CHAPTER 2 *(DAY 1 Fore noon)*

DATA TYPES IN C LANGUAGE
VARIABLES
CONSTANTS
IDENTIFIERS
VOID TYPE
NUMERIC DATA TYPES
MODIFICATIOS OF NUMERIC DATA TYPES (Reading)
SIGNED AND UNSIGNED VARIABLES
DATA TYPE CONVERSION (Reading)
CHARACTERS AND STRINGS
CHARACTER STRINGS IN C
SPECIAL CHARACTERS (Reading)
STRING HANDLING FUNCTIONS
DECLARATION OF VARIABLES (Reading)
USER DEFINED TYPE DECLARATIONS (Reading)
KEYWORDS (Reading)
VOLATILE (Reading)
C STORAGE CLASSES
DEFINING SYMBOLIC CONSTANTS
DECLARING VARIABLE AS CONSTANT
PRINTING AND FORMATTING VARIABLES (Reading)
SIMPLE PROGRAMS IN C

CHAPTER – 2

DATA TYPES IN C LANGUAGE (Reading)

MODIFICATIONS OF NUMERIC DATA TYPES

The three data types above have the following modifiers.

- **short**
- **long**
- **signed**
- **unsigned**

The modifiers define the amount of storage allocated to the variable.

Type	Bytes	Bits	Range
short int	2	16	-32,768 -> +32,767 (32kb)
unsigned short int	2	16	0 -> +65,535 (64Kb)
unsigned int	4	32	0 -> +4,294,967,295 (4Gb)
int	4	32	-2,147,483,648 -> +2,147,483,647 (2Gb)
long int	4	32	-2,147,483,648 -> +2,147,483,647 (2Gb)
signed char	1	8	-128 -> +127
unsigned char	1	8	0 -> +255
float	4	32	
double	8	64	
long double	12	96	

DATA TYPE CONVERSION

An operator must have operands of the same type before it can carry out the operation. Because of this, C will perform some automatic conversion of data types.

These are the general rules for binary operators (* + / % etc):

- If either operand is long double the other is converted to long double.
- Otherwise, if either operand is double the other is converted to double
- Otherwise, if either operand is float the other is converted to float
- Otherwise, convert char and short to int
- Then, if an operand is long convert the other to long.

SPECIAL CHARACTERS

Special Characters

The following special patterns are used to represent a single character in C programs. The leading backslash in the single quotes indicates that more information is to follow.

C code	Meaning
'\014'	Bit pattern for Form Feed
'\n'	Newline
'\t'	Tab
'\\ '	Backslash
'\''	Single Quote
'\"'	Double Quote
'\b'	Backspace
'\r'	Carriage Return
'\f'	Form Feed
'\0'	NULL (String Terminator)

DECLARATION OF VARIABLES

Every variable used in the program should be declared to the compiler. The declaration does two things.

The general format of any declaration

data type v1, v2, v3, …….. vn;

Where v1, v2, v3 are variable names. Variables are separated by commas. A declaration statement must end with a semicolon.

Example:

```
int sum;
int number, salary;
double average, mean;
```

USER DEFINED TYPE DECLARATION

In C language a user can define an identifier that represents an existing data type. The user defined data type identifier can later be used to declare variables. The general syntax is

typedef **type** identifier;

here type represents existing data type and ‘identifier’ refers to the ‘row’ name given to the data type.

Example:

```
typedef  int salary;
typedef  float average;
```

KEYWORDS

An alphabetical summary of each of the keywords is as follows:

auto	**double**	**int**	**struct**
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**

VOLATILE

Indicates that a variable can be changed by a background routine:
Keyword volatile is an extreme opposite of const. It indicates that a variable may be changed in a way which is absolutely unpredictable by analyzing the normal program flow (for example, a variable which may be changed by an interrupt handler). This keyword uses the following syntax:
volatile data-definition;

Every reference to the variable will reload the contents from memory rather than take advantage of situations where a copy can be in a register.

PRINTING AND FORMATTING VARIABLE TYPES

Printing Out and Inputting Variables:

C uses formatted output. The printf function has a special formatting character (%) -- a character following this defines a certain format for a variable:

%c -- characters
%d -- integers
%f – floats

printf (``%c %d %f", ch, i, x);

Format statement enclosed in ``...", variables follow after. Make sure order of format and variable data types match up.

scanf () is the function for inputting values to a data structure: Its format is similar to printf:

scanf (``%c %d %f", &ch, &i, &x);

example:

```
#include <stdio.h>

int main(void)
{
        int a = 1023;               /* simple integer type */
        char c = ‘a’;                /* character type */
        char s[] = “Hello”;       /* string up to 256 characters */
        /* 'initializing' : truncation from 'const double' to 'float' */

        float f = 3.14159;        /* float point type */

        printf (“a = %d\n”, a);       // decimal output
        printf (“c = %c\n”, c);       // ASCCII string output
```

```
        printf ("s = %s\n", s);       // ASCII string output
        printf ("f = %f\n", f);        // floating output
        printf ("a = %7d\n", a);    // use a field width of 7
        printf ("a = %-7d\n", a);  // left justify in a field of 7
        printf ("f = %.3f\n", f);    // use 3 decimal places

        return 0;
}
```

Output of the above program:

a = 1023
c = a
s = Hello
f = 3.141590
a =   1023
a = 1023
f = 3.142

```
#include <stdio.h>
int main(void)
{
        int sum = 100;
        char letter = ‘Z’;
        float set1 = 23.567;
        double num2 = 11e+23;

        printf (“ integer value is %d\n”, sum);
        printf (“ float variable is %f\n”, set1);
        printf (“ character is %c\n”, letter);
        printf (“ double variable is %e\n”, num2);

        return 0;
}
```

example output:

integer variable is 100
Character variable is Z
Float variable is 23.567000
Double variable is 11.000000e23