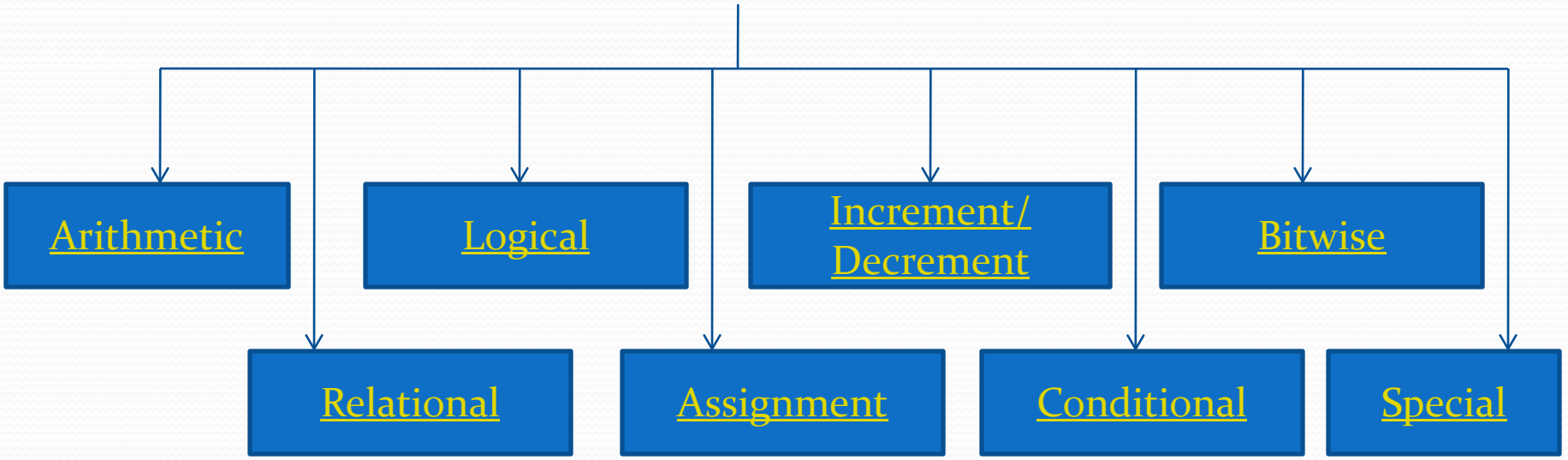
Operators and Hierarchy

Types of Operators



Precedence of Operators

- There are 2 different priorities of arithmetic operators
  - High Priority: * / %
  - Low Priority: + -

- The equation is evaluated in two passes
  - First pass: High priority operators
  - Second pass: Low priority operators

Expression: x=9-12/3+3*2-1

- 1st Pass

  x=9-4+3*2-1

  x=9-4+6-1

- 2nd Pass

  x=5+6-1

  x=11-1

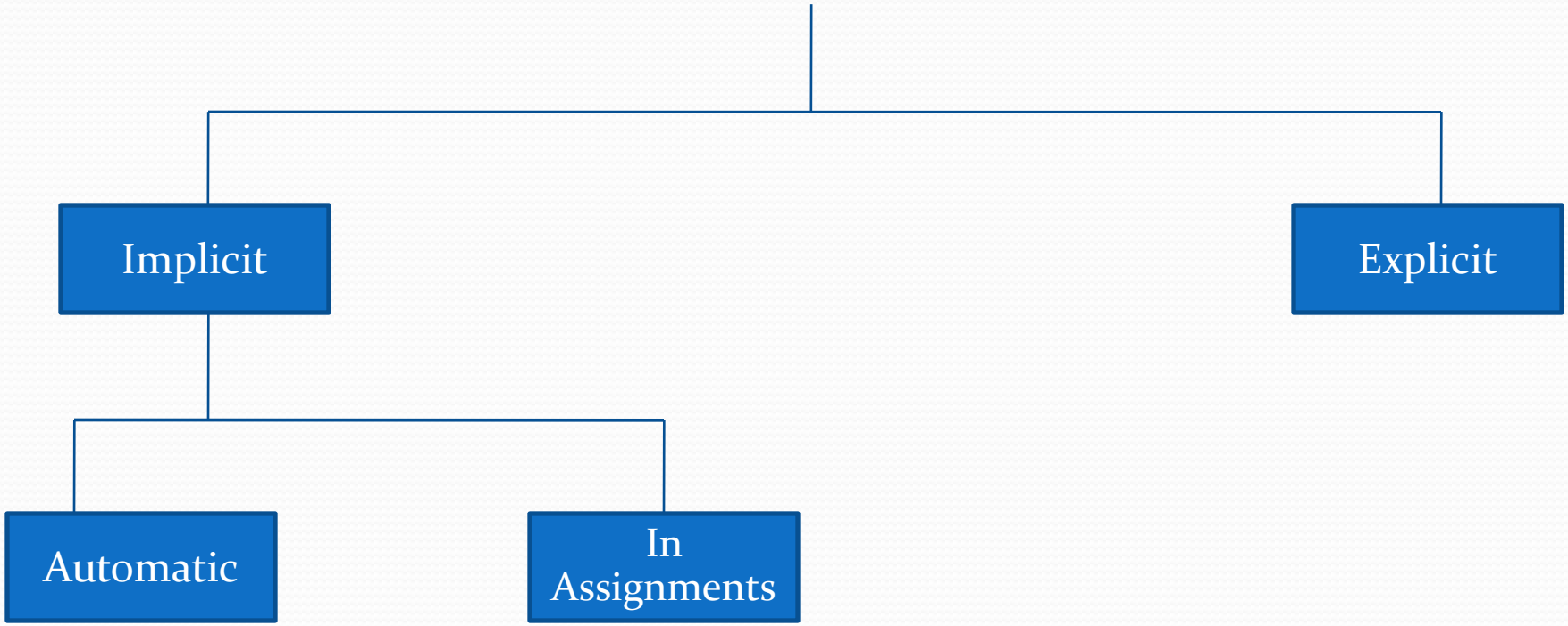
  x=10

Rules for Evaluation of Expression

- Parenthesized sub expression from left to right are evaluated
- If parenthesis are nested evaluation begins with innermost braces
- If operators of same precedence are encounter then associativity is used
- Arithmetic expression are evaluated from left to right

Operator	Description	Precedence	Associativity
() [] . -> ++ --	Parentheses (function call) (see Note 1) Brackets (array subscript) Member selection via object name Member selection via pointer Postfix increment/decrement (see Note 2)	1	left-to-right
++ -- + - ! ~ (type) * & sizeof	Prefix increment/decrement Unary plus/minus Logical negation/bitwise complement Cast (change type) Dereference Address Determine size in bytes	2	right-to-left
* / %	Multiplication/division/modulus	3	left-to-right
+ -	Addition/subtraction	4	left-to-right
<< >>	Bitwise shift left, Bitwise shift right	5	left-to-right
< <= > >=	Relational less than/less than or equal to Relational greater than/greater than or equal to	6	left-to-right
== !=	Relational is equal to/is not equal to	7	left-to-right
&	Bitwise AND	8	left-to-right
^	Bitwise exclusive OR	9	left-to-right
|	Bitwise inclusive OR	10	left-to-right
&&	Logical AND	11	left-to-right
||	Logical OR	12	left-to-right
?:	Ternary conditional	13	right-to-left
= += -= *= /= %= &= ^= |= <<= >>=	Assignment Addition/subtraction assignment Multiplication/division assignment Modulus/bitwise AND assignment Bitwise exclusive/inclusive OR assignment Bitwise shift left/right assignment	14	right-to-left
,	Comma (separate expressions)	15	left-to-right

Type Conversions

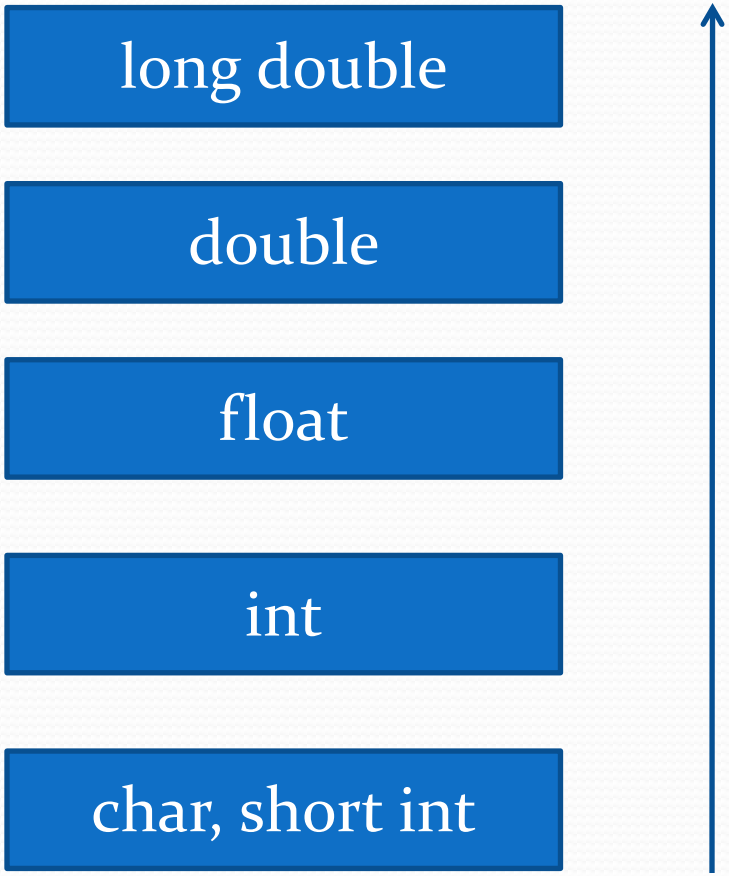


The data type of one operand is converted into data type of another operand

Implicit Type Conversion

- Implicit type conversion, also known as coercion
- An automatic type conversion by the compiler
- If operands are of different types then lower type is automatically converted to higher type

Automatic



In Assignment

- Type of right hand side is converted to type of left hand side
- If right hand operand is lower rank then it will be promoted
  - float = int
  - int = char
- If right hand operand is higher rank than it will be demoted
  - char=int
  - int=float

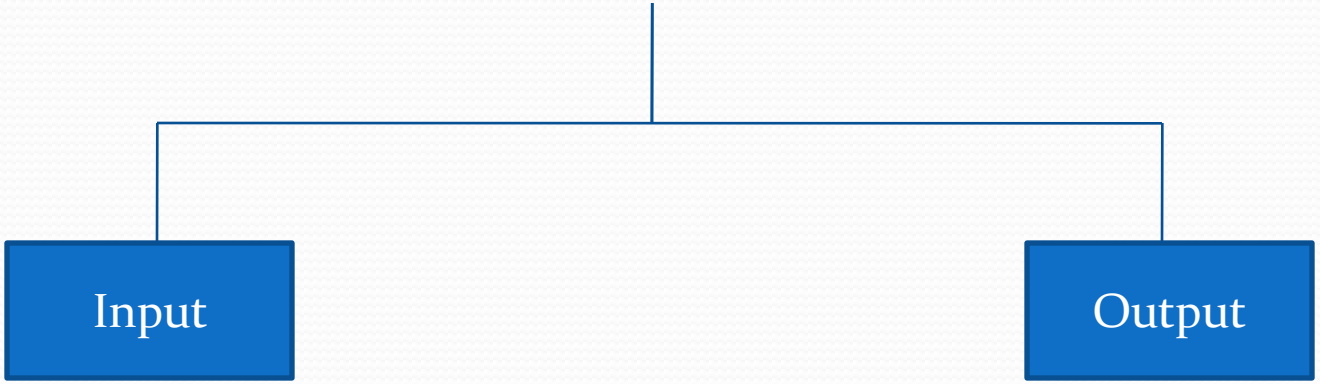
Explicit/Type Casting

- Is done with the help of cast operator
- Cast operator is a unary operator that is used for converting an expression to a particular data type
- Syntax:
  - (datatype) expression
- Ex:

```
int x,y;
float x=(float)x/y;
```

Input/Output

Types of Operations



The set of library functions that perform input-output operation is known as standard input/output library (stdio.h)

Reading a Character

- getchar();

- Accepts any character keyed in including
  - return (enter)
  - tab space

- Ex:

```
char  variable_name;
variable_name=getchar();
```

Writing a Character

- putchar(variable_name);
- Displays char represented by var_name on the terminal
- Ex:

```
char c=getchar();
putchar(c);
```

Conversion Specifications

Specifier	meaning
%c	a single character
%d or %i	decimal integer
%f	floating point number
%lf	long range floating point (double)
%Lf	long double
%h	short int
%s	string
%u	unsigned decimal integer
%o	octal integer
%x	hexadecimal

Formatted Input

- C provides scanf() function for entering input data
- Syntax
  - scanf(“control string”, address1, address2….);
  - Control string specifies the format in which data has to be entered
  - address1, address2 specifies the address of locations where data is to be stored

Examples Integer Numbers

- Format: %wd
  - w is the field width

- Ex 1

```
int marks;
scanf("%d",&marks);
```

- Ex 2

```
char str[30];
scanf("%s",str);
```

- Ex 3

```
int basic,da;
scanf("%d%d",&basic,&da);
```

- Ex 4

```
int hra,da;
scanf("%d:%d",&hra,&da);
```

- Ex 5

```
int num1,num2;
scanf("%2d %5d",&num1,&num2);
```

  - 21 will be assigned to num1 and 345 will be assigned to num2 and 50 that is unread will be assigned to next scanf call

Examples Real Numbers

- Ex 1

  ```
  float x;
  scanf("%f",&x);
  ```

  - Assigns: 4.321 to x

- Ex 2

  ```
  double y;
  scanf("%lf",&y);
  ```

Examples char and string

- Ex 1

```
char name[20];
scanf("%s",&name);
```

- Ex 2

```
char name[20];
gets(name);
puts(name);
```

Rules for scanf

- Each variable must have a field specification
- For each field specification there must be variable address
- The scanf reads until
  - A white space is found in numeric specification
  - the maximum number of characters have been read
  - An error is detected
  - The end of file is reached

Formatted Output

- printf() is used for printing results
- printf("control string", arg1,arg2…..);
- Control String specifies
  - characters that will be printed on screen
  - Format Specifications
  - Escape sequence characters

Examples

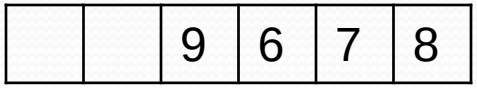
- printf(“Programming in C”);
- printf(“\n”);
- printf(“%d”,x);
- printf(“x=%d\n”,x);
- printf(“The value of a is %d”,a);

- printf does not supply new line automatically. Thus ‘\n’ is used

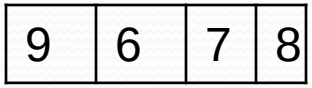
Integer Examples

- printf(“%d”,9678);

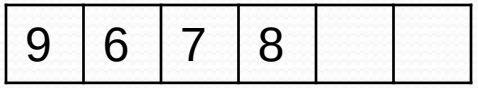
- printf(“%6d”,9678);



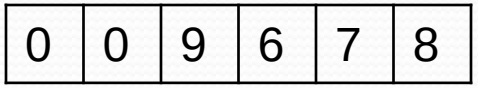
- printf(“%2d”,9678);



- printf(“%-6d”,9678);



- printf(“%06d”,9678);



Real Examples

- Syntax: %w.pf
  - w indicates the number of digits used for display
  - p indicates the number of digits to be displayed after decimal
  - Let y=98.7654;
- printf("%7.4f",y);

9	8	.	7	6	5	4

- printf("%7.2f",y);

		9	8	.	7	7

- printf("-7.2f",y);

9	8	.	7	7		

String Examples

- Syntax: %w.ps
  - w specifies width of field
  - p specifies only first p characters of string are displayed

- Ex:
  - char a[20]="Hello World";
  - printf("%s",a);

H	e	l	l	o		W	o	r	l	d									

- printf(“%25s”,a);

- printf(“%25.4s”,a);

- printf(“%-25.7s”,a);