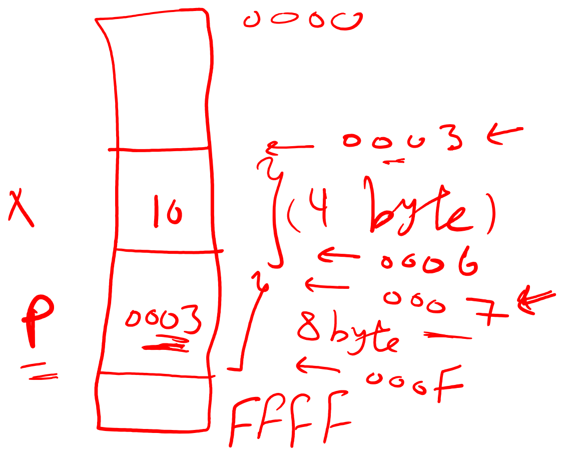
Session 9

Mostafa Akram

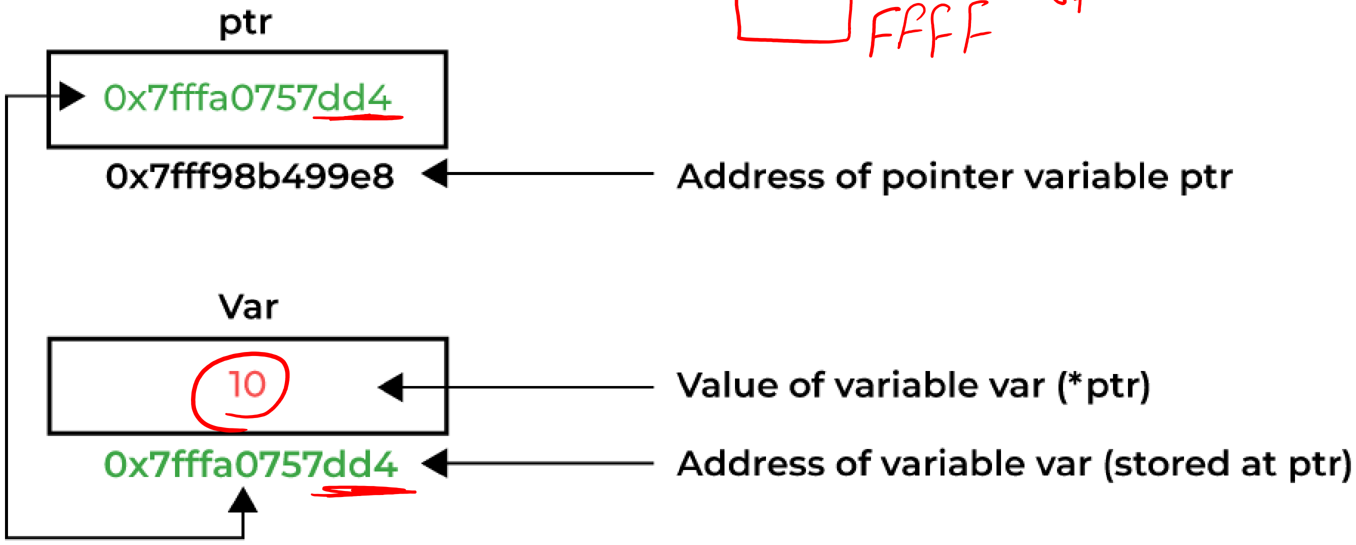
int *p;

int x = 10;



C Pointers

- Pointers are one of the core components of the C programming language. A pointer can be used to store the memory address of other variables, functions, or even other pointers. The use of pointers allows low-level memory access, dynamic memory allocation, and many other functionality in C.



ptr

0x7fffa0757dd4

0x7fff98b499e8 ← Address of pointer variable ptr

Var

10 ← Value of variable var (*ptr)

0x7fffa0757dd4 ← Address of variable var (stored at ptr)

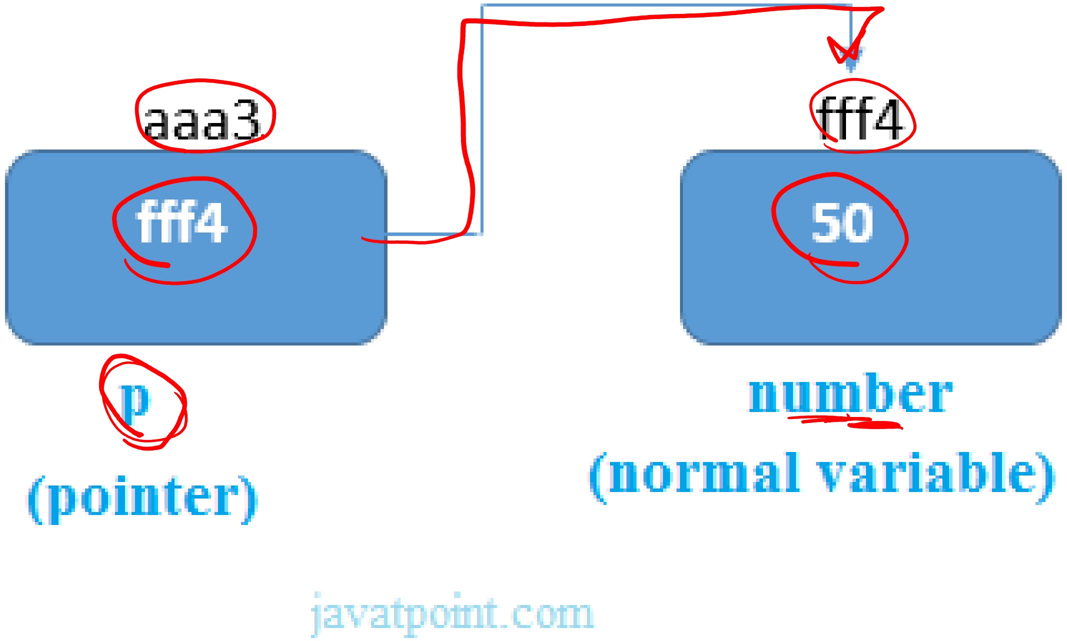
&ptr → 0x7fff ----9e8

ptr → 0x7fff -----dd4

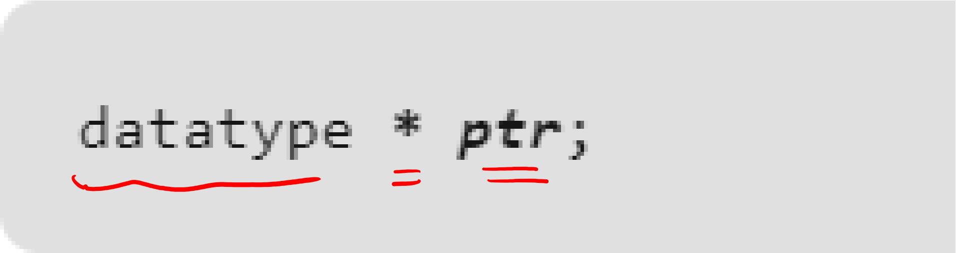
*ptr → 10

What is a Pointer in C?

- *A pointer is defined as a derived data type that can store the address of other C variables or a memory location. We can access and manipulate the data stored in that memory location using pointers.*



Syntax of C Pointers



```
datatype * ptr;
```

- **ptr** is the name of the pointer.
- **datatype** is the type of data it is pointing to.

char

int

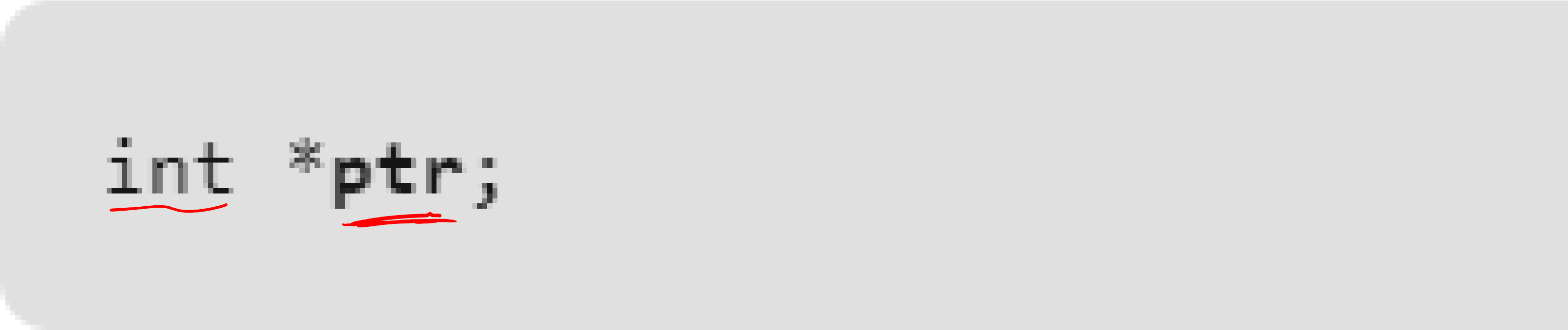
long

:

etc.

How to Use Pointers?

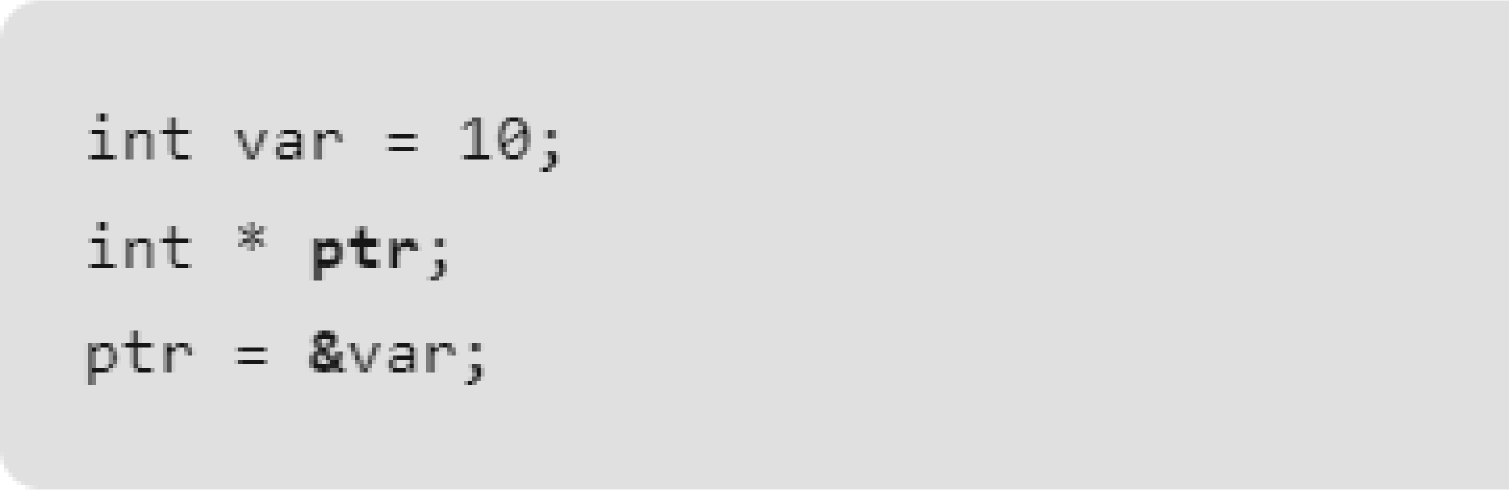
1.Pointer Declaration



```
int *ptr;
```

How to Use Pointers?

2.Pointer Initialization

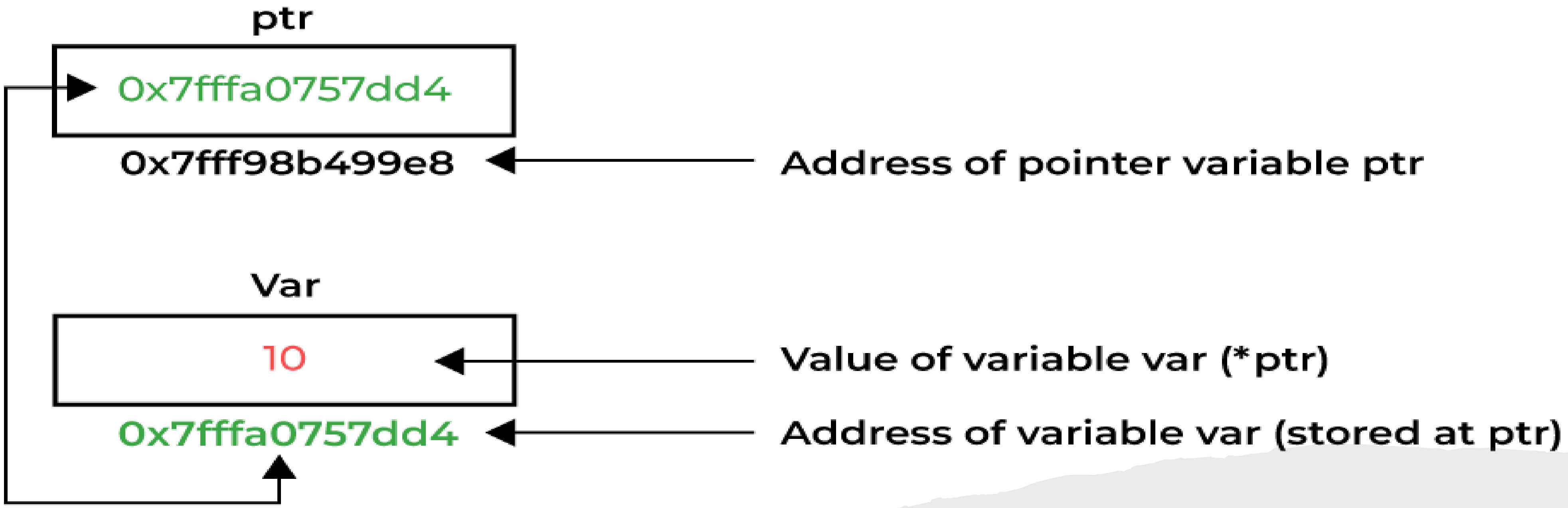


```
int var = 10;
int * ptr;
ptr = &var;
```

How to Use Pointers?

3.Pointer Dereferencing

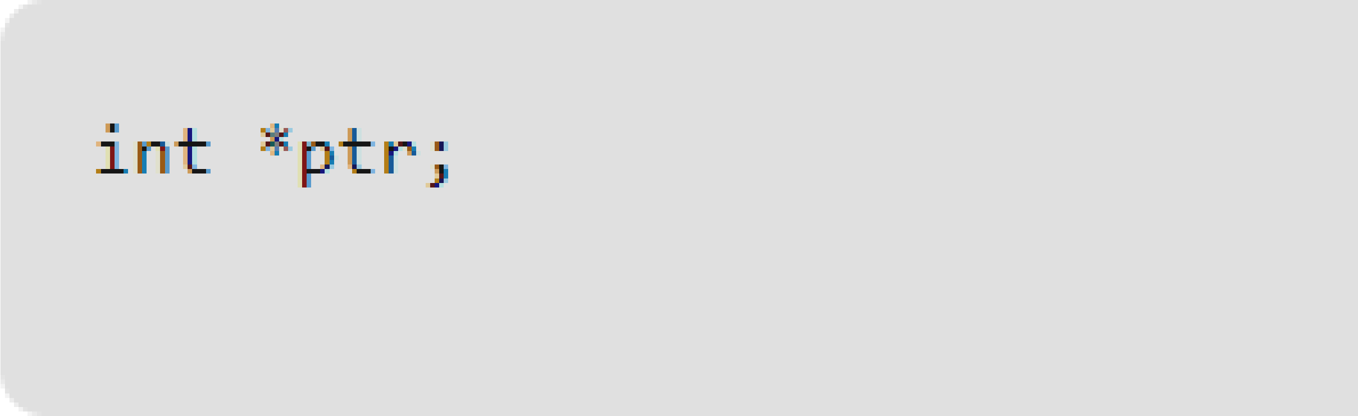
- We use the same **(*) dereferencing operator** that we used in the pointer declaration.



ptr
0x7fffa0757dd4
0x7fff98b499e8
Address of pointer variable ptr
Var
10
Value of variable var (*ptr)
0x7fffa0757dd4
Address of variable var (stored at ptr)

Types of Pointers in C

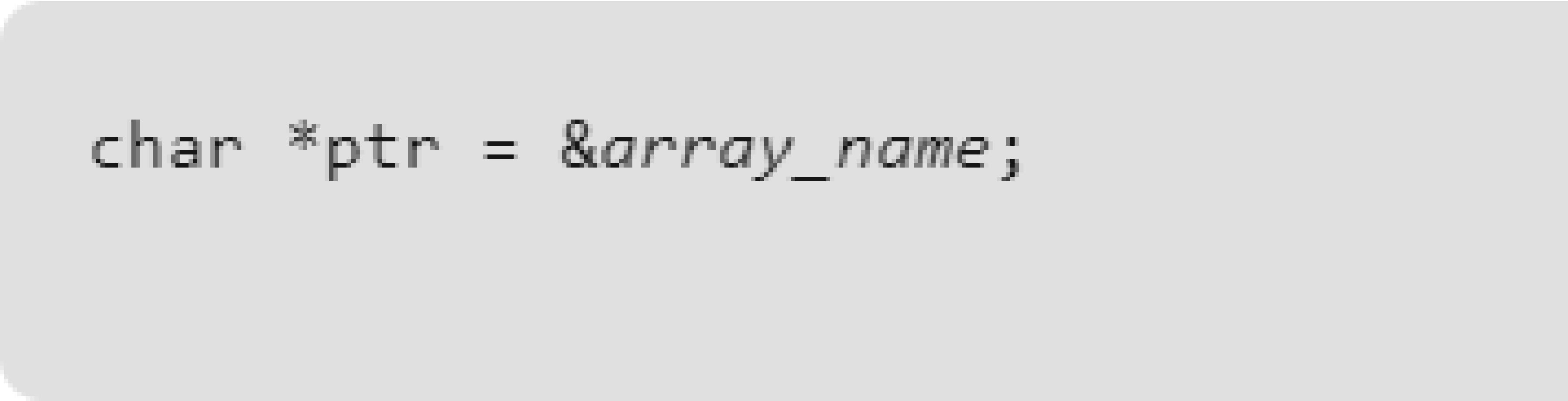
- **1. Integer Pointers**



```
int *ptr;
```

Types of Pointers in C

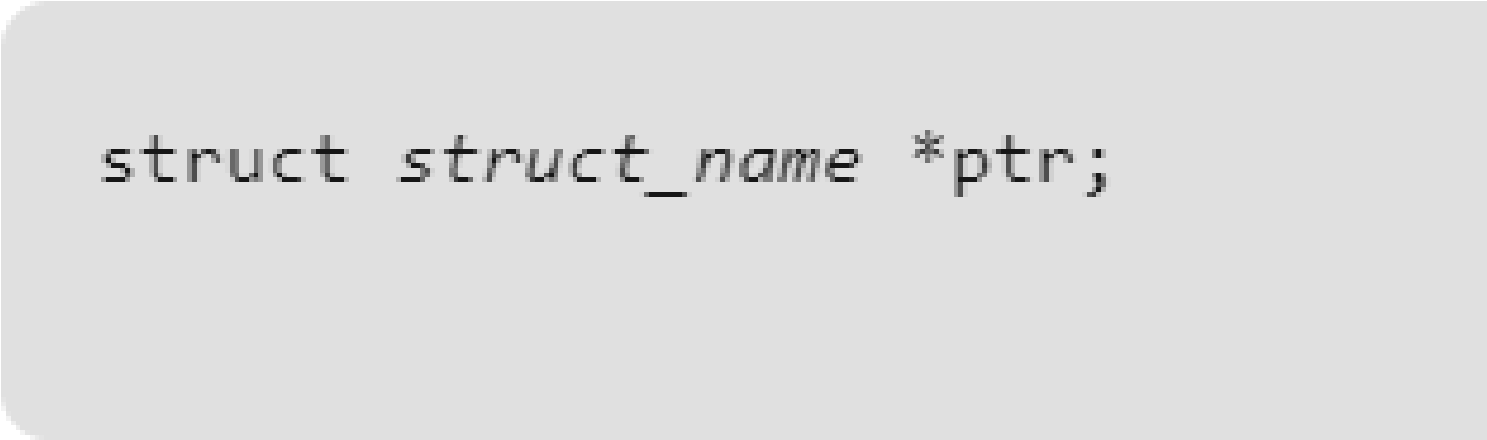
- **2. Array Pointer**



```
char *ptr = &array_name;
```

Types of Pointers in C

- **3. Structure Pointer (Later..)**



```
struct struct_name *ptr;
```

Types of Pointers in C

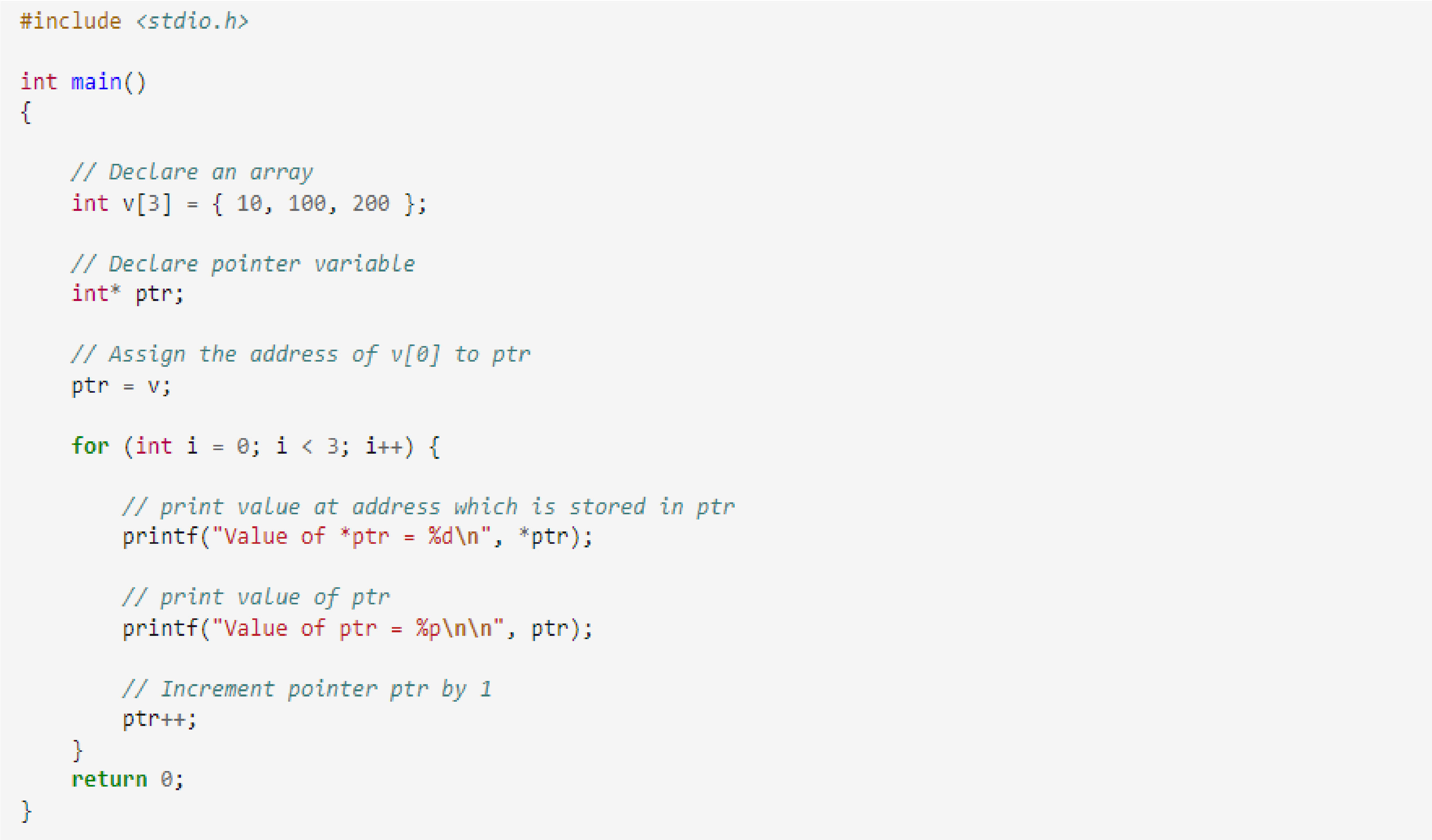
- **Dangling pointer (Later ..)**

```c
int* ptr = (int*)malloc(sizeof(int));

    // After below free call, ptr becomes a dangling pointer
    free(ptr);
    printf("Memory freed\n");

    // removing Dangling Pointer
    ptr = NULL;
```

Size of Pointers in C

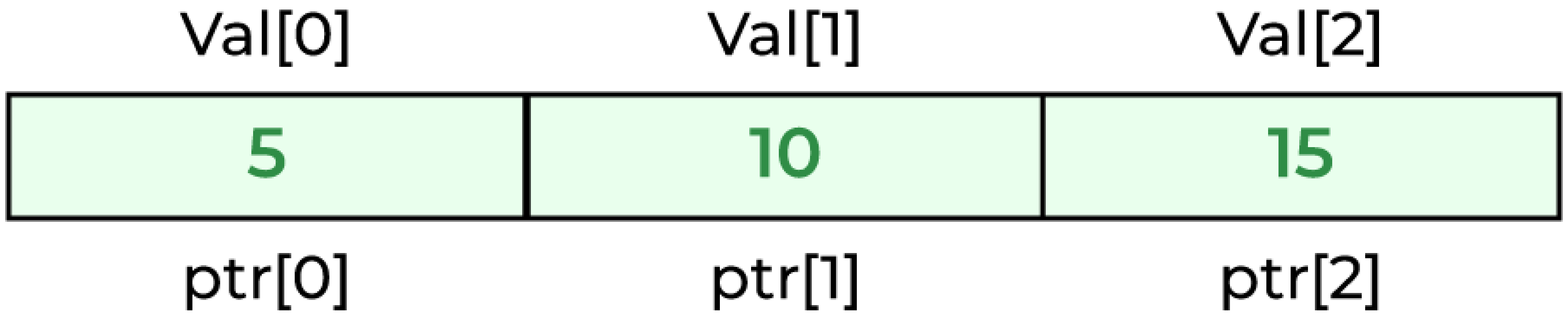
- **8 bytes** for a **64-bit System**
- **4 bytes** for a **32-bit System**

C Pointer Arithmetic



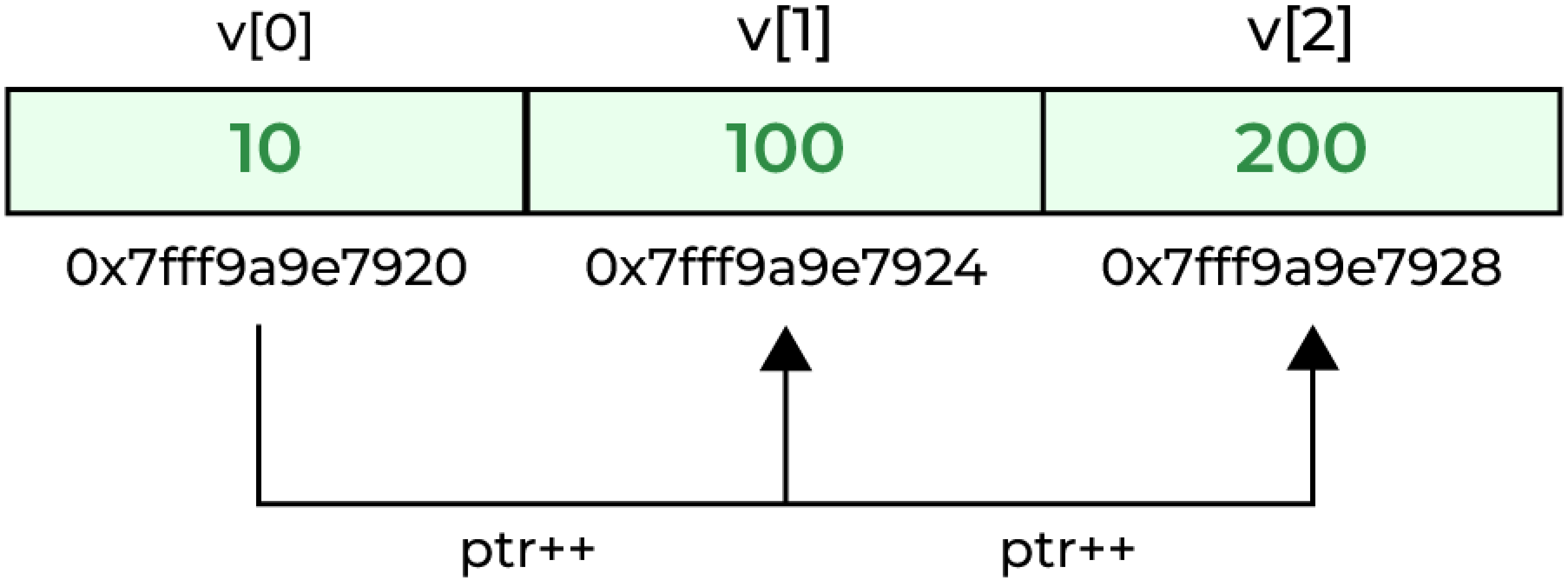
```c
#include <stdio.h>

int main()
{

    // Declare an array
    int v[3] = { 10, 100, 200 };

    // Declare pointer variable
    int* ptr;

    // Assign the address of v[0] to ptr
    ptr = v;

    for (int i = 0; i < 3; i++) {

        // print value at address which is stored in ptr
        printf("Value of *ptr = %d\n", *ptr);

        // print value of ptr
        printf("Value of ptr = %p\n\n", ptr);

        // Increment pointer ptr by 1
        ptr++;
    }
    return 0;
}
```

C Pointers and Arrays



Val[0]	Val[1]	Val[2]
5	10	15
ptr[0]	ptr[1]	ptr[2]

Accessing Array Elements using Pointer Arithmetic



v[0]	v[1]	v[2]
10	100	200
0x7fff9a9e7920	0x7fff9a9e7924	0x7fff9a9e7928

ptr++

ptr++

Uses of Pointers in C

1.Pass Arguments by Reference → by adress

2.Accessing Array Elements

3.Return Multiple Values from Function

4.Dynamic Memory Allocation

5.Implementing Data Structures

6.In System-Level Programming where memory addresses are useful.

7.In locating the exact value at some memory location.

8.To avoid compiler confusion for the same variable name.

9.To use in Control Tables.

Advantages of Pointers

- Pointers are used for dynamic memory allocation and deallocation.
- An Array or a structure can be accessed efficiently with pointers
- Pointers are useful for accessing memory locations.
- Pointers are used to form complex data structures such as linked lists, graphs, trees, etc.
- Pointers reduce the length of the program and its execution time as well.

Disadvantages of Pointers

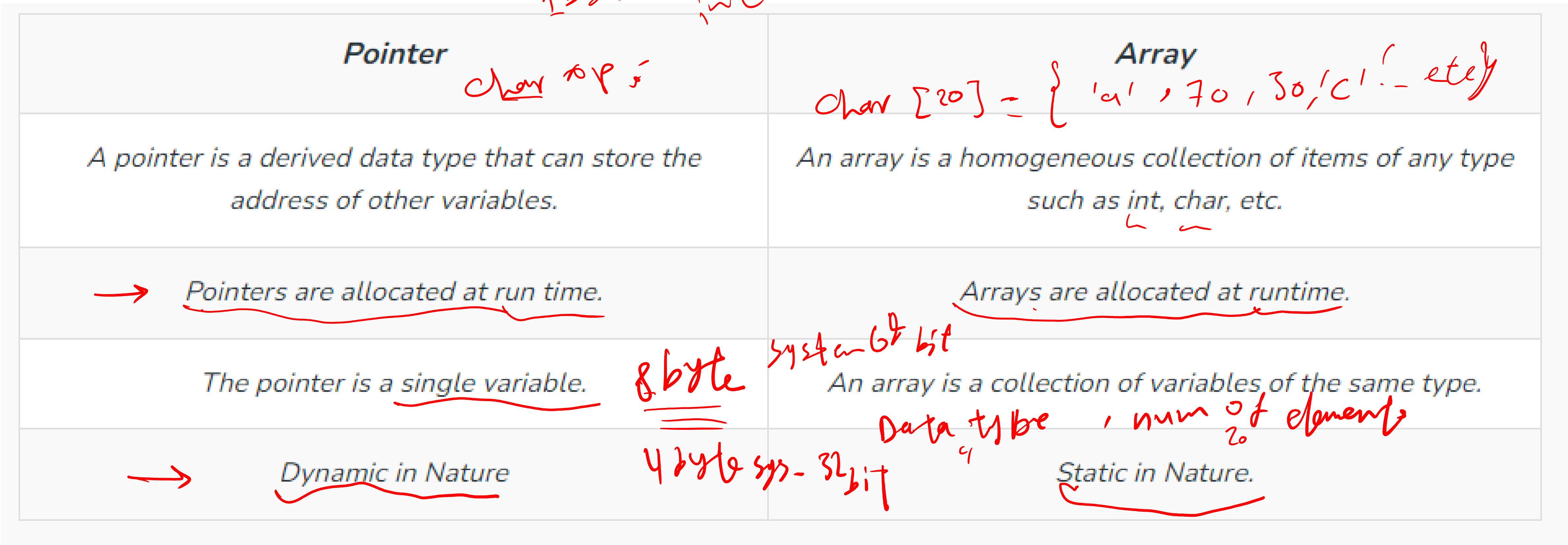
- Memory corruption can occur if an incorrect value is provided to pointers.
- Pointers are a little bit complex to understand.
- Pointers are majorly responsible for memory leaks in C.
- Pointers are comparatively slower than variables in C.

What are the differences between an array and a pointer?

4 byte

int *p

1 byte



Pointer char *p;	Array char [20] = { 'a' , 70 , 30, 'c' ... etc }
A pointer is a derived data type that can store the address of other variables.	An array is a homogeneous collection of items of any type such as int, char, etc.
→ Pointers are allocated at run time.	Arrays are allocated at runtime.
The pointer is a single variable.	An array is a collection of variables of the same type. Data type, num of elements 4 20
→ Dynamic in Nature	Static in Nature.

8 byte system 64 bit

4 byte sys - 32 bit

Task

- Pass array to function of 10 students marks {15,65,56,20,98,100,51,58,50,7}
- Then return array of pass students to main function then print pass student in the main function.

Links