SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

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 ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

COURSE NAME : 23ITT101- PROBLEM SOLVING & C PROGRAMMING

I YEAR /II SEMESTER

Unit 4- FUNCTIONS AND POINTERS

Topic 7: Pointer: Pointer operation-Pointer arithmetic

Brain Storming

1. How to access memory location?

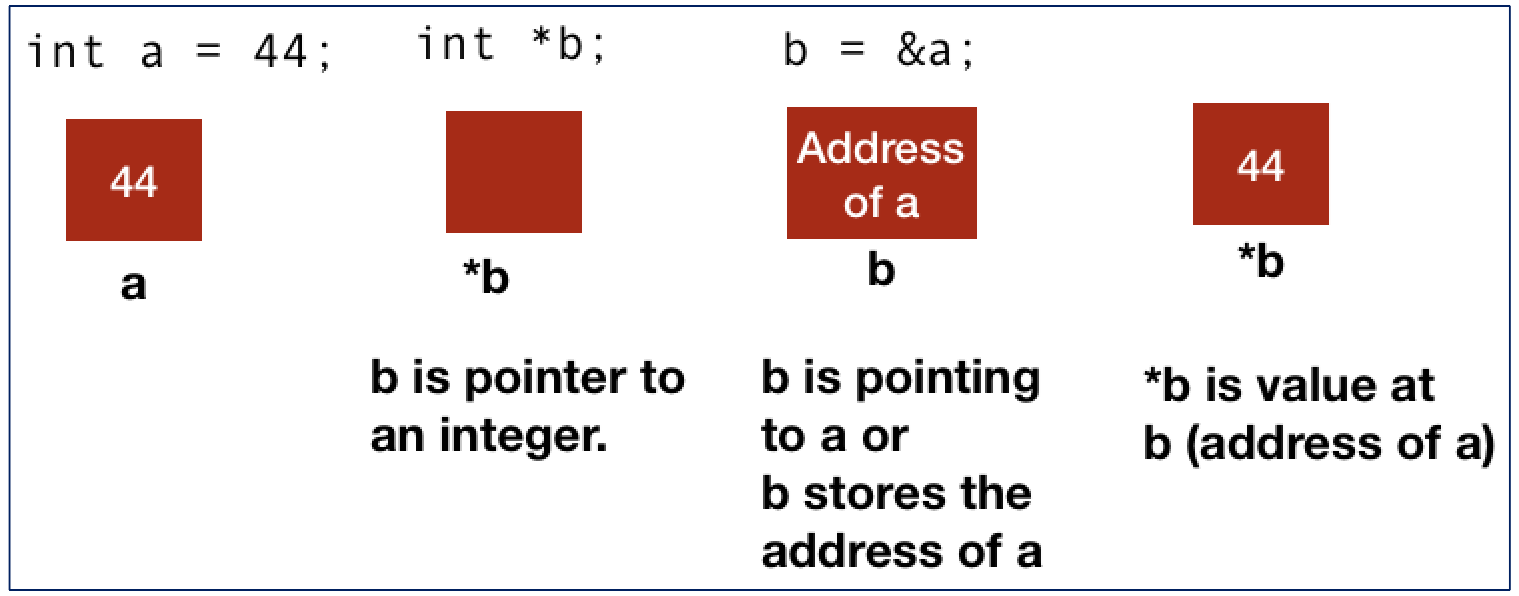
- Hint: int a=5;
- Single storage location is alloted for 5 in a variable “a”.
- How to access memory location?

Pointer

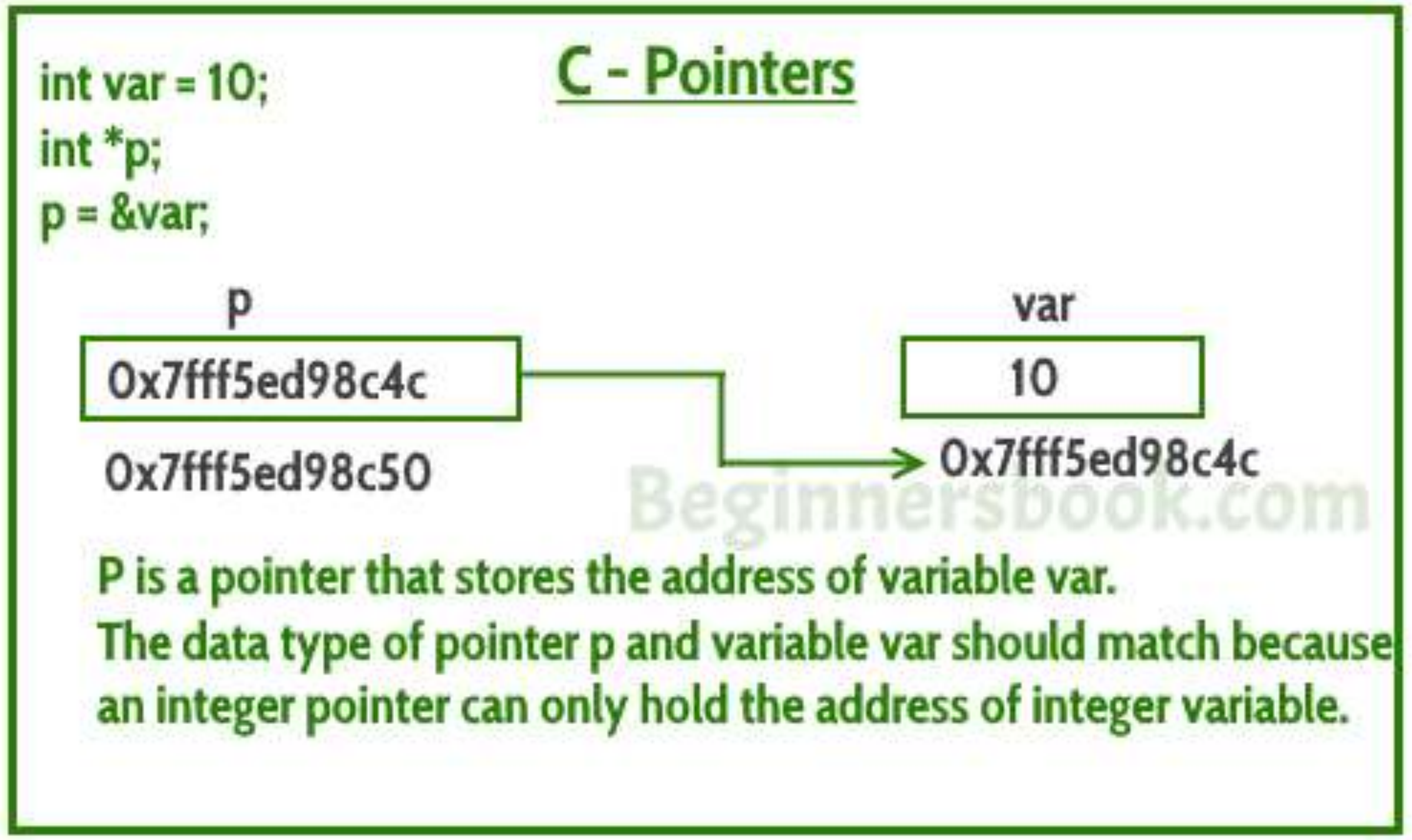
- The pointer in C language is a variable which stores the address of another variable.
- This variable can be of type int, char, array, function, or any other pointer.
- The size of the pointer depends on the architecture.
- **However, in 32-bit architecture the size of a pointer is 2 byte.**

Example

- int *a;//pointer to int
- char *c;//pointer to char



int a = 44;
int *b;
b = &a;
44
a
*b
Address of a
b
44
*b
b is pointer to an integer.
b is pointing to a or b stores the address of a
*b is value at b (address of a)



P is a pointer that stores the address of variable var.
The data type of pointer p and variable var should match because an integer pointer can only hold the address of integer variable.

Pointer Operator

Operator	Operator Name	Purpose
*	Value at Operator	Gives Value stored at Particular address
&	Address Operator	Gives Address of Variable

Example program

```
#include<stdio.h>
int main()
{
int number=50;
int *p;
p=&number; // or int *p=&number
printf("Address of p variable is %x \n",p);
printf("Value of p variable is %d \n",*p);
return 0;
}
```

OUTPUT:

```
Address of p variable is fff4
Value of p variable is 50
```

Address Of (&) Operator

- The address of operator '&' returns the address of a variable.
- But, we need to use %u to display the address of a variable.

Example…

```
#include<stdio.h>
int main(){
int number=50;
printf("value of number is %d, address
of number is %u",number,&number);
return 0;
}
```

Output

```
value of number is 50,
address of number is fff4
```

NULL Pointer

- A pointer that is not assigned any value but NULL is known as the NULL pointer.
- If you don't have any address to be specified in the pointer at the time of declaration, you can assign NULL value.

- **int *p=NULL;**

Pointer Arithmetic

- Following arithmetic operations are possible on the pointer in C language:
- Increment
- Decrement
- Addition
- Subtraction
- Comparison

Incrementing Pointer in C

- If we increment a pointer by 1, the pointer will start pointing to the immediate next location.
- This is somewhat different from the general arithmetic since the value of the pointer will get increased by the size of the data type to which the pointer is pointing.
- The Rule to increment the pointer is given below:
- **new_address= current_address + i * size_of(data type)**

Conti…

Where i is the number by which the pointer get increased.

32-bit:

For 32-bit int variable, it will be incremented by 2 bytes.

64-bit:

For 64-bit int variable, it will be incremented by 4 bytes.

Let's see the example of incrementing pointer variable on 64-bit architecture.

```
#include<stdio.h>
int main(){
int number=50;
int *p;//pointer to int
p=&number;//stores the address of number variable
printf("Address of p variable is %u \n",p);
p=p+1;
printf("After increment: Address of p variable is %u \n",p); // in our case,
p will get incremented by 4 bytes.
return 0;
}
```

Output

- Address of p variable is 3214864300
- After increment: Address of p variable is 3214864304

- **This is similar for Decrementing Pointer**
- Address of p variable is 3214864300
- After Decrement: Address of p variable is 3214864296

Traversing an array by using pointer

```
#include<stdio.h>
void main ()
{
    int arr[5] = {1, 2, 3, 4, 5};
    int *p = arr;
    int i;
    printf("printing array elements...\n");
    for(i = 0; i< 5; i++)
    {
        printf("%d  ",*(p+i));
    }
}
```

OUTPUT:

```
printing array elements...
   1 2 3 4 5
```

C Pointer Addition

- We can add a value to the pointer variable. The formula of adding value to pointer is given below:
- **new_address= current_address + (number * size_of(data type))**

  - **32-bit**
  - For 32-bit int variable, it will add 2 * number.
  - **64-bit**
  - For 64-bit int variable, it will add 4 * number.

Let's see the example of adding value to pointer variable on 64-bit architecture.

```
#include<stdio.h>
int main(){
int number=50;
int *p;                    //pointer to int
p=&number;      //stores the address of number variable
printf("Address of p variable is %u \n",p);
p=p+3;                          //adding 3 to pointer variable
printf("After adding 3: Address of p variable is %u \n",p);
return 0;
}
```

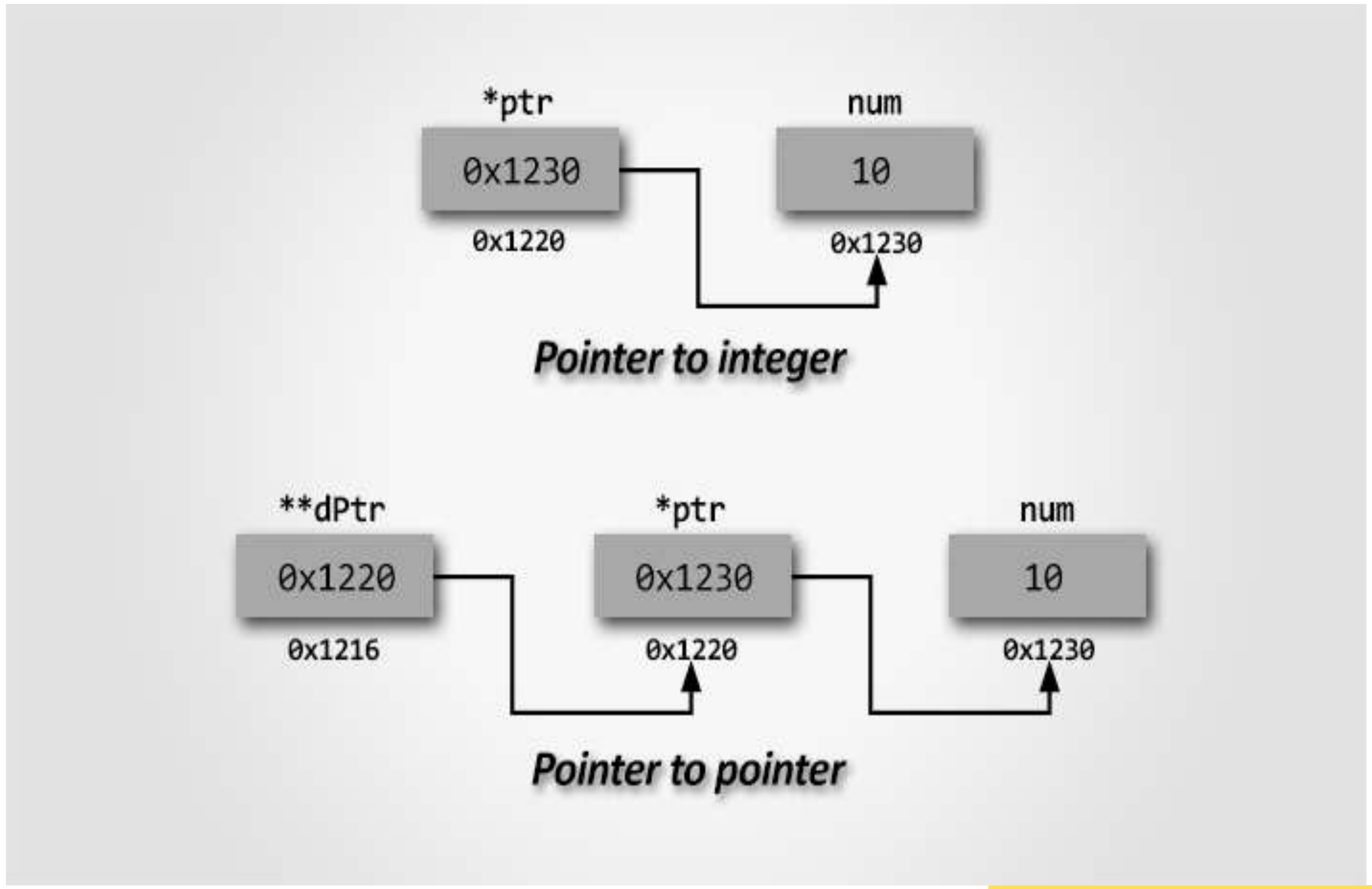
Output

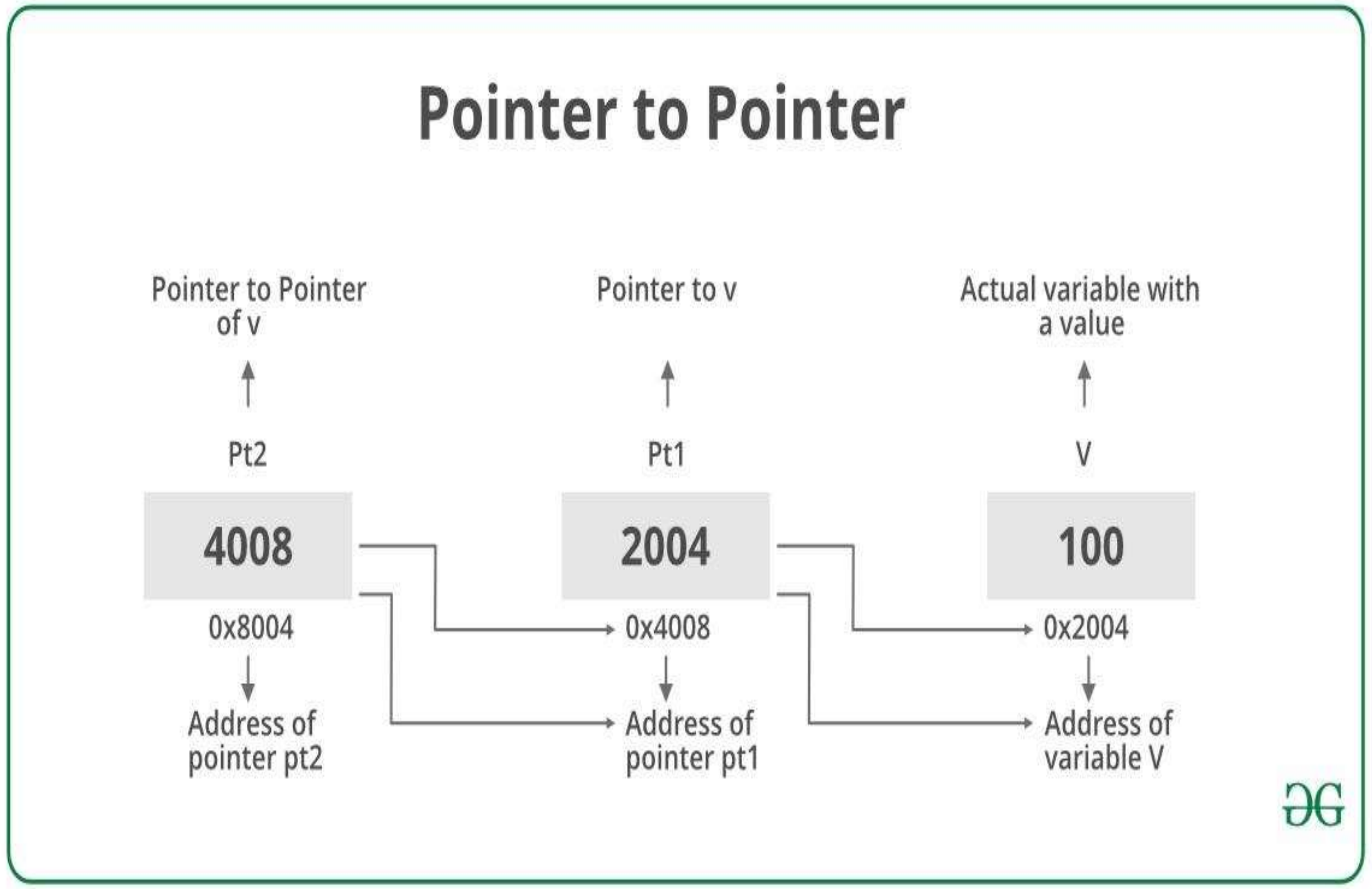
- Address of p variable is 3214864300
- After adding 3: Address of p variable is 3214864312

- This is similar for Pointer Subtraction
- Address of p variable is 3214864300
- After subtracting 3: Address of p variable is 3214864288

Pointer to Pointer / Double Pointer

- A pointer to a pointer is a form of multiple indirection, or a chain of pointers.
- Normally, a pointer contains the address of a variable.
- When we define a pointer to a pointer, the first pointer contains the address of the second pointer, which points to the location that contains the actual value as shown below.





Pointers and arrays

Example 1: Pointers and Arrays

```c
#include <stdio.h>
int main() {
  int i, x[6], sum = 0;
  printf("Enter 6 numbers: ");
  for(i = 0; i < 6; ++i) {
  // Equivalent to scanf("%d", &x[i]);
      scanf("%d", x+i);

  // Equivalent to sum += x[i]
      sum += *(x+i);
  }
  printf("Sum = %d", sum);
  return 0;
}
```

```
Enter 6 numbers:  2
 3
 4
 4
 12
 4
Sum = 29
```

References

1. Reema Thareja, “Programming in C”, Oxford University Press, Second Edition, 2016

Thank You