CHAPTER - 2
DATA TYPES IN C LANGUAGE

CHAPTER 2

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

CHAPTER - 2
DATA TYPES IN C LANGUAGE

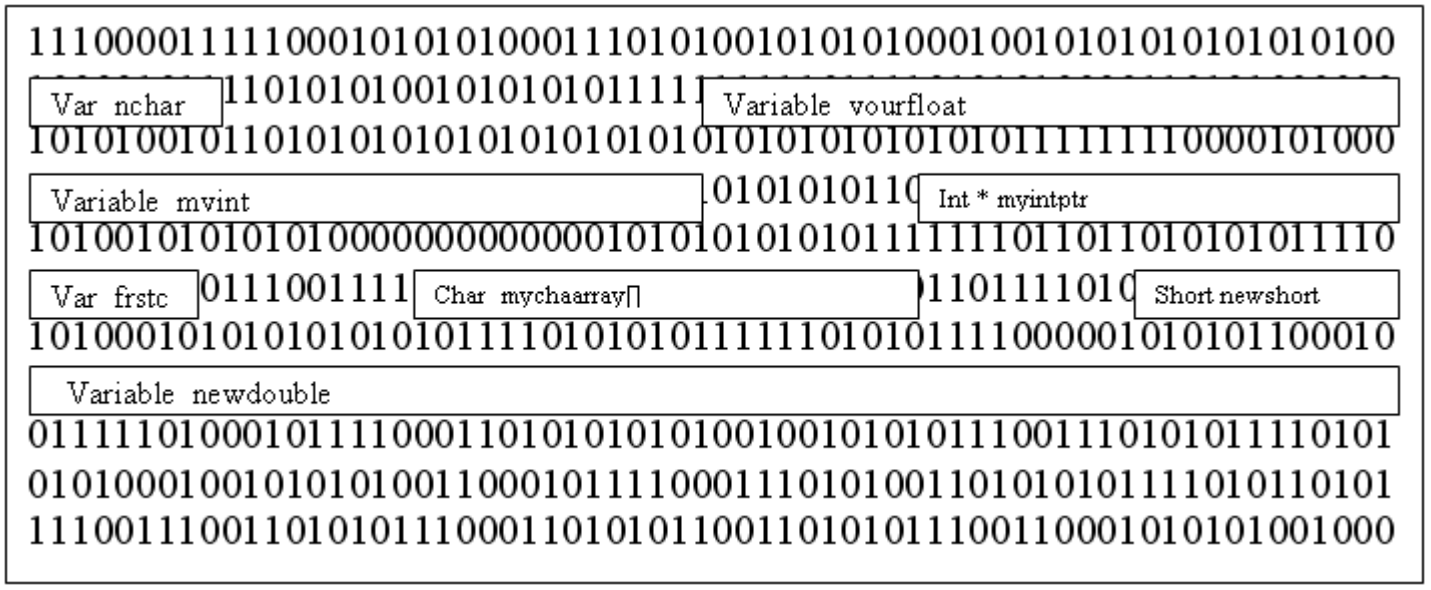
CHAPTER 2

DATA TYPE CONVERSION **(Reading)**
DECLARATION OF VARIABLES **(Reading)**
USER DEFINED TYPE DECLARATIONS **(Reading)**
KEYWORDS **(Reading)**
VOLATILE **(Reading)**
C STORAGE CLASSES
DEFINING SYMBOLIC CONSTANTS
PRINTING AND FORMATTING VARIABLES **(Reading)**
SIMPLE PROGRAMS IN C

DATA TYPES IN C LANGUAGE

VARIABLES/CONSTANTS/IDENTIFIERS

- **Variables are place holders for a data type.**
- **Variables have to be defined before they are used.**



DATA TYPES IN C LANGUAGE

VARIABLES/CONSTANTS/IDENTIFIERS

- **Sample partial memory in the course material displays some variables how they are placed and used.**
- **Constants are similar to variables but do not change their value during program execution.**
- **Identifiers are names given to variables and other program elements.**

VOID TYPE IDENTIFIERS

void *called_function* (argument data types);

- **void is a data type that does not belong to any specific data type.**
- **When a function does not return anything void is returned.**

DATA TYPES IN C LANGUAGE

NUMERIC DATA TYPE

```
int Count;
float miles;
```

- Integer data types are int, short and long.
- int data type takes 32 bits of length, short will take 16 bits and long will take 64 bits.
- There are signed and unsigned integers with the highest bit is set to ‘1’ or ‘0’.
- Real numbers are float and double.
- float will take 32 bits and double 64 bits.

DATA TYPES IN C LANGUAGE

CHARACTERS AND STRINGS

- **Character data types are declared as char before the variable name.**
- **Character constants are represented in 8 bits. There are 256 character sets.**
- **Characters are also 8 bit integer values.**
- **Character strings are enclosed in double quotes and single characters are within single quotes.**
- **There is a list of string handling functions in the course material to *copy*, *compare* and *concatenate* strings.**
- **Special characters are given with escape characters with a special meaning.**
- **'\0' null character, '\b' back space, '\a' alert, '\f' form feed, '\n' new line, and '\t' for tab.**

DATA TYPES IN C LANGUAGE

TYPEDEF

typedef previous_data_type *new_data_type*

- **typedef can be used with a previous declaration and current declaration.**
- **The advantage of typedef is that the programmer does not have to declare the whole long declaration each time.**

C STORAGE CLASSES

auto data_type *variable_name*

- **auto storage class has variables defined with in the scopes of a function, file or a block.**
- **Storage for the variables is automatically allocated and de-allocated after the closing scopes or end of file. auto keyword before the variable declaration is optional.**
- **Initialization of auto variable is not done at compile.**

DATA TYPES IN C LANGUAGE

C STORAGE CLASSES

register data_type ***variable_name***

- **Register classes are used for fast calculations and accumulators with a keyword register before the variable declaration.**

static data_type ***variable_name***

- **Static is an interesting storage class declared with static keyword before the declaration with file scope or function scope.**
- **The static variable is initialized at compile time with a default initial value and retains the value.**
- **Subsequent updates of static variables are retained with the updated values.**

DATA TYPES IN C LANGUAGE

C STORAGE CLASSES

extern data_type *variable_name*

- **Extern variables are used a keyword extern before the variable declaration.**
- **Extern variables are declared in another file.**

firstfile

```
#define MAXSTUD 100
int grades[MAXSTUD];
float reverse (int stdgr[], int length)
{
        ……..
}
end of firstfile
```

secondfile

```
extern int grades[];
float avrg = average (grades[], size);
printgrades (grades[])

float average (int stdgr[], int length)
{
        ……..
}
end of secondfile
```

DATA TYPES IN C LANGUAGE

DEFINING SYMBOLIC CONSTANTS

- **A preprocessor statement and used in the program as any other constant value.**
- **Symbolic constants are defined before it is referenced in the program.**
- **The general form of a symbolic constant is**

 # define *symbolic_name* value of constant

valid examples of symbolic constant definitions:

 # define marks 100
 # define pi 3.14159

DATA TYPES IN C LANGUAGE

DECLARING ING VARIABLE CONSTANTS

- **A preprocessor statement and used in the program as any other constant value.**
- **Symbolic constants are defined before it is referenced in the program.**
- **The general form of a symbolic constant is**

 # define *symbolic_name* value of constant

valid examples of symbolic constant definitions:

 # define marks 100
 # define pi 3.14159

DATA TYPES IN C LANGUAGE

SIMPLE PROGRAMS IN C

- ***Preprocessor directives.***
- ***Global declarations*.**
- ***Main start function***
- ***Main declarations.***
- ***Statements***
- ***Return type.***
- ***Prototypes* are function declarations as signatures of the functions.**
- ***Prototypes* are declared with the return type, function name and the function arguments before they are used.**
- ***System calls*.**

DATA TYPES IN C LANGUAGE

SIMPLE PROGRAMS IN C

```
#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;
}
```

DATA TYPES IN C LANGUAGE

SIMPLE PROGRAMS IN C

Output of the above program:

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