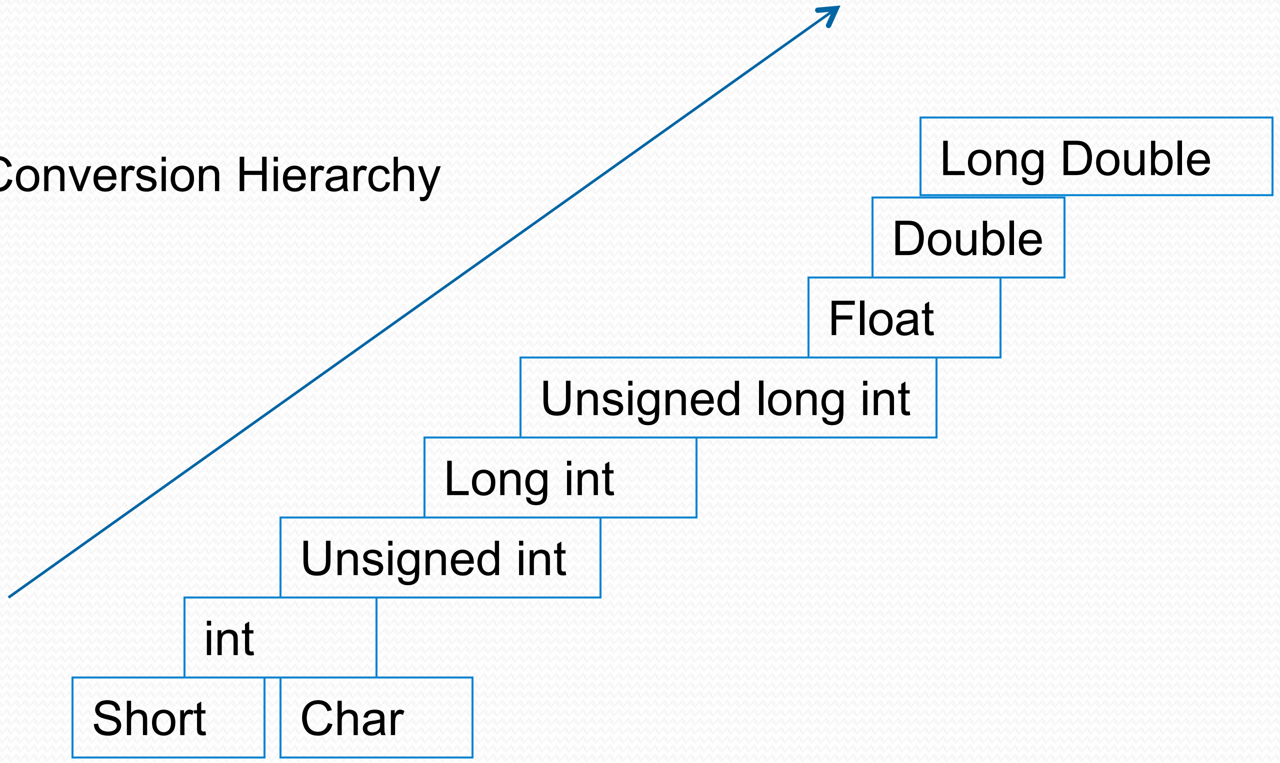
Implicit Type Conversion

int i, x; float f; double d;   long int y;



Implicit Type Conversion

Conversion Hierarchy



Explicit Type Conversion

We have just discussed how c performs type conversion automatically. However, there are instances when we want to force a type conversion in a way that is different from automatic type conversion.

Ratio= female_number/male_number;

Ratio=(float) female_number/male_number;

X=(int)7.5;   // 7.5 is converted to integer by truncation

A=(int)21.3/(int)4.5;   // Evaluated as 21/4 and the result=5

Y = (int)(a+b); //The result of a+b is converted to integer

Z = (int)a+b; // a is converted to integer and then added to b.

Operator	Description	Associativity
( )	Parentheses (function call)	left-to-right
[ ]	Brackets (array subscript)	
.	Member selection via object name	
->	Member selection via pointer	
++ --	Postfix increment/decrement	
++ --	Prefix increment/decrement	right-to-left
+ -	Unary plus/minus	
! ~	Logical negation/bitwise complement	
(type)	Cast (convert value to temporary value of type)	
*	Dereference	
&	Address (of operand)	
sizeof	Determine size in bytes on this implementation	
* / %	Multiplication/division/modulus	left-to-right
+ -	Addition/subtraction	left-to-right
<< >>	Bitwise shift left, Bitwise shift right	left-to-right
< <=	Relational less than/less than or equal to	left-to-right
> >=	Relational greater than/greater than or equal to	
== !=	Relational is equal to/is not equal to	left-to-right
&	Bitwise AND	left-to-right
^	Bitwise exclusive OR	left-to-right
|	Bitwise inclusive OR	left-to-right
&&	Logical AND	left-to-right
| |	Logical OR	left-to-right
? :	Ternary conditional	right-to-left
=	Assignment	right-to-left
+= -=	Addition/subtraction assignment	
*= /=	Multiplication/division assignment	

Operator Precedence and Associativity

- If(x==10+15 && y<10)

  if(x=25 && y<10)

  x==25 is false (0)

  y<10 is true(1)

  if(false && True) - false

Programs

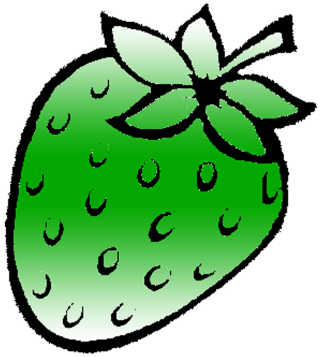
- Write a program to read three values using scanf statement and print the following results:

  1. Sum of values
  2. Average of three values
  3. Largest of three values
  4. Smallest of three values

- Write a program to compute and display sum of all integers that are divisible by 6 and not by 4 and lie between 0 and 100. The program should also count and display the number of such values.
- Write a program using do while loop to calculate and print the fibbonanci series.

1 1 2 3 5 8 13…

STRAWBERRY



/strawberrydevelopers

/strawberry_app

For more visit:

Strawberrydevelopers.weebly.com