SNS COLLEGE OF TECHNOLOGY

Coimbatore-35
An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A+' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF INFORMATION TECHNOLOGY

PROGRAMMING FOR PROBLEM SOLVING

I YEAR - I SEM

UNIT 4 – FUNCTIONS AND POINTERS

TOPIC 3 – Pointers

INTRODUCTION

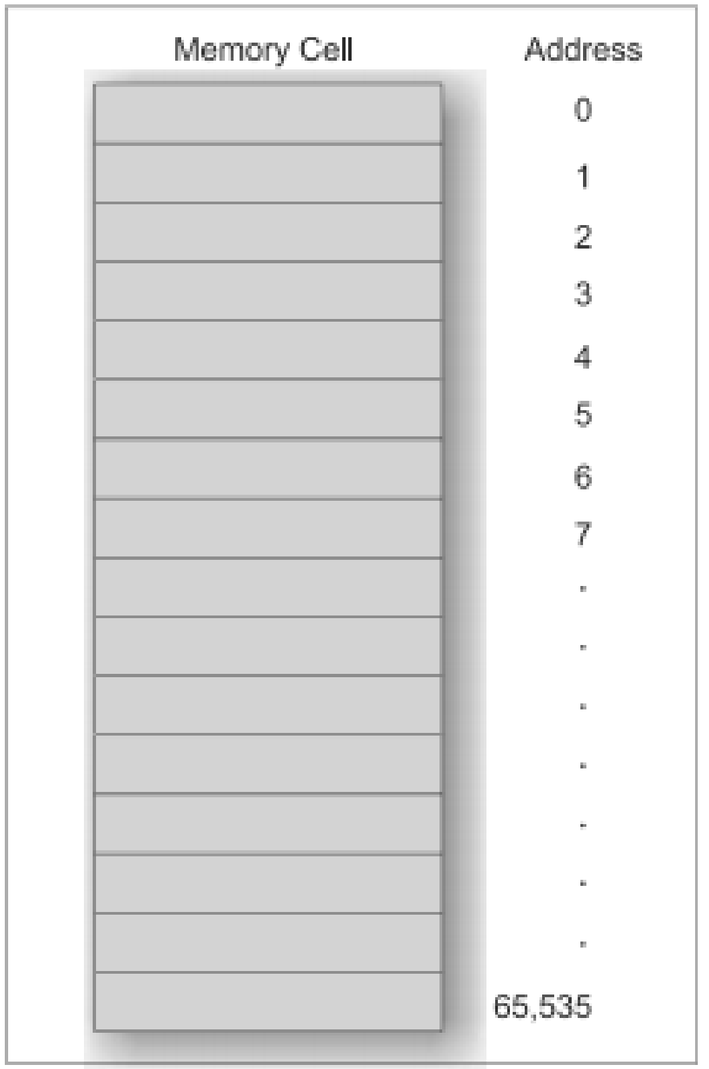
- A pointer is a derived data type in C.
- It is built from one of the fundamental data types available in C.
- Pointers contain memory addresses as their values.
- Since these memory addresses are the locations in the computer memory where program instructions and data are stored, pointers can be used to access and manipulate data stored in the memory.
- It has added power and flexibility to the language.
- Although they appear little confusing and difficult to understand for a beginner, they are a powerful tool and handy to use once they are mastered.
- Pointers are used frequently in C, as they offer a number of benefits to the programmers.

BENEFITS OF POINTERS

- 1. Pointers are more efficient in handling arrays and data tables.
- 2. Pointers can be used to return multiple values from a function via function arguments.
- 3. Pointers permit references to functions and thereby facilitating passing of functions as arguments to other functions.
- 4. The use of pointer arrays to character strings results in saving of data storage space in memory.
- 5. Pointers allow C to support dynamic memory management.
- 6. Pointers provide an efficient tool for manipulating dynamic data structures such as structures, linked lists, queues, stacks and trees.
- 7. Pointers reduce length and complexity of programs.
- 8. They increase the execution speed and thus reduce the program execution time

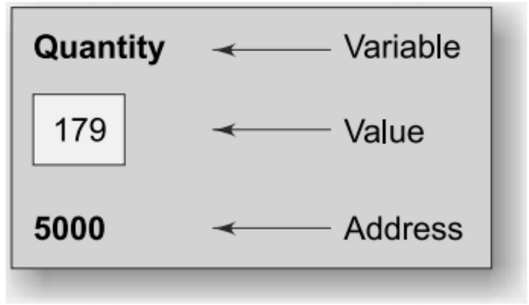
UNDERSTANDING POINTERS

- The computer's memory is a sequential collection of storage cells as shown in Fig
- Each cell, commonly known as a byte, has a number called address associated with it.
- Typically, the addresses are numbered consecutively, starting from zero.
- The last address depends on the memory size.
- A computer system having 64 K memory will have its last address as 65,535 .



UNDERSTANDING POINTERS

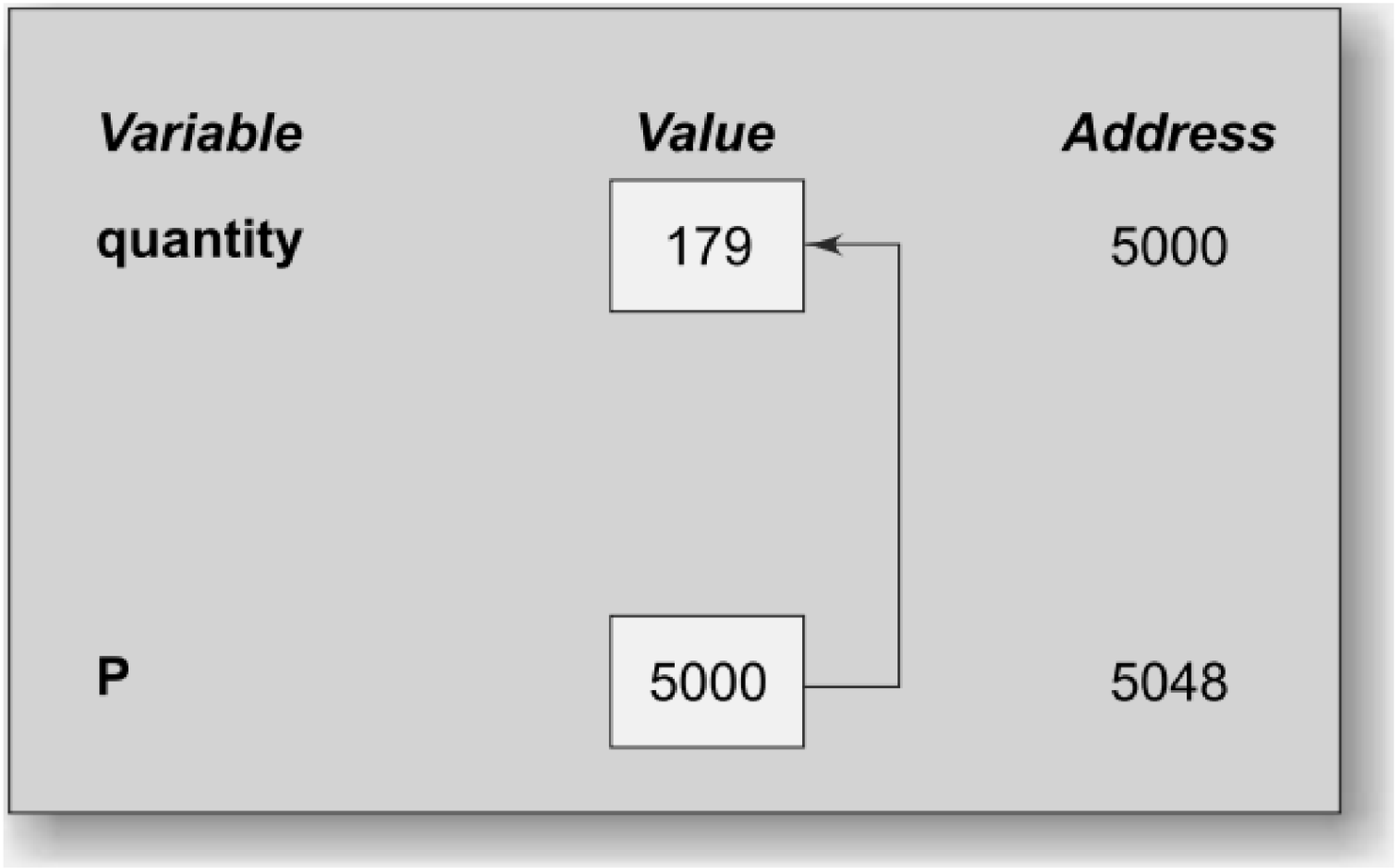
- Whenever we declare a variable, the system allocates, somewhere in the memory, an appropriate location to hold the value of the variable.
- Since, every byte has a unique address number, this location will have its own address number.
- Consider the following statement
  - **int quantity = 179;**
- This statement instructs the system to find a location for the integer variable quantity and puts the value 179 in that location.
- Let us assume that the system has chosen the address location 5000 for quantity.



UNDERSTANDING POINTERS

- During execution of the program, the system always associates the name quantity with the address 5000.
- (This is something similar to having a house number as well as a house name.)
- We may have access to the value **179** by using either the name **quantity** or the address 5000.
- Since memory addresses are simply numbers, they can be assigned to some variables, that can be stored in memory, like any other variable.
- Such variables that hold memory addresses are called **pointer variables**.
- A pointer variable is, therefore, nothing but a variable that contains an address, which is a location of another variable in memory.
- Remember, since a pointer is a variable, its value is also stored in the memory in another location.
- Suppose, we assign the address of quantity to a variable p.
- The link between the variables p and quantity can be visualized as shown in Fig
- The address of p is 5048.

UNDERSTANDING POINTERS



Pointer variable

UNDERSTANDING POINTERS

- Pointers are built on the three underlying concepts as illustrated in fig:
- Memory addresses within a computer are referred to as **pointer constants**.
- We cannot change them; we can only use them to store data values.
- They are like **house numbers**.
- We cannot save the value of a memory address directly.
- We can only obtain the value through the variable stored there using the address operator (&).
- The value thus obtained is known as **pointer value**.
- The pointer value (i.e. the address of a variable) may change from one run of the program to another.
- Once we have a pointer value, it can be stored into another variable.
- The variable that contains a pointer value is called a **pointer variable**.

ACCESSING THE ADDRESS OF A VARIABLE

- The actual location of a variable in the memory is system dependent and therefore, the address of a variable is not known to us immediately.
- How can we then determine the address of a variable?
- This can be done with the help of the operator **&** available in C.
- We have already seen the use of this address operator in the scanf function.
- The operator **&** immediately preceding a variable returns the address of the variable associated with it.

- For example, the statement **p = &quantity;**
- would assign the address 5000 (the location of quantity) to the variable **p**.
- The & operator can be remembered as ‘address of’.

ACCESSING THE ADDRESS OF A VARIABLE

- The **&** operator can be used only with a simple variable or an array element.

- The following are **illegal** use of address operator:

- 1. &125 (pointing at constants).

- 2. int x[10];
- &x (pointing at array names).

- 3. &(x+y) (pointing at expressions).