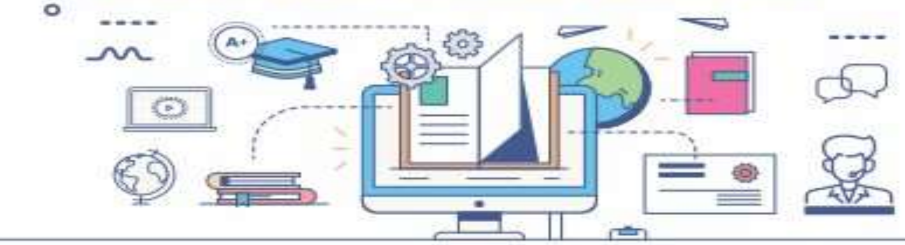
Pointer



CHAPTER-1

Pointer

Pointer

A **pointer** is a variable that stores the **memory address** of another variable. Instead of holding a direct value, it holds the address where the value is stored in memory. There are **2 important operators** that we will use in pointers concepts i.e.
Dereferencing operator(*) used to declare pointer variable and access the value stored in the address.
Address operator(&) used to returns the address of a variable or to access the address of a variable to a pointer.

Steps for accessing the value of a variable using pointer in C

- Declare a normal variable, assign the value
- Declare a pointer variable with the same type as the normal variable
- Initialize the pointer variable with the address of normal variable
- Access the value of the variable by using asterisk (*****) - it is known as **dereference operator**

Declaration & Initialization of Pointer

Syntax

```
datatype *ptr_variablename;
```

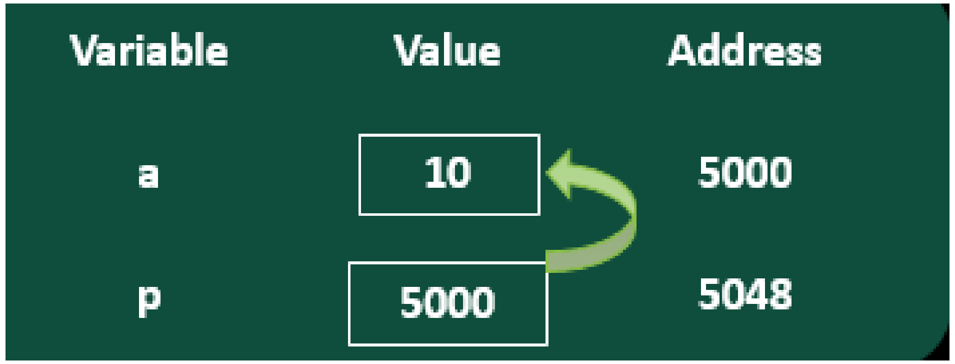
Example

```
void main()
{
    int a=10, *p; // assign memory address of a
    to pointer variable p
    p = &a;
    printf("%d %d %d", a, *p, p);
}
```

Output

```
10 10 5000
```

- ❑ p is integer pointer variable
- ❑ & is address of or referencing operator which returns memory address of variable.
- ❑ * is indirection or dereferencing operator which returns value stored at that memory address.
- ❑ & operator is the inverse of * operator
- ❑ x = a is same as x = *(&a)



Variable	Value	Address
a	10	5000
p	5000	5048

Why use Pointer?

- ❑ C uses pointers to create dynamic data structures, data structures built up from blocks of memory allocated from the heap at run-time. Example linked list, tree, etc.
- ❑ C uses pointers to handle variable parameters passed to functions.
- ❑ Pointers in C provide an alternative way to access information stored in arrays.
- ❑ Pointer use in system level programming where memory addresses are useful. For example shared memory used by multiple threads.
- ❑ Pointers are used for file handling.

This is the reason why C is versatile.

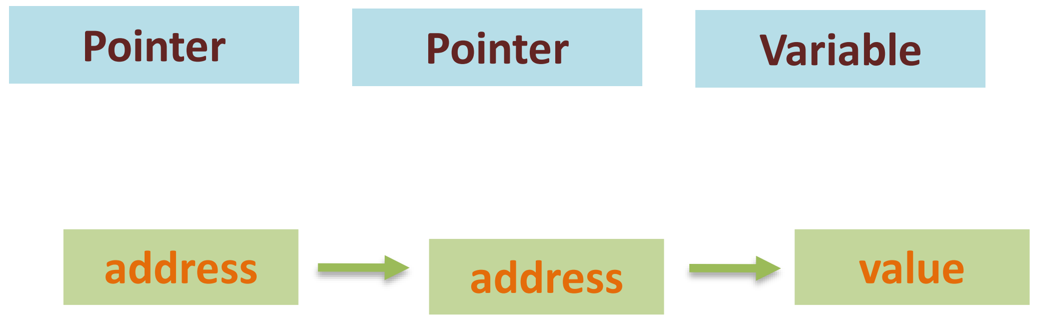
Pointer to Pointer – Double Pointer

- ❑ Pointer holds the address of another variable of same type.
- ❑ When a pointer holds the address of another pointer then such type of pointer is known as pointer-to-pointer or double pointer.
- ❑ The first pointer contains the address of the second pointer, which points to the location that contains the actual value.

Syntax

```
datatype **ptr_variablename;
```

```
int **ptr;
```



Write a program to print variable, address of pointer variable and pointer to pointer variable.

```c
#include <stdio.h>
int main () {
    int var;
    int *ptr;
    int **pptr;
    var = 3000;
    ptr = &var; // address of var
    pptr = &ptr; // address of ptr using address of operator &
    printf("Value of var = %d\n", var );
    printf("Value available at *ptr = %d\n", *ptr );
        printf("Value available at **pptr = %d\n", **pptr);
        return 0;
}
```

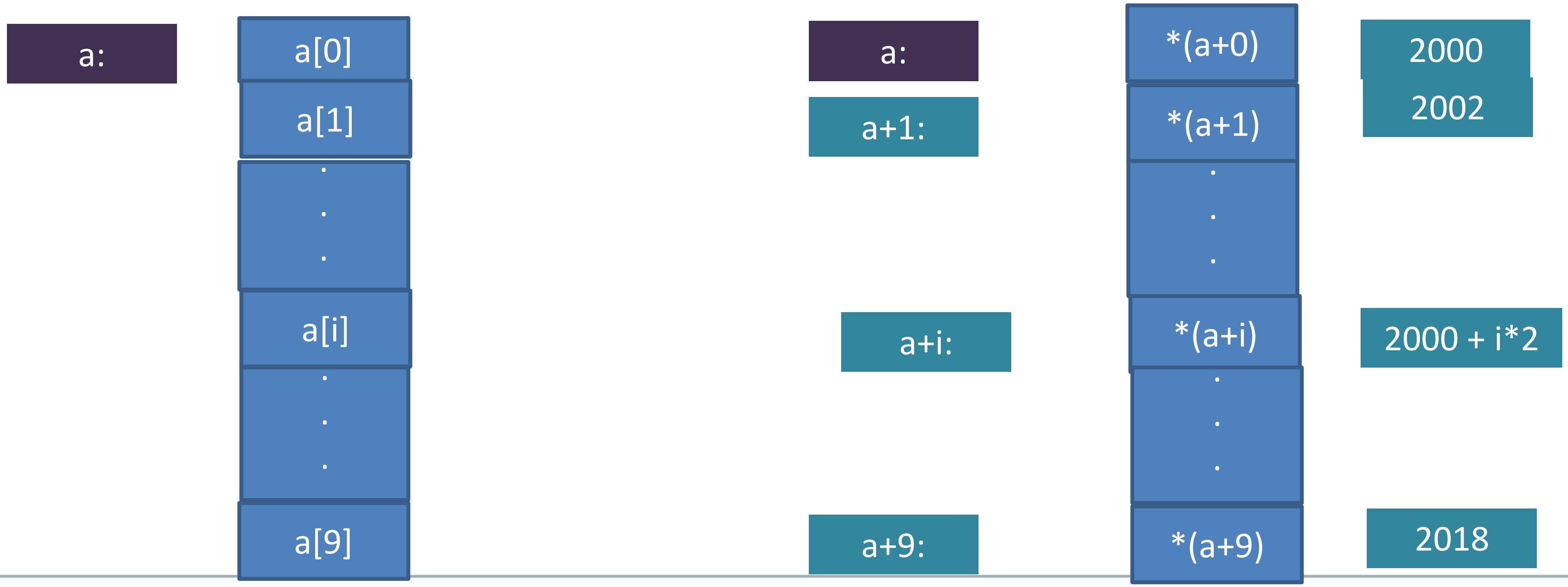
Output

```
Value of var = 3000
Value available at *ptr = 3000
Value available at **pptr = 3000
```

Relation between Array & Pointer – Cont.

Example: `int a[10], *p;`

a[0] is same as *(a+0), a[2] is same as *(a+2) and a[i] is same as *(a+i)



Parul® University

DIGITAL LEARNING CONTENT

Array of Pointer

- As we have an array of char, int, float etc, same way we can have an array of pointer.
- Individual elements of an array will store the address values.
- So, an array is a collection of values of similar type. It can also be a collection of references of similar type known by single name

Syntax

```
datatype *name[size];
```

```
int *ptr[5]; //declares an array of integer pointer of size 5
```

Relation between Array & Pointer

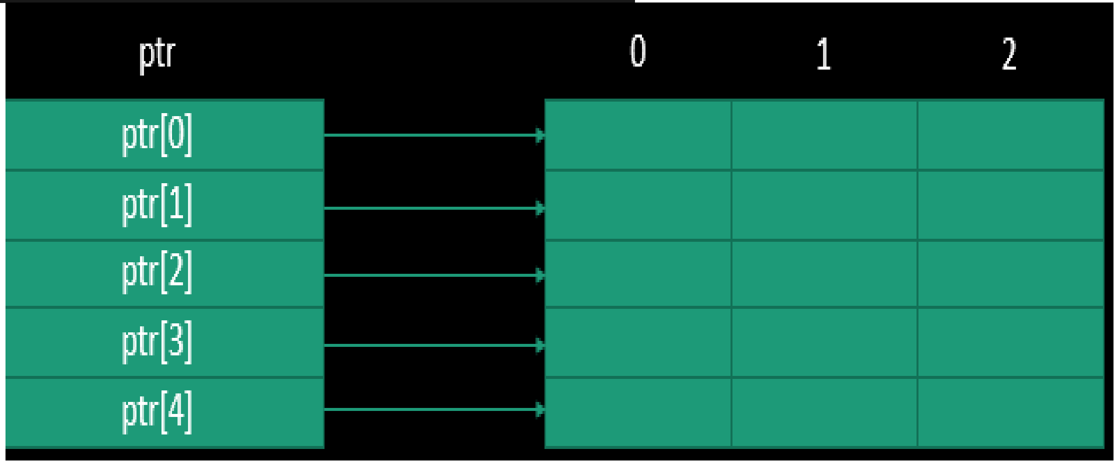
- When we declare an array, compiler allocates continuous blocks of memory so that all the elements of an array can be stored in that memory.
- The address of first allocated byte or the address of first element is assigned to an array name.
- Thus array name works as pointer variable.
- The address of first element is also known as base address.

Array of Pointer – Cont.

An array of pointers ptr can be used to point to different rows of matrix as follow:

```
for(i=0; i<5; i++)
{
        ptr[i]=&mat[i][0];
}
```

By dynamic memory allocation, we do not require to declare two-dimensional array, it can be created dynamically using array of pointers



Call By Value in C

In call by value method of parameter passing, the values of actual parameters are copied to the function’s formal parameters.

There are two copies of parameters stored in different memory locations.

One is the original copy and the other is the function copy.

Any changes made inside functions are not reflected in the actual parameters of the caller.

```
#include <stdio.h>
// Swap functions that swaps
// two values
void swapx(int x, int y) {
    int t;
    t = x;
    x = y;
    y = t;
    printf("Inside swapx: x = %d y = %d\n", x, y);
}
// Main function
int main() {
    int a = 10, b = 20;

    // Pass by Values
    swapx(a, b);
    printf("Inside main: a = %d b = %d", a, b);
    return 0;
}
```

```
Inside swapx: x = 20 y = 10 Inside main: a =
10 b = 20
```

Call by Reference in C

In call by reference method of parameter passing, the address of the actual parameters is passed to the function as the formal parameters. In C, we use pointers to achieve call-by-reference.

Both the actual and formal parameters refer to the same locations.

Any changes made inside the function are actually reflected in the actual parameters of the caller.

```
// Function to swap two variables
// by references
void swapx(int* x, int* y) {
   int t;
   t = *x;
   *x = *y;
   *y = t;
   printf("Inside swapx: x = %d y = %d\n", *x, *y);
}

int main() {
   int a = 10, b = 20;

   // Pass by reference
   swapx(&a, &b);

   printf("Inside main: a = %d b = %d", a, b);
   return 0;
}
```

DIGITAL LEARNING CONTENT

Parul® University

www.paruluniversity.ac.in