DATA STRUCTURES TO BE READ

**#include <iostream>**: This line includes the iostream header, which provides functionality for input and output operations. It allows you to use **cout** for output and **cin** for input.

**using namespace std;**: This line declares that you want to use the **std** namespace in your code. The **std** namespace contains many standard C++ elements, including those related to input and output. This line is optional, but it makes your code more concise by allowing you to use **cout** and **cin** without the **std::** prefix.

Don’t forget return 0;

OOP-It is a way of designing and organizing code to promote modularity, reusability, and flexibility in software development.

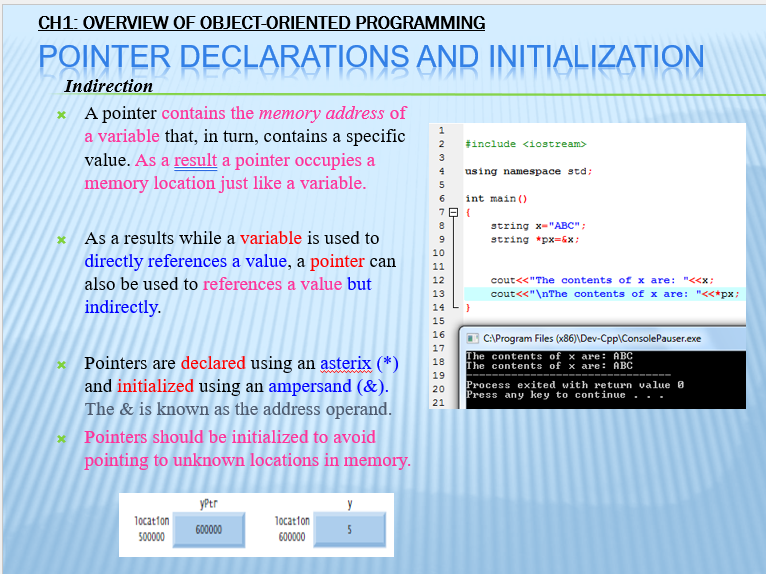
**Objects:** Objects are instances of classes. A function and its associated data is called an object

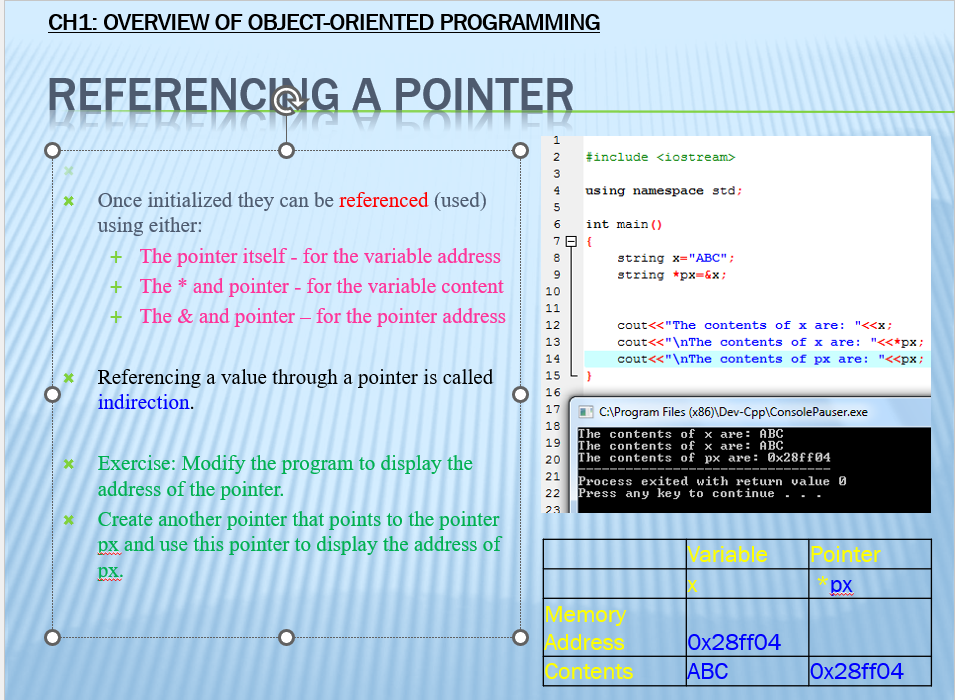
**Inheritance:** Inheritance allows a class (subclass or derived class) to inherit the properties and behaviors of another class (superclass or base class). It promotes code reuse

Notice the keywords public and private in the employee class. These keywords are called access modifiers

For example empName preceded by private can only be accessed by functions in class employee

all the functions in in class employee are public, so they can be accessed by functions in other classes or main().





A screenshot of a computer program

Description automatically generated

* **A data structure** is an arrangement of data in a computer’s memory (or sometimes on a disk). Data structures include arrays, linked lists, stacks, binary trees, and hash tables, among others.
* **Algorithms** manipulate the data in these structures in various ways, such as inserting a new data item, searching for a particular item, or sorting the items.

A computer screen shot of a computer

Description automatically generated

**Recursion** - a function calling itself

BINARY SEARCH  
#include <iostream>

using namespace std;

int brokenBinarySearch(int arr[], int left, int right, int target) {

while (true) {

// Intentionally break the condition

if (left > right) {

// Return a special value indicating an error

return -9999;

}

int mid = left + (right - left) / 2;

// Check if the target is present at the middle

if (arr[mid] == target) {

return mid;

}

// If the target is greater, ignore the left half

if (arr[mid] < target) {

left = mid + 1;

}

// If the target is smaller, ignore the right half

else {

right = mid - 1;

}

}

}

int main() {

int arr[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

int n = sizeof(arr) / sizeof(arr[0]);

int target;

cout << "Enter the number to search: ";

cin >> target;

int result = brokenBinarySearch(arr, 0, n - 1, target);

if (result != -9999) {

cout << "Element found at index " << result << endl;

} else {

cout << "Unexpected error: Left became greater than right" << endl;

}

return 0;

}

DELETE A NUMBER  
#include <iostream>

using namespace std;

int scores[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; // Assuming your array size is 10

void delete1() {

int index = -1;

int number;

cout << "Choose a number to be deleted: ";

cin >> number;

for (int i = 0; i < 10; i++) {

if (scores[i] == number) {

index = i;

break;

}

}

if (index != -1) {

for (int j = index; j < 9; j++) {

scores[j] = scores[j + 1];

}

cout << "Number deleted" << endl;

} else {

cout << "Number not found" << endl;

}

cout << "Array after deletion: ";

for (int i = 0; i < 9; i++) {

cout << scores[i] << " ";

}

}

int main() {

delete1();

return 0;

}