Programming Principle & Algorithm

Class- BCA Ist Semester

Dr. Dharm Raj Singh

Assistant Professor, (HOD)

Department of Computer Application

JagatpurP. G. College, Varanasi

Mobile No. 9452070368, 7275887513

Email- dharmrajsingh67@yahoo.com

Outline

1. MODULE 1

- Unit 1 : Basics of Programming
- Unit 2 : Fundamentals
- Unit 3 : C Operators

1.1 Introduction

- Computer programming is a set of instructions through which one tells the computer to do the desired task.
- This set of instructions written in human readable computer languages called Source Code. Every program has two parts namely code and data.
- There are two models of programming namely Structured Programming and Object Oriented Programming.
- In Structured Programming codes are executed one after one in a serial fashion. Example for this is ‘C’ Language.
- In object oriented programming, data is accessed through objects. There is no single flow. Here objects freely interact with one another by passing messages.

1.2 Problem Solving Techniques

To develop the solution for the given problem, the following programming techniques are used.

Problem solving techniques

i) Algorithm

It is a set of logical procedure steps to solve the problem.

ii) Flow Charts

It is a diagrammatic representation of the sequence of operation for a given problem.

iii) Pseudo codes

These are the instructions written in ordinary English using mathematical and logical symbols to display the program logic.

Steps in Problem Solving

- First produce a general algorithm (one can use ***pseudocode***)
- Refine the algorithm successively to get step by step detailed ***algorithm*** that is very close to a computer language.
- ***Pseudocode*** is an artificial and informal language that helps programmers develop algorithms. Pseudocode is very similar to everyday English.

Pseudocode & Algorithm

- **Example 1:** Write an algorithm to determine a student's final grade and indicate whether it is passing or failing. The final grade is calculated as the average of four marks.

Pseudocode & Algorithm

Pseudocode:

- *Input a set of 4 marks*
- *Calculate their average by summing and dividing by 4*
- *if average is below 50*

    Print “FAIL”

  else

    Print “PASS”

Pseudocode & Algorithm

- Detailed Algorithm
- Step 1: Input M1,M2,M3,M4
  Step 2: GRADE ← (M1+M2+M3+M4)/4
  Step 3: if (GRADE < 50) then
  Print “FAIL”
  else
  Print “PASS”
  endif

iv) Decision Tables

A decision table consists of many independent conditions with several actions, written in table format to solve the given problem.

1.3 Algorithm

The word 'Algorithm' is the name of one Persian author meaning rules of restoration and reduction. Once the problem is analyzed, its solution is broken into a number of sample steps. A problem in a finite sequence is called an algorithm.

1.3.1 Properties of an Algorithm

- **Finiteness:** An algorithm must always terminate after a finite number of steps.
- **Definiteness:** Each step must be clearly defined that actions carried out must be unambiguous.
- **Input:** Input should be provided at the beginning of algorithm.
- **Output:** Algorithm must produce on or more output.
- **Effectiveness:** All the operations defined must be sufficiently basic that they can be done exactly in finite length of time manually.

1.3.2 Basic Statements Used and Examples

- Algorithm always begins with the word **'Start'** and ends with the word **'Stop'.**
- Step wise solution is written in distinguished steps. This is as shown in

example 1.1

Start

Step 1:

Step 2:

.

.

.

Step n:

End

- **Input Statement:** Algorithm takes one or more inputs to process. The statements used to indicate the input is Read a or Input b. **for example**

  Let a , b be the names of the Input

  Input a or Read a

  Input b or Read b

  Where a and b are variable names.

- **Output Statements:** Algorithm produces one or more outputs. The statement used to show the output is output a or print b.

  Syntax: Output variable name

  Print variable name

For example output a or print a

output b or print b

where a and b are variable names.

➢ Assignment Statements:

Processing can be done using the assignment statement.

i.e. L.H.S = R.H.S

On the L.H.S is a variable.

While on the R.H.S is a variable or a constant or an expression. The value of the variable, constant or the expression on the R.H.S is assigned in L.H.S.

- **The L.H.S and R.H.S should be of the same type**. Here **‘ = ’** is called assignment operator.

  For example Let the variables be x, y. The product be z this can be represented by as **Read x, y**

  Z = x * y

- Order in which the steps of an algorithm are executed is divided in to 3 types namely

  i) Sequential Order

  ii) Conditional Order

  iii) Iterative Order

➢ Sequential Order

Each step is performed in serial fashion i.e. in a step by step procedure for example

Write an algorithm to add two numbers.

Step 1 : Start

Step 2 : Read a

Step 3 : Read b

Step 4 : Add a , b

Step 5 : Store in d

Step 6 : Print d

Step 7 : End

Conditional Order

Based on fact that the given condition is met or not the algorithm selects the next step to do. If statements are used when decision has to be made. Different format of if statements are available they are

a) Syntax : **if** (condition)

Then {set of statements S1}

Here condition means Boolean expressions which evacuates to **TRUE** or **FALSE**. If condition is **TRUE** then the statements S1 is evaluated. If **FALSE** S1 is not evaluated Programme skips that section. For example

Write an algorithm to check equality of numbers.

Step 1 : Start

Step 2 : Read a, b

Step 3 : if a = b, print numbers are equal to each other

Step 4 : End

b) Syntax – if else (condition)

if (condition)

Then {set of statements S1}

else

Then {set of statements S2}

Here if condition evaluates to true then S1 is executed otherwise else statements are executed. For example Write an algorithm to print the grade.

Step 1 : Start

Step 2 : Read marks

Step 3 : Is marks greater than 60 ?

Step 4 : if step 3 is TRUE

print ’GRADE A’

Step 5 : Other wise

print ’GRADE B’

Step 6 : End

c) Syntax – Nested if else (condition)

If (condition 1)

Then S1

Else

If (condition 2)

Then S2

Else

Then S3

Here if and else condition is in a nested fashion this is more suited for the programs have been multiple conditions. For example

Write an algorithm to find grades of the marks.

Step 1 : Start

Step 2 : Read marks

Step 3 : Is marks greater than 60 ?

Step 4 : if step 3 is TRUE

print ’GRADE A’

Step 5 : else if marks greater than 50 less than 60

print ’GRADE B’

Step 6 : else print ’GRADE C’

Step 7 : End

iii) Iterative Order

Here algorithm repeats the finite number of steps over and over till the condition is not meet. Iterative operation is also called as looping operation. For example

Add ‘n’ natural numbers till the sum is 5.

Step 1 : Start

Step 2 : set count to 0

Step 3 : add 1 to count

Step 4 : if count is less than 5,

Repeat steps 3 & 4

Step 5 : otherwise print count

Step 6 : End

1.4 Flow Charts

Algorithm for large problems becomes complex and there by difficult to write the code. Problem analysts found ‘Flow charts’ an easier way to solve the problem. Here each step is represented by a symbol and also contains a short description of the process steps within the symbol. Flow charts are linked by arrows. The other names of flow chart are flow diagram, process chart, and business flow diagram etc., most often it is called by name flow chart only. Flow charts can be used for following aspects

- Define and analyze
- Build step by step picture of a process
- To find the areas of improvement in the process

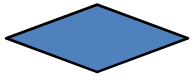
1.4.2 Symbols Used in Flowcharts

The flow chart being symbolic representation standard symbols is used for each specific operation. These symbols are used to represent the sequence of operations and flow of data and documents required for programming. Flow should be from top to bottom. The most commonly used symbols are

Start or end of the programmer

Computational steps or processing function of program

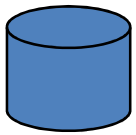
Input or output operation



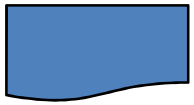
Decision making and branching

Connector or joining of two parts of program

Subroutine



Database



Document printout

Example 2

- Write an algorithm and draw a flowchart to convert the length in feet to centimeter.

Pseudocode:

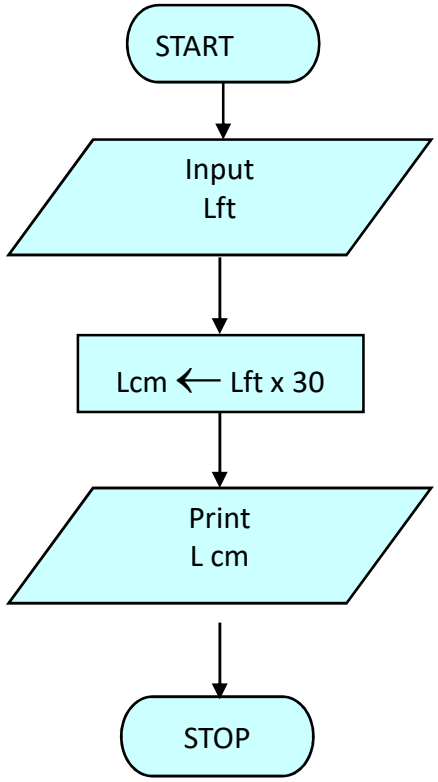
1. Input the length in feet (Lft).
2. Calculate the length in cm (Lcm) by multiplying LFT with 30.
3. Print length in cm (LCM).

Example 2

Algorithm

- Step 1: Input Lft
- Step 2: Lcm ← Lft x 30
- Step 3: Print L cm

Flowchart



DECISION STRUCTURES

- The expression A>B is a logical expression
- *it describes a **condition** we want to test*
- ***if A>B is true (if A is greater than B)** we take the action on left*
- print the value of A
- ***if A>B is false (if A is not greater than B)** we take the action on right*
- print the value of B

IF–THEN–ELSE STRUCTURE

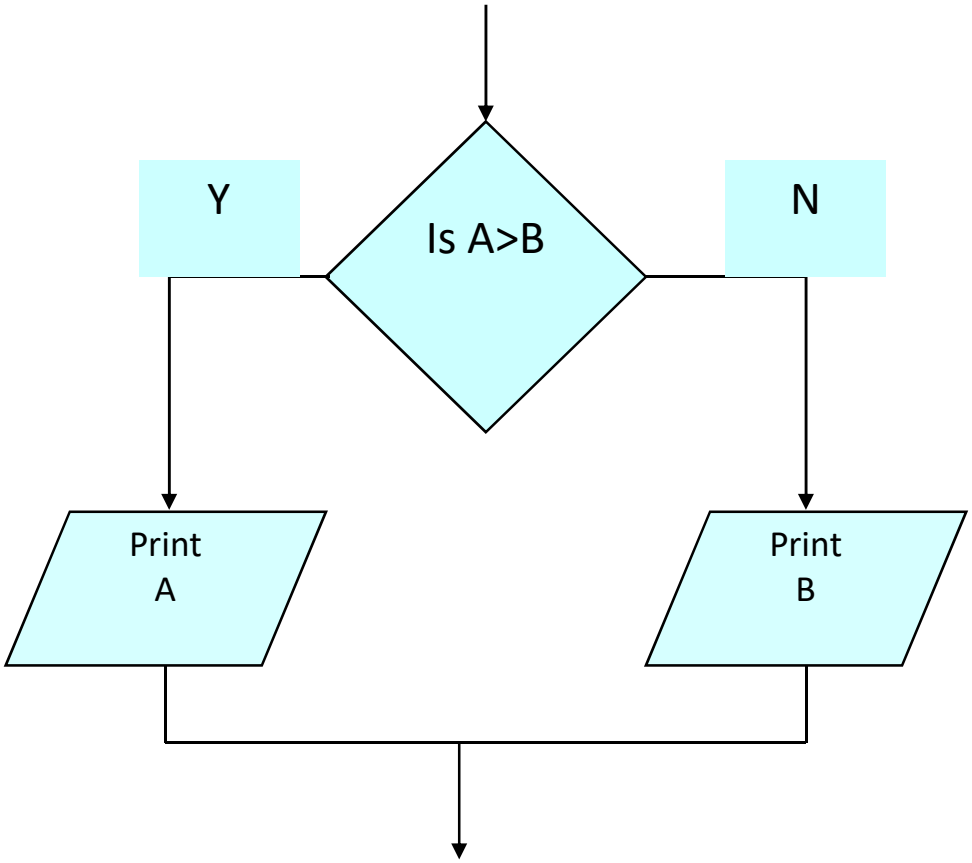
- The algorithm for the flowchart is as follows:

If A>B then

print A

else

print B



➢ Important points to remember:

- ❖ Every C program requires a main() function. Use of more than one main() is illegal. The place of main() is where the program execution begins.
- ❖ The Execution of the function begins at the opening brace and ends at the closing brace.
- ❖ C programs are written in lowercase letters. However uppercase letters may be used for symbolic names and constants.
- ❖ All the words in a program line must be separated from each other by at least one space or a tab, or a punctuation mark.
- ❖ Every statement must end with a semicolon.
- ❖ All variables must be declared for their type before they are used in the program.
- ❖ Compiler directives such as define and include are special instructions to the compiler, so they do not end with a semicolon.
- ❖ When braces are used in the program make sure that the opening brace has corresponding ending brace.
- ❖ C is a free form language and therefore proper form of indentation of various sections would improve the legibility of the program.

What is C ?

- The C programming language was developed by Dennis Ritchie in Bell Telephone Laboratories, in 1970. It was initially implemented on the system that used UNIX operating system.
- The C language is derived from the B language, which was written by Ken Thompson at AT&T Bell Laboratories. The B language was adopted from a language called BCPL (Basic Programming Language), which was developed by Martin Richards at Cambridge University.
- In 1982 a committee was formed by ANSI (American National Standards Institute) to standardize the C language. Finally in 1989, the standard for C language was introduced known as ANSI C.

Basic Structure of C programs

Documentation Section

Link Section

Definition Section

Global Declaration Section

main () Function Section
{

Declaration Part
Execution Part

}

Subprogram section

Function 1
Function 2
……..
Function n

(User - Defined functions)

```
/*  Calculating Area of Circle  */
#include<stdio.h>    /* Link Section*/
#include<conio.h>  /* Link Section*/
#define PI   3.14    /* Definition Section*/
void main()
{
  float  r, area;       /* Declaration */
  printf ("input the value of radius ");
  scanf("%f ", &r);
  area=PI*r*r;
  printf ("Area of Circle=%f is", area);
  getch();
}
```

C character set

The character set of the C language consists of basic symbols of the language. A character indicates any English alphabet, digit or special symbol including arithmetic operators.

The C language character set includes

- Letter, Uppercase A, B, …,Y, Z, Lower case a, b,...,y, z.
- Digits, Decimal digits 0….9.
- Special Characters, such as comma, period. semicolon; colon: question mark?, apostrophe‘ quotation mark “ Exclamation mark ! vertical bar | slash / backslash \ tilde ~ underscore _ dollar $ percent % hash # ampersand & caret ^ asterisk * minus – plus + <, >, (,), [,], {, }
- White spaces such as blank space, horizontal tab, new line and form feed.

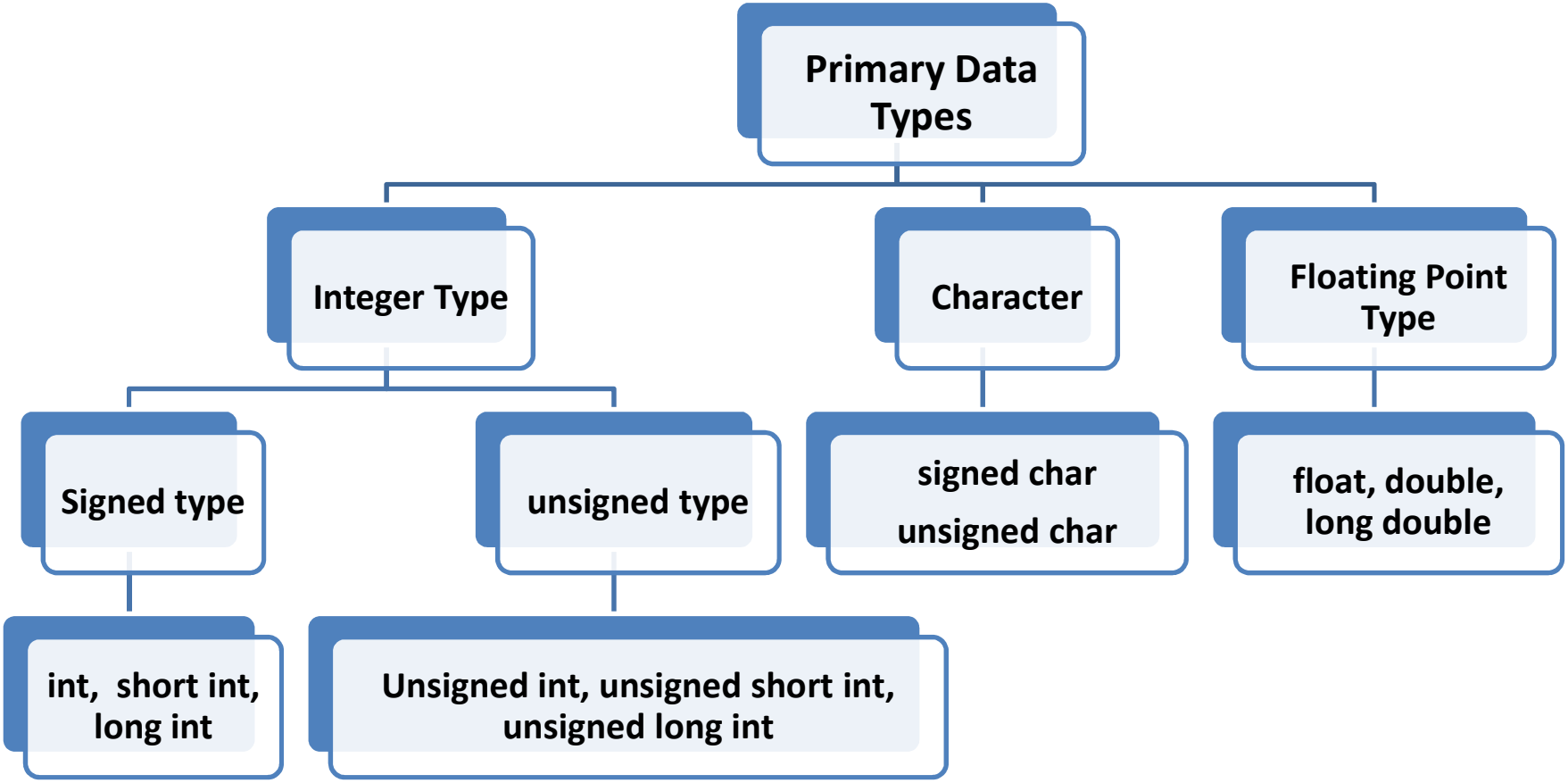
Keywords

Key words or Reserve words of the C language are the words whose meaning is already defined and explained to the C language compiler. Therefore Reserve words can not be used as identifiers or variable names. C language has 32 keywords. Following are given

auto	double	int	sturct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

Basic Data Types

In the C programming language, data types refer to an extensive system used for declaring variables or functions of different types. The type of a variable determines how much space it occupies in storage and how the bit pattern stored is interpreted.



Data Types

Size and Range of Data Types on a 16 – bit Machine			
Type	Size(bits)	Range	Remark
char or signed char	8	-128 to 127	
unsigned char	8	0 to 255	
int or signed int	16	-32,768 to 32,767	
unsigned int	16	0 to 65535	
short int or signed short int	8	-128 to 127	
unsigned short int	8	0 to 255	
long int or signed long int	32	-2,147,483,648 to 2,147,483,647	
unsigned long int	32	0 to 4,294,967,295	
float	32	3.4E-38 to 3.4E+38	6 digit for precision
double	64	1.7E-308 to 1.7E+308	14 digit for precision
long double	80	3.4E-4932 to 1.1E+4932	

Variables

- A variable is an identifier that may be used to store data value. A value or a quantity which may vary during the program execution can be called as a variable.

❖ **Rules for constructing the Variable names(identifiers)**

- They must begin with a letter and underscore is considered as a letter.
- It must consist of single letter or sequence of letters, digits or underscore character.
- Uppercase and lowercase are significant. For ex: Sum, SUM and sum are three distinct variables.
- Keywords are not allowed in variable names.
- Special characters except the underscore are not allowed.
- White space is also not allowed.

Valid Identifiers: (1) Count_2 (2) Num1

Invalid Identifiers: (1) Reg no (2) Emp-no

Declaration of Variable

- A variable declaration specifies a data type and contains a list of one or more variables of that type as follows:

Data-type v1, v2, v3, ……., vn ;
int count; int numebr, total; double ratio; char temp;

- Variables can be initialized (assigned an initial value) in their declaration.

```
Data-type variable_name = val
```

```
int d = 3, f = 5;
float z = 22.0;
```

Preprocessors

The C Preprocessor is not part of the compiler, but is a separate step in the compilation process. In simplistic terms, a C Preprocessor is just a text substitution tool and they instruct compiler to do required pre-processing before actual compilation. We'll refer to the C Preprocessor as the CPP. All pre Following section lists down all important preprocessor directives: processor commands begin with a pound symbol (#).

Directive	Description
#define	Substitutes a preprocessor macro
#include	Inserts a particular header from another file
#undef	Undefines a preprocessor macro
#ifdef	Returns true if this macro is defined
#ifndef	Returns true if this macro is not defined
#if	Tests if a compile time condition is true
#else	The alternative for #if
#elif	#else an #if in one statement
#endif	Ends preprocessor conditional
#error	Prints error message on stderr
#pragma	Issues special commands to the compiler, using a standardized method

A header file

A header file is a file with extension **.h** which contains C function declarations and macro definitions and to be shared between several source files. There are two types of header files: the files that the programmer writes and the files that come with your compiler.

You request the use of a header file in your program by including it, with the C preprocessing directive **#include** like you have seen inclusion of **stdio.h** header file. which comes along with your compiler

Include Syntax: Both user and system header files are included using the preprocessing directive #include. It has following two forms: #include <file>

#include "file"

Input Output Statements

- In C there are two types of I/O functions. Console I/O function takes Input from keyboard and produces Output on the screen.

The printf() Function:-

- Printf() is used to print or display data on the console in a formatted form.

```
printf( “control string”, list of arguments);
```

- The percentage (%) followed by conversion character is called format specifier. This indicates the type of the corresponding data item.

Format specifier	Meaning
% d or % i	decimal integers
% u	unsigned decimal integer
% x	unsigned hexadecimal (lower case letter)
% X	unsigned hexadecimal (upper case letter)
% o	octal
% c	character
% f	floating point
% s	strings
% lf	double
% ld	long signed integer
% lu	long unsigned integer
% p	displays pointer
% %	prints a % sign
% e	scientific notation (e lower case)
% E	scientific notation (e upper case)

```
/*PRINT*/
#include<stdio.h>
#include<conio.h>
void main()
{
clrscr();
int a = 5;
float d = 10.3;
printf( "the value of integer a is %d\n", a );
printf( "the value of float d is %f ", d );
getch();
}
```

The scanf() Function

- The scanf() reads the input data from standard input device. i.e. keyboard. The general format of the scanf() function is

```
scanf (“control string”, arg1, arg2, …., argn)
```

arg’s are address of variable.

Control string: %c for character, %d for integer, %f for float, %s for string, etc..

```
#include<stdio.h>        /*SCANF*/
#include<conio.h>
void main()
{ int a;
  float b;
  char c;
  printf("input the value of a, b, and c");
  scanf("%d %f %c ", &a, &b, &c );
  printf("\n %d %f %c ", a, b, c );
getch();
}
```

Formatted Integer input

Formatted input and output mans that data is entered and displayed in a particular format. Through format specifications, better presentation of result can be obtained.

%wd

Here ‘d’ is the conversion specification character for integer value and ‘w’ is an integer number specifying the maximum field width of input data. If the length of input is more than this maximum field width then the values are not stored correctly

Formatted Integer input

```
#include<stdio.h>  /*FORMATIO*/
#include<conio.h>
void main()
{ int a,b;
  printf(" \ again input value of a=6,b=394 is =");
  scanf("%2d %3d",&a,&b);
  printf("\n%d %d",a,b);              /*output a=6 b=394*/
  printf(" \n again input value of a=269,b=3845 is =");
  scanf("%2d %3d",&a,&b);
  printf("\n%d %d", a, b);          /*output a=26 b=9*/
  getch();
}
```

Formatted integer output

%wd

Here ‘w’ is an integer number specifying the minimum field width of the output data. If the length of variable is less than the specified field width, then the variable is right justified with leading blanks.

```
#include<stdio.h>    /*FORMATOP*/
#include<conio.h>
void main()
{
 clrscr();
 int a=4000,b=200,c=15;
 printf(" a=%d\n b=%d \n c=%d \n", a, b, c); /* output a=4000 b=200 c=15*/
 printf(" a=%4d\n b=%4d\n c=%4d",a,b,c);   /* output a=4000 b= 200 c=  15 */
  getch();
}
```

Format for floating point numeric input

%Wf

Here ‘w’ is the integer number specifying the total width of the input data (including the digits before and after decimal and the decimal itself). By default 6 digits are printed after the decimal point.

```
#include<stdio.h>   /FORMATFP*/
#include<conio.h>
void main()
{ float a,b;
printf(" \n input value of a=5.3,b=5.92 is =");
  scanf("%3f%4f",&a,&b);
  printf("\na=%f\nb=%f",a,b);    /* outputa=5.300000 b=5.920000 */
  printf(" \n again ninput value of a=5.93,b=65.92 is =");
  scanf("%3f%4f",&a,&b);
  printf("\na=%f\nb=%f",a,b);    /* output a=5.900000 b=3.000000 */
  getch();
}
```

Format for floating point numeric output

%w.nf

Here ‘w’ is the integer number specifying the total width of the data and n is the number of digits to be printed after decimal point. By default 6 digits are printed after the decimal point.

```
#include<stdio.h>   /*FORMATFO*/
#include<conio.h>
void main()
{ float a=8,b=5.9;

  printf("\na=%4.1f\nb=%7.2f",a,b);    /* outputa= 8.0 b=   5.90 */
   a=25.3,b=1635.92;
  printf("\na=%4.1f\nb=%7.2f",a,b);    /* output a=25.3 b=1635.92 */
  a=15.231,b=65.875948;
  printf("\na=%4.1f\nb=%7.2f",a,b);    /* output a=15.2 b=  65.88 */
  getch();
}
```

Format for string output

%w.ns

Here w is the specified field width. Decimal point and 'n' are optional. If present then 'n' specifies that only first n character of the string will be displayed and (w-n) leading blanks are displayed before string.

```
#include<stdio.h>   /*FORMATSO*/
#include<conio.h>
void main()
{
  printf("=%3s","sureshkumar");      /* output=sureshkumar */
  printf("\n=%10s","reeta");          /* output=     reeta */
  printf("\n=%.3s","sureshkumar");   /* output=sur */
  printf("\n=%8.3s","sureshkumar"); /* output=     sur */
  getch();
}
```

Unformatted I / O Function

A simple reading of data from keyboard and writing to I / O device, without any format is called unformatted I / O functions.

Character Input/Output(I/O)

In order to input and output a single character I / O functions are used. The functions under character I / O are

(I) getchar()

(II) putchar()

(III) getch()

(IV) getche()

Single Character Input/Output(I/O)

The getchar( ) function:

Single characters can be entered into the computer using the C library function getchar.

```
Character variable=getchar();
```

The putchar( ) function:

Single characters can be displayed using the C library function putchar.

```
putchar(character variable);
```

The two common alternative functions to getchar() are

1. getch() 2. getche()

- The getch() function reads a single character at a time and it waits for key press. It does not echo the character on the screen.
- The getche() is same as getch()/ but the key is echoed.

```
/*GETCHAR*/
#include<stdio.h>
#include<conio.h>
void main()
{
   char a;
   printf("input the value of a \n");
   a= getchar();
   printf("output is=");
   putchar(a);
   getch();
}
```

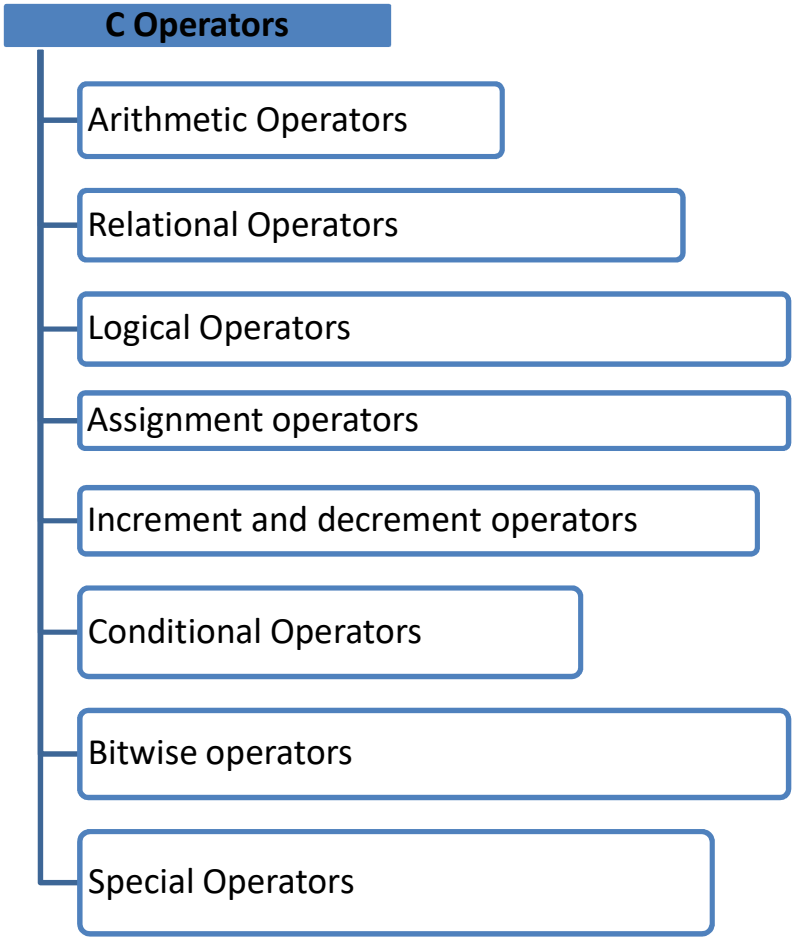
String I / O

- In order to read and write string of characters the functions gets() and puts() are used gets() function reads the string and puts() function takes the string as argument and writes on the screen.

```
#include<stdio.h>  /*GETS*/
#include<conio.h>
void main()
{
 clrscr();
 char name [50];
 puts ("Enter your name");
 gets (name);
 puts(" The name entered is ");
 puts(name);
 getch();
} Output:
Enter your name: ram
The name entered is: ram
```

operator

- An operator is a symbol which acts on operands to produce certain result as output. For example in the expression a+b; + is an operator, a and b are operands.



OPERATORS AND EXPRESSIONS

Arithmetic Operators

Operator	Meaning	Remark
+	Addition or unary plus	a + b; +a
-	Subtraction or unary minus	a – b; -a
*	Multiplication	a*b
/	Division	15/10 = 1; 15/10.0 = 1.5
%	Modulo division	14%4 = 2; 14%4.0 = wrong

```
#include <stdio.h>   /*ARITHMAT*/
#include<conio.h>
main()
{
  clrscr();
  int a = 21;
  int b = 10;
  int c ;
   c = a + b;
  printf("Line 1 - Value of c is %d\n", c );
   c = a * b;
  printf("Line 2 - Value of c is %d\n", c );
   c = a / b;
  printf("Line 3 - Value of c is %d\n", c );
   c = a % b;
  printf("Line 4 - Value of c is %d\n", c );
  getch();
}
```

OPERATORS AND EXPRESSIONS

Relational Operators

Operator	Meaning	Remark
<	is less than	a<b; value is true or false
<=	is less than or equal to	a<=b; value is true or false
>	is greater than	a>b; value is true or false
>=	is greater than or equal to	a>=b; value is true or false
==	is equal to	a==b; value is true or false
!=	is not equal to	a!=b; value is true or false

OPERATORS AND EXPRESSIONS

Logical Operators

Operator	Meaning	Remark
&&	Logical AND	(Condition1) && (Condition2) ; (a>b) && (x==10)
||	Logical OR	(Condition1) || (Condition2) ;(a>b) || (x==10)
!	Logical NOT	!(a>b)

OPERATORS AND EXPRESSIONS

Assignment Operators :

vop= exp it is equivalent to v = v op (exp); op is operator

Operator	Meaning	Remark
=	a = y;	value in y assigned to a;
+=	a+=b	a = a+b;
-=	a-=b	a = a-b;
*=	a*= n+1	a=a*(n+1);
/=	a/= n+1	a = a / (n+1)
%=	a %= b	a = a % b

OPERATORS AND EXPRESSIONS

Increment and Decrement Operators : ++, - -

Operator	Meaning	Remark
++	++m, m++	m = m+1; m +=1 For example Let i = 1; Print f (“ i = %d \n ”, i); Print f (“ i = %d \n ”, i + +); Printf(“i=%d\n”,++i); Output i=1 i=1 i = 3
--	--m, m--	m = m-1; m -=1

```
#include <stdio.h>    /*INCREMEN*/
#include<conio.h>
void main()
{
 clrscr();
 int x = 8;
 printf("\n x=%d", ++x);   /*  output x=9  */
 printf("\n x=%d", --x);    /*  output x=8  */
 printf("\n x=%d", x++);  /*  output x=8  */
 printf("\n x=%d", x--);  /*  output x=9   */
 printf("\n x=%d", x);    /*  output x=8  */
 getch();
}
```

OPERATORS AND EXPRESSIONS

Conditional Operators		
Operator	**Meaning**	**Remark**
? :	exp1?exp2 :exp3;	if (exp1) then exp2 else exp3.

```
#include<stdio.h>     /* conditional*/
#include<conio.h>
void main()
{
  clrscr();
  int a, b, max;
  printf(" input the value of a and b:");
  scanf("%d%d", &a, &b);
  max=a>b?a:b;
  printf(" larest between a and b=%d", max);
  getch();
}
```

OPERATORS AND EXPRESSIONS

Bitwise Operators		
Operator	**Meaning**	**Remark**
&	Bitwise AND	101 & 010 = 000
|	Bitwise OR	101 & 010 = 111
^	Bitwise EX-OR	111 & 010 = 101
<<	Shift left	<<100 = 001
>>	Shift right	>>100 = 010
~	Ones complement	~101 = 010

```
#include <stdio.h>   /*BITWISE*/
#include<conio.h>
void main()
{
  unsigned int a = 60;                    /* 60 = 0011 1100 */
  unsigned int b = 13;                    /* 13 = 0000 1101 */
  int c = 0;
  c = a & b;                             /* 12 = 0000 1100 */
  printf("Line 1 - Value of c is %d\n", c );
  c = a | b;                              /* 61 = 0011 1101 */
  printf("Line 2 - Value of c is %d\n", c );
  c = a ^ b;                              /* 49 = 0011 0001 */
  printf("Line 3 - Value of c is %d\n", c );
  c = ~a;                                 /*-61 = 1100 0011 */
  printf("Line 4 - Value of c is %d\n", c );
  c = a << 2;                            /* 240 = 1111 0000 */
  printf("Line 5 - Value of c is %d\n", c );
  c = a >> 2;                             /* 15 = 0000 1111 */
  printf("Line 6 - Value of c is %d\n", c );
  getch();
}
```

Comma Operators		
Operator	Meaning	Remark
,	Evaluated from left to right	value = (x=10, y=5, x+y); for (n=1, m=10; n<=m; m++, n++)

Precedence and associativity of operators

For evaluation of expressions having more then one operator, there are certain precedence and associativity rules defined in C.

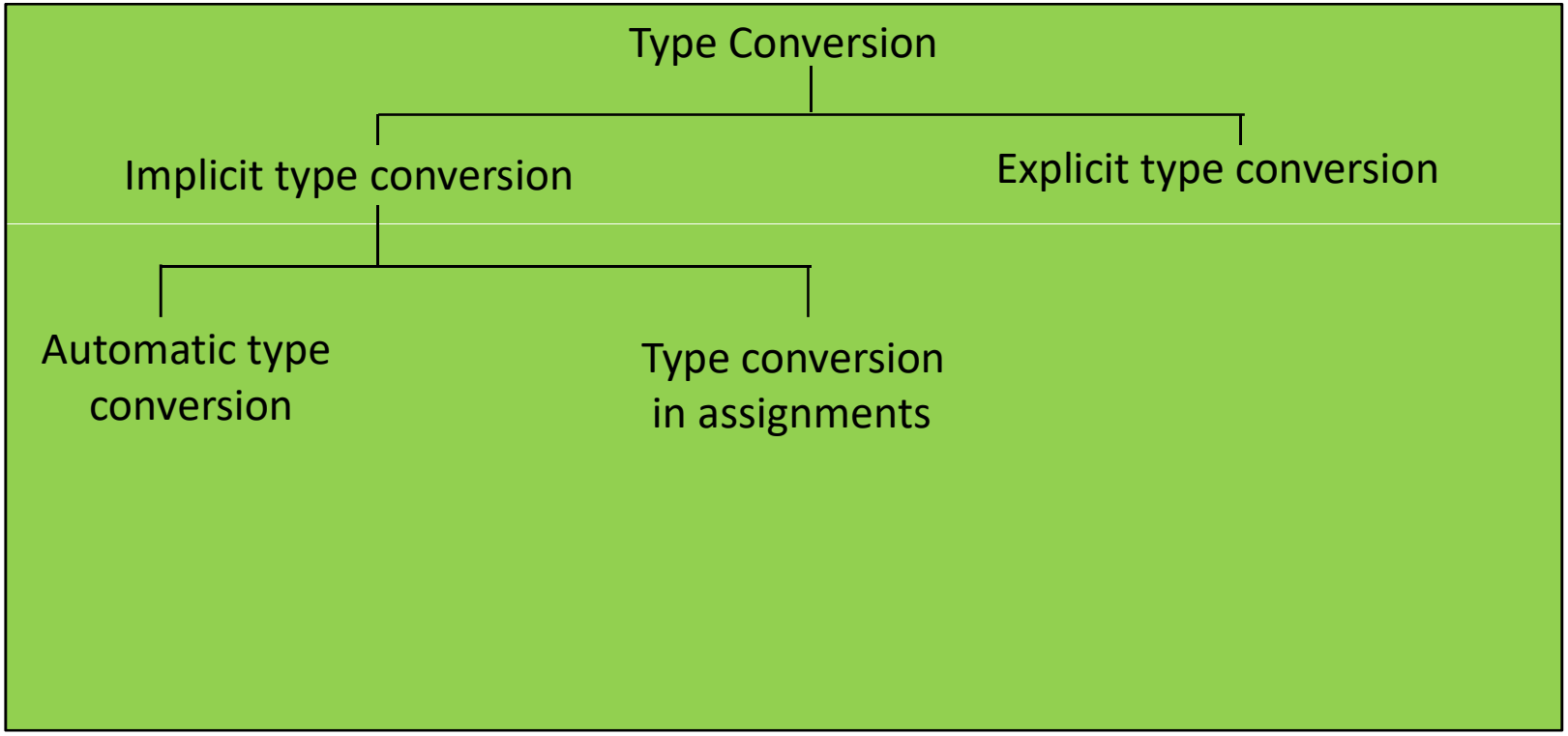
Operator category	Operators	Associativity
Postfix operators	() [] -> . ++ - -	Left to right
Unary operators	+ - ! ~ ++ - - (type)* & sizeof	Right to left
Multiplicative operators	* / %	Left to right
Additive operators	+ -	Left to right
Shift operators	<< >>	Left to right
Relational operators	< <= > >=	Left to right
Equality operators	== !=	Left to right
Bitwise AND operator	&	Left to right
Bitwise XOR operator	^	Left to right
Bitwise OR operator	|	Left to right
conditional operator	? :	Right to left
assignment operators	= += -= *= /= %=	Right to left
Comma operator	,	Left to right

OPERATORS AND EXPRESSIONS

Precedence of Arithmetic Operators		
High priority	* / %	Expression : x=a-b/3 +c*2-1 Value for a= 9, b=12, and c=3 Pass 1: x = 9-4 + 3*2-1 x = 9-4+6-1 Pass2 : x = 5+6-1 x = 11-1 x = 10
Low priority	+ -	

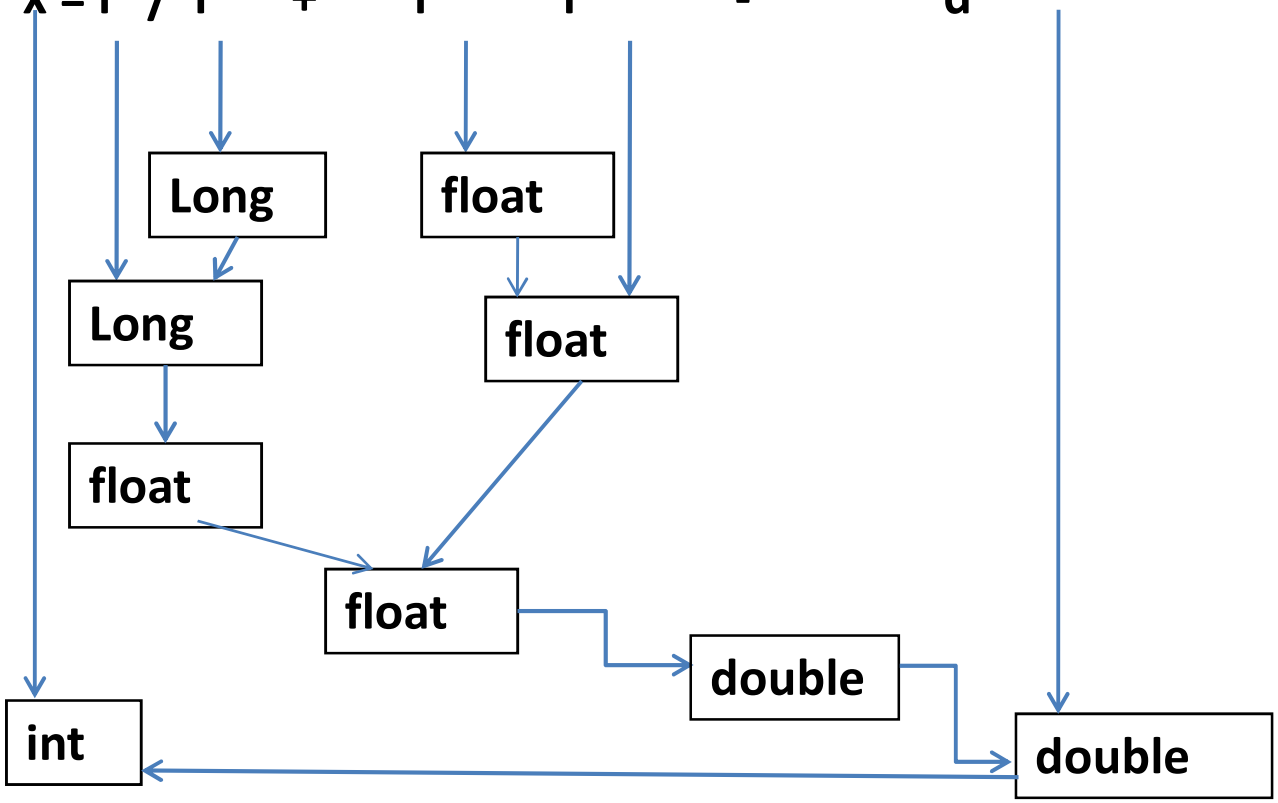
Type Conversion

C provides the facility of mixing different types of variables and constants in an expression. In these types of operations data type of one operand is converted into another data type operand. this is known as type conversion.



Automatic Data Type Conversion and Assignment conversion

```
int    i, x;
float  f;
double  d;
long int l;
 x = l   /  i     +      i   *    f          -              d
```



Explicit type conversion

Casting a Value

(type-name) expression;

```
#include<stdio.h>      /*TYPECOST*/
#include<conio.h>
void main()
{
  int a, b;
  float z;
  a= 10;
  b= 15;
  z = b/a;  /* automatic and assignment conversion*/
   printf(" z=%f ",z);
   z =  (float) b/a;      /* type casting */
    printf("\n z=%f ",z);
    getch();
}   output put
      gives  z = 1.0
                z= 1.5;
```

Exercise

1. Create Addition, Subtraction, Multiplication and Division programs using float and double data types.
2. Calculate area and circumference of circle.
3. Write a program input any character and print the ASCII number associated with it.
4. Write a program find root of quadratic equation.
5. Write a program input five subject marks then print average and percentage of marks.
6. Convert temperature from degree centigrade to Fahrenheit.
7. Find values of integers s, a and b for following code snippet with initial value of a and b equal to 1.
    1. s= ++a + b + ++a;
    2. s= ++a + b + a++;
    3. s= a++ + b + ++a;
    4. s= ++a + ++b + ++a;
    5. s= ++a + ++b + ++a;

References

- Kanetkar, Yashavant P. "Let UsC Fifth Edition." (2017).
- Kernighan, Brian W., and Dennis M. Ritchie. *The C programming language*. 2006.
- Ritchie, Dennis M., Brian W. Kernighan, and Michael E. Lesk. The C programming language. Englewood Cliffs: Prentice Hall, 1988.
- McGraw-Hill, Herbert Schildt Tata. "The Complete Reference C fourth Edition”." (2005).
- Griffiths, David, and Dawn Griffiths. Head First C: A Brain-Friendly Guide. " O'Reilly Media, Inc.", 2012.
- Programming in C-Balguruswamy

Declaration

“The content is exclusively meant for academic purpose and for enhancing teaching and learning. Any other use for economic/commercial purpose is strictly prohibited. The users of the content shall not distribute, disseminate or share it with anyone else and its use is restricted to advancement of individual knowledge. The information provided in this e-content is authentic and best as per knowledge”.

Dr. Dharm Raj Singh
Assistant Professor, (HOD)
Department of Computer Application
JagatpurP. G. College, Varanasi

Thanks