```
C+++pere"|ponteers;
   pointteress"erleijedter"tercenccrelte ! er');
porceccrostie"popteres" (7);
    pointesttes"eesintemiessi | teher"| dettayless |)
     p pointers "effe" | ee "tceiclfee);
  Crnin-= pointtetters"tencade );
         pointers s"[ete]);
         p-- [ert [e()]
    p++| +-estereeesttetis"tgcntertee/));
         pointers "tes pointters"-rrtertretestoffet [])
   p#+1le-sstet_"itotnntres"et"-Lloretn "&e));
         prenters"[ eleee"p] pinterrs];
       poointers "licc" lontes aojul. ernneee.ij
    T++ r-estetressteooo"loldre" (ctetiteretellaff)
           piters "letlod etsmeerris i[lette);
        pce+"jesnn"lentnters*tee (eg)
            rjolictco "jecite");
```





# Pointers and Function Parameters in C++

Lecture 9

Asst. Lect. Ali Al-khawaja

## What is a pointer?

- A pointer is a variable that stores the memory address of another variable. It essentially "points to" the location in memory where data is stored rather than storing the actual data value itself.
- Pointers are powerful features in C++ that enable direct memory manipulation, dynamic memory allocation, and efficient parameter passing to functions.
- Working with pointers involves two key operators:
  - The address operator (&) returns the memory address of a variable
  - The dereference operator (\*) accesses the value stored at the memory address
- Understanding pointers is essential for advanced C++ programming and memory management.

#### pointer-swareseemet)) vilitumar) valter=≡ramanel); vairetters≡≡( ■ poirmiler=" II C+> y oillsvel "amaresesteel); poiltter \* merbreeme); portetuessner==); vihtemen) porteteor="=" =" porter==eamfineoel); porter==mnahtee\*= . porter==panmtee\*meteestes)) porter==ramhessel}; poiretcellee == )) portatrer=II=1 porter==bowlbesel}; porterer = tet -

## Syntax:

#### Pointer Declaration

```
int a = 10;
int* ptr = &a; //
'ptr' stores the
address of 'a'
```

#### Address Operator

&a  $\rightarrow$  gives the address of variable a

#### Dereference Operator

\*ptr  $\rightarrow$  accesses the value stored at the address ptr is pointing to (dereferencing)

## Example:

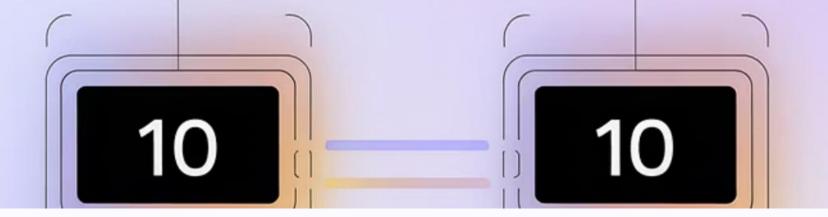
```
#include <iostream>
using namespace std;
int main() {
    int x = 5;
    int* p = &x;
    cout << "Value of x: " << x << endl;</pre>
    cout << "Address of x: " << &x <<
endl;
    cout << "Pointer p stores address: "</pre>
<< p << endl;
    cout << "Value pointed by p: " << *p</pre>
<< endl;
    return 0;
```



## **Key Concepts:**

- \* Dereferencing
  - \* is used for dereferencing
- & Address Operator

& is used to get the address



### Why use pointers in functions?

Modify Original Variables

To allow the function to modify the original variable (pass by reference)

Efficiency

....

To avoid copying large data structures

#### A. Passing by Value (Default)

```
void change(int a) {
    a = 20;
}
int x = 10;
change(x);
cout << x; // x remains 10</pre>
```

# P Pass—by-pointer C++ Pass-by-pointer

# B. Passing by Pointer (Reference using pointer)

Define Function with Pointer Parameter

```
void change(int* a) {
    *a = 20;
}
```

Call Function with Address

```
int x = 10;
change(&x);
```

Check Result

```
cout << x; // x becomes 20
```

The function modifies the original variable because it receives its address.

## Full Example:

```
#include <iostream>
using namespace std;
void update(int* num) {
    *num += 5;
int main() {
    int a = 15;
    update(&a);
    cout << "Updated value: " << a << endl; // Output: 20</pre>
    return 0;
```

#### Homework

