Brief Introduction to C Internals

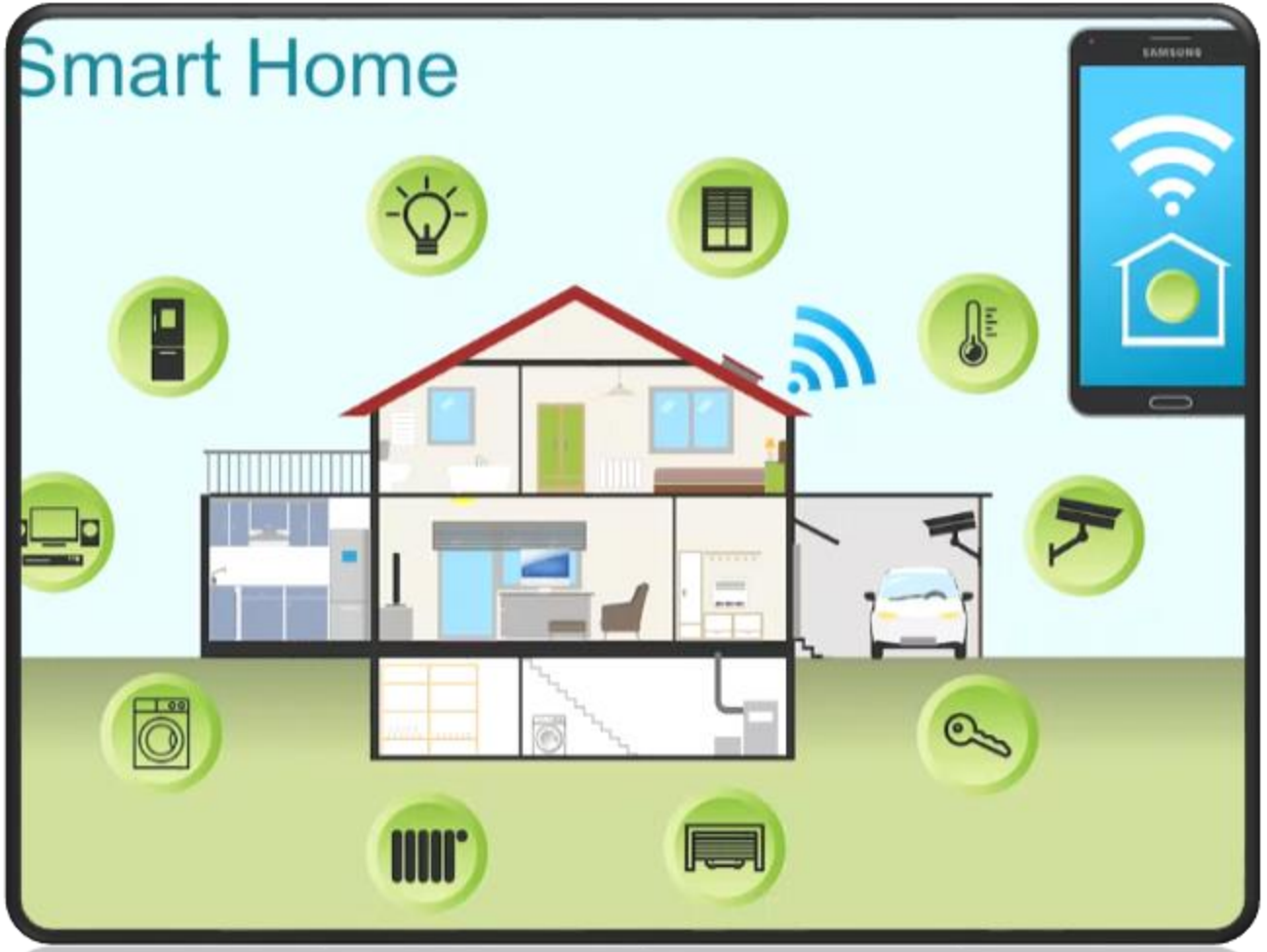
Organised & Supported by **RuggedBOARD**

Agenda

- Home Automation
- Goal of this Internship
- Why C program?
- Structure of C program
- I/O instructions
- Compiling and Running “Hello world “ example
- Basic Data Types
- Derived Data Types
- Data Types & Range
- Variable

Home Automation

What are the components need to make Smart Home?



Home Automation

1.Wifi internet connection
2.Need to control my AC
3.Sensors
4.Cameras
5.Security system
6.Microcontroller
7.Customised Embedded board
8.Automatic gate lock and open
9.Through Mobile need to access
10.Microwave oven
11.Health gadgets

With one device where I can control all the system in Home.
Example:SBC (Single Board Computer)

Goal of this Internship

To achieve this we need to gain knowledge about those controling devices and to interface those devices.
Example:Communictions protocols like UART,SPI,I2C ,Mqtt,Webserver etc
Most importantly programming languages like C , Python

End of this Eight week intership all must complete atleast one project

Note: In this course all above requirements are covered in eight weeks

Why C program?

- Being a middle-level language, C reduces the gap between the low-level and high-level languages.
- It is used for writing operating systems (OS) as well as doing application level programming.
- We can do inline assembly code inside C program to access registers of the processor.
- C is extensively used in Embedded Programming.
  [Embedded Programming is also referred to as micro-controller programming, where C program is used to control micro-controllers. Microcontrollers and embedded programming is widely used in auto-motives, Robotics, Hardware etc]

Structure of C-Program

A C program basically consists of the following parts –

- Preprocessor Commands
- Functions
- Variables
- Statements & Expressions
- Comments

```
#include <stdio.h>                    //Preprocessor Commands
int main()                            // Program execution begins
{
      /* my first program in C */     //comments
      printf("Hello, World! \n");     // Display the output in screen
      return 0;                       //Program termintaes hetre
}
```

Header file:

- It is a pre-defined program.
- It contains function (Sub programs), variables and constants etc.

Syn: **# include <headerfile>**

Here, the symbol **'#'** reps. pre-processor.
The word **"include"** is a system code.

main ()

Here, the word main is followed by a pair of parentheses ().
That represents a function.

Each and Every C program must start its execution from this point only.

This is defined by programmer only. so, it is user – defined function.

I/O Instructions

To read data from key board (std. input device) using a statement. That is called "Input instruction".

To display the information on the monitor (std. output device) using a statement. That is called "OutputInstruction".

Syn: scanf ( ) ; //This is the standard input instruction in C.
printf ( ) ; //This is the standard output Instruction in C.

I/O Instructions – scanf printf

It is a Library function.
It is defined in the header file 'stdio.h'.
It can read any type of data from KB (Std. input device).
It is the standard input function in C language.

Syn : scanf (" formatting chars " , list of vars);
Ex: scanf(" %d " , &num);

It is a Library function.
it is defined in the header file 'stdio.h'.
It can display any type of messages on the monitor only.

Syn: printf("Formatting string " , List of variables);

Ex: printf(" Roll No = %d Name = %s Average = %f " , no, name, per);

Compiling and Running “Hello world “ example

```
#include <stdio.h>  //Header
Int main()
{
    printf("Hello World!\n");
}
```

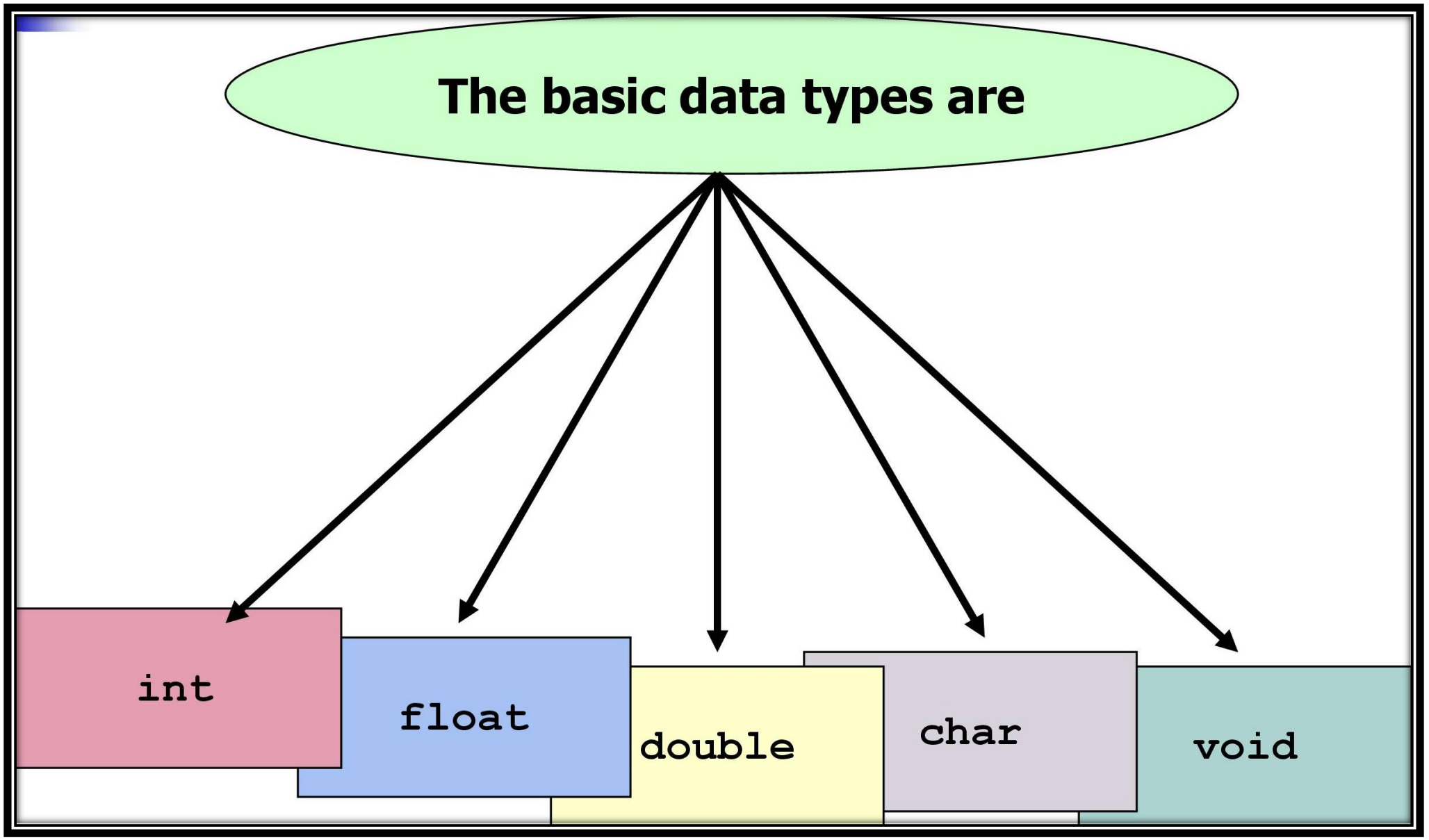
Compiling and Running on Linux:

$ gcc test.c
The resulting executable file is a.out file.
To run this executable you must type:
$ ./a.out

Rugged Board

Program execution

Basic data type



Basic Data Types

Data types :

A data type describes the kind of data that will fit into the name of the variable is preceded with the data type .

Syntax : data type variblename

Ex: int varName

The basic data types

Int , float , char , double , void

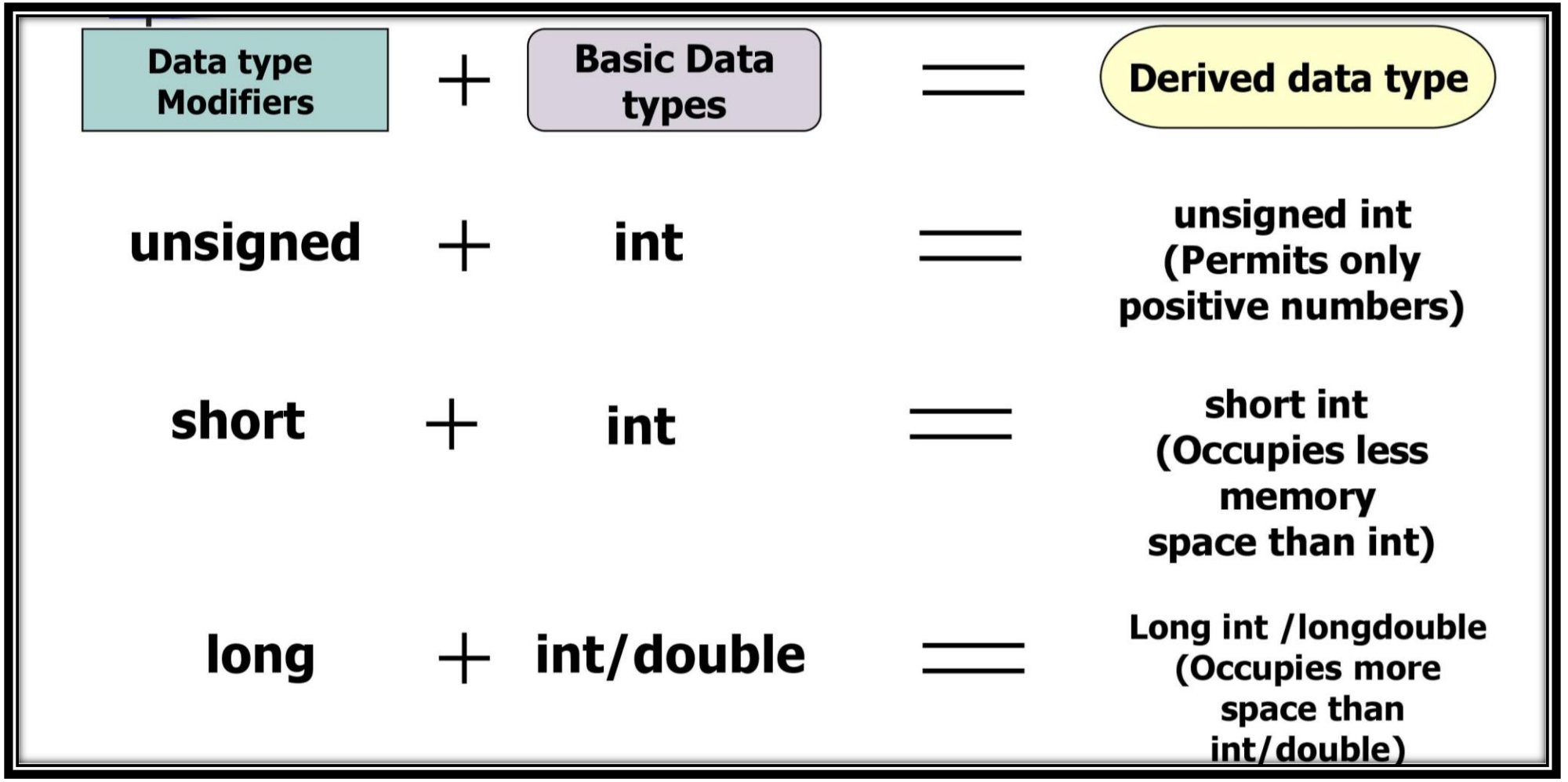
Type int ,float, char

Stores numeric data
int num;
[Cannot store any other type of data like "Alan" or "abc"]
32 bits (4 bytes)
Integers in the range -65545 to 65545
Examples: 12322, O, -232

Stores values containing decimal places
float num;
Precision of upto 6 digits
32 bits (4 bytes) of memory
Examples: 23.05, 56.5, 32

Stores a single character of information
char gender ;
gender= 'M'
8 bits (1 byte) of memory
Examples: 'a', 'm','$','%','1','5'...
Type void
Stores nothing
[Indicates the compiler that there is nothing to expect]

Derived data type



Signed , Unsigned ,long and short types

unsigned type specifies that a variable can take only positive values
unsigned int varNum;
varNum=23123;
varNum is allocated 2 bytes
modifier may be used with the int and float data types
unsigned int supports range from 0 to 65535

signed type specifies that a variable can take positive & negative values
int varNum;
varNum=-12;
varNum is allocated 2 bytes
modifier may be used with the int and float data types
unsigned int supports range from -32,768 to +32767

A short int occupies 8 bits (1 byte)
allows numbers in the range -128 to 127
long int occupies 32 bits (4 bytes)
long double occupies 128 bits (16 bytes)

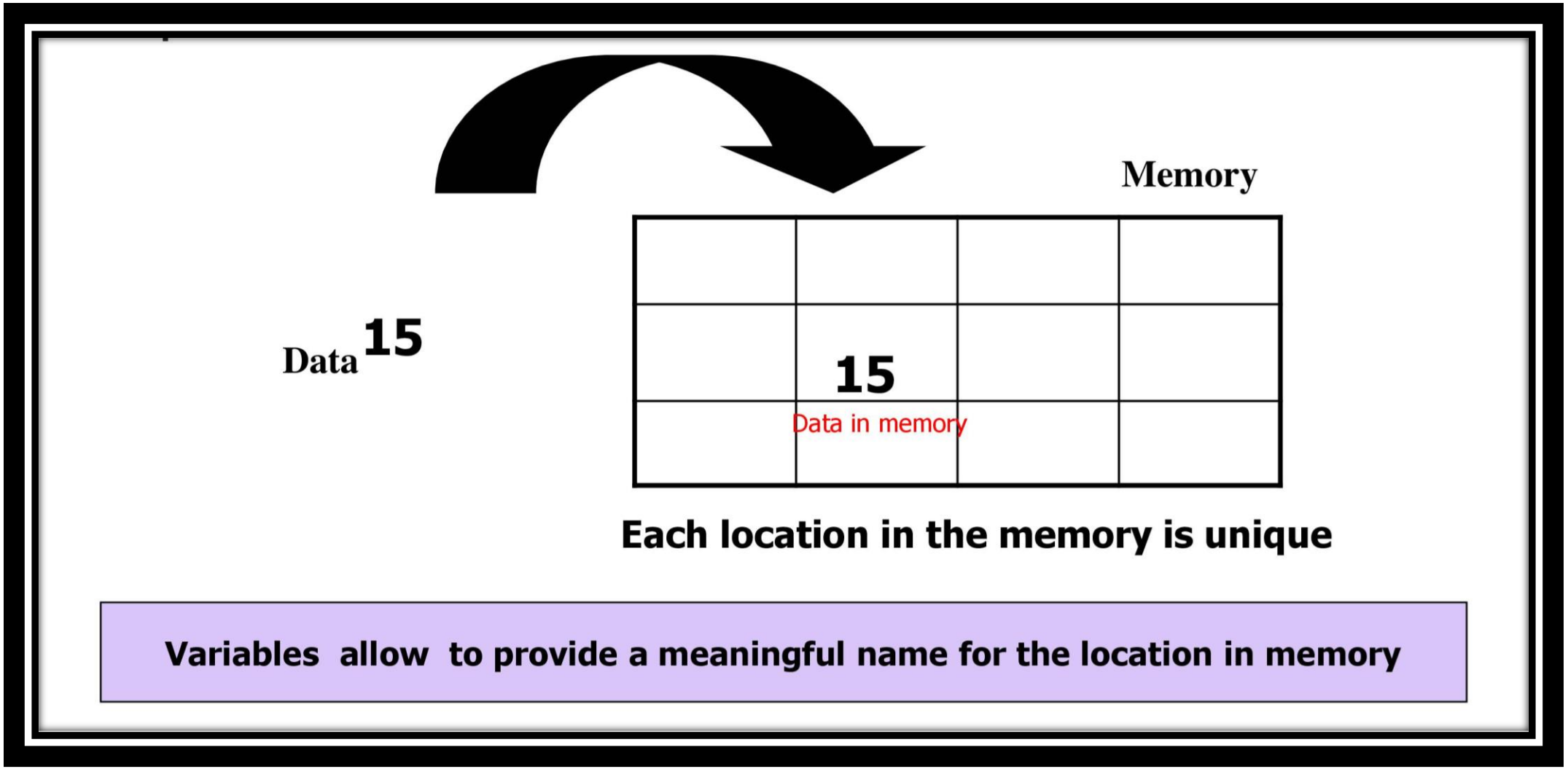
data types & range

Type	Approximate Size in Bits	Minimal Range
char	8	-128 to 127
unsigned	8	0 to 255
signed char	8	-128 to 127
int	16	-32,768 to 32,767
unsigned int	16	0 to 65,535
signed int	16	Same as int
short int	16	Same as int
unsigned short int	8	0 to 65, 535

data type & range

Type	Approximate Size in Bits	Minimal Range
signed short int	8	Same as short int
signed short int	8	Same as short int
long int	32	-2,147,483,647 to 2,147,483,647
signed long int	32	0 to 4,294,967,295
unsigned long int	32	0 to 4,294,967,295
float	32	Six digits of precision
double	64	Ten digits of precision
long double	128	Ten digits of precision

Variables


Memory
Data 15
15
Data in memory
Each location in the memory is unique
Variables allow to provide a meaningful name for the location in memory

Variables

Rugged Board

- Variable names should begin with an alphabet
- The first character can be followed by alphanumeric characters
- Proper names should be avoided while naming variables
- A variable name should be meaningful and descriptive
- Confusing letters should be avoided
- Some standard variable naming convention should be followed while programming

Sample-Declaration

```
main ()
{
   char abc;        /*abc of type character */
   int xyz;    /*xyz of type integer */
   float length;  /*length of type float */
   double area;   /*area of type double */
   long liteyrs;  /*liteyrs of type long int */
   short arm;       /*arm of type short integer*/
}
```

Rugged Board

Thank You