BASICS OF C PROGRAMMING

DATA TYPES & MODIFIERS & OPERATORS

PRIMARY DATA TYPES:

Integer - real numbers without decimal points (.5 – not acceptable)

Memory size of int – 4 bytes (depends on computer architecture)

Float - real numbers with decimal points - 4 bytes

Char - includes alphabets, numbers, special char

1 byte

Double – real numbers with decimal point upto 15 decimal places

8 bytes (.000000000000000)

MODIFIERS IN C:

short – limits the user to store integer value

- Takes 2 bytes of memory (32 bit architecture)
- 32767 to -32768

Signed – signed (-ve and +ve)

Unsigned – ‘ unsigned’to store only +ve value (merging both +ve and – ve range)

Long - to store large range of integer

[Time complexity:

Space Complexity:

- File size
- Program statements (Logic)
- Variables used (data to be processed)]

FORMAT SPECIFIER:

In scanf or printf statement it is used to represent the kind of data will be processed

%d – integer

%i – signed integer

%f – float or double

%e – exponential format of data - conversion specifier

%E – capital letter (E) is used in exponential form of data

%o – octal value of integer will be found - conversion specifier

%x or %X – hexadecimal number of given integer - conversion specifier

%p – pointer (address stored in a pointer)

%c – char (single char)

%s – to process a sequence of chars (string)

BINARY NUMBER SYSTEM:

2 bytes

1 2 4 8 16 32 64 128 256 512 1024 2048 4096 8192 16384 32768

1000000000000000 - -ve

0 to 32767

LENGTH MODIFIER & PRECISION MODIFIER:

LENGTH MODIFIER:

SHORT INT - %hd - length modifier

 # - modifier with flag optionwith format specifier to display hexadecimal (OX will be added with output)

Zero flag – length and precision will be used on integer data

OPERATORS

ARITHMETIC OPERATORS – arithmetic operations

ADDITION +

SUBTRACTION –

MULTIPLICATION *

DIVISION - /

MODULO - % (RETURNS REMAINDER OF THE DIVSION OPERATION)

INCREMENT OPERATORS (++)

PREINCREMENT -> ++VAR

POSTINCREMENT -> VAR++

DECREMENT OPERATOR (--)

PREDECREMENT -> --VAR

POSTDECREMENT -> VAR—

RELATIONAL OPERATOR <, >, <=, >=, !=

LOGICAL OPERATOR:

LOGICAL AND - &&,

LOGICAL OR -||

LOGICAL NOT - ! (negation operator) false->true, true->false

More than one condition to be verified – logical operator

If (Brand=”Samsung” && product = “tv”) 1 && 1 = 1

1 && 0 = 0

0 &&& 1 = 0

0 && 0 =0

Purchase

If (Brand=”Samsung” || product = “tv”)

Purchase

1 || 1 = 1

1 || 0 = 1

0 || 1 = 1

0 || 0 = 0

If(brand=”Samsung”)

If(product=”tv”)

purchase

BITWISE OPERATOR:

BIT BY BIT DATA PROCESSED IN A GIVEN VALUE

A = 60 BINARY 0011 1100 ~A = 1100 0011

B = 13 BINARY 0000 1101

&, |, ^, ~

A & B = 0000 1100 = 12

A | B = 0011 1101 = 61

A ^ B = 0011 001 (XOR OPERATION)

LEFTSHIFT AND RIGHTSHIFT OPERATOR

A<<2 1111 0000

A>>2 0000 1111

PROGRAMS DISCUSSED:

MODIFIERS & PRECISION:

```
#include<stdio.h>
int main()
{
	int x=10;
	float y = 323.456;
	printf("y = %.2f\n",y);
	printf("x = %d\n",x);
	printf("x = %05d\n",x);  //ZERO FLAG LENGTH & PRECISION
	printf("hexa of x = %#X\n",x);
	printf("hexa of x = %X",x);
}
```

MEMORY SIZE OF DATA TYPES:

```
#include<stdio.h>
int main()
{
        printf("sizeofint = %d\n",sizeof(int));
        printf("sizeofchar = %d\n",sizeof(char));
        printf("sizeoffloat = %d\n",sizeof(float));
        printf("sizeofshort = %d\n",sizeof(short));
        printf("sizeoflong = %d\n",sizeof(long));
        printf("sizeofdouble = %d\n",sizeof(double));
}
```

MODIFIERS EXAMPLE:

```
#include<stdio.h>
int main()
{
	int a=10;float b=65546.678;
	double d = 65.000072777777777;
	short signed x = 65535;
	printf("a=%d\n",a);
	printf("a=%i\n",a);
	printf("octal of a = %o\n",a);
	printf("Hexadecimal of a =%X\n",a);
	printf("short data x = %d\n",x);
	printf("b value =%E\n",b);
	printf("d = %f\n",d);
}
```

INCREMENT AND DECREMENT OPERATORS:

```
#include<stdio.h>
int main()
{
	int x=10;
	printf("x = %d\n",x);
	printf("++x = %d\n",++x);
	printf("x = %d\n",x);
	printf("x++ = %d\n",x++);
	printf("x = %d\n",x);

	printf("--x = %d\n",--x);
	printf("x = %d\n",x);
	printf("x-- = %d\n",x--);
	printf("x = %d\n",x);

}
```

BITWISE OPERATORS:

```
#include<stdio.h>
int main()
{
        int A=60, B=13;
        printf("A & B = %d\n",A&B);
        printf("A | B = %d\n",A|B);
        printf("A ^ B = %d\n",A^B);
        printf("~A = %d\n",~A);
        printf("A<<2 = %d\n",A<<2);
        printf("A>>2 = %d\n",A>>2);
        printf("A = %d",A);
}
```