**C++ 101 – Session 8 Notes**

**Topics: References, Memory Addresses, Pointers, Dereferencing, and Modifying Pointers**

**1. References in C++**

A **reference** in C++ is like a **nickname** or **alias** for another variable. Once a reference is bound to a variable, it cannot be changed to refer to another variable.

**📌 Syntax:**

type &refName = variable;

**✅ Example:**

string food = "Pizza";

string &meal = food; // meal is a reference to food

Now, both meal and food refer to the **same memory location**.  
Any change to meal is reflected in food, and vice versa.

🧠 **Use Cases:**

* Passing variables to functions by reference (to avoid copying)
* Updating the original variable from inside a function

**2. Memory Addresses**

Every variable in your program is stored in memory, and each has a **unique address**.

You can get the memory address of a variable using the **address-of operator (&)**.

**✅ Example:**

string food = "Pizza";

cout << &food; // This prints the memory address of food

📌 This is useful when working with pointers, debugging, or optimizing memory usage.

**3. Pointers**

A **pointer** is a variable that stores the **memory address of another variable**.

**📌 Syntax:**

type \*pointerName = &variable;

* \*pointerName → dereferencing: gets the value stored at the pointer's address
* &variable → gets the address of the variable

**4. Code Breakdown: Class Example**

#include <iostream>

using namespace std;

int main(){

string food = "Pizza"; // A regular string variable

string &meal = food; // Reference to food

string \*ptr = &food; // Pointer to food (stores address)

cout << "Original food: " << food << endl;

cout << "Meal reference: " << meal << endl;

cout << "Food pointer: " << ptr << endl;

\*ptr = "Humburger"; // Dereferencing and modifying value

return 0;

}

**🔍 What's happening?**

| **Line** | **Explanation** |
| --- | --- |
| string food = "Pizza"; | A normal string variable |
| string &meal = food; | meal is another name for food. Changing one changes both |
| string \*ptr = &food; | ptr is a pointer that stores the address of food |
| cout << ptr; | Prints the memory address stored in ptr |
| \*ptr = "Humburger"; | Dereferences the pointer → accesses the value and updates it |

**🧾 Output:**

Original food: Pizza

Meal reference: Pizza

Food pointer: 0x61feec (some memory address)

After \*ptr = "Humburger";, if we printed food or meal, the output would be:  
"Humburger" — because all three (food, meal, \*ptr) point to the same data in memory.

**5. Dereferencing a Pointer**

To **dereference** a pointer means to access the value stored at the memory location it points to. This is done using the \* operator.

string food = "Pizza";

string \*ptr = &food;

cout << \*ptr; // Outputs: Pizza

**6. Modifying Values Using Pointers**

You can also change the **original value** through a pointer:

\*ptr = "Burger";

cout << food; // Outputs: Burger

📌 The pointer goes to the memory address of food and updates its value.

**🧠 Summary**

| **Concept** | **Meaning** |
| --- | --- |
| & | Gets the memory address (address-of operator) |
| \* | Accesses or modifies the value at the address (dereference operator) |
| Reference | An alias to an existing variable |
| Pointer | A variable that stores a memory address |

**🛠️ Student Task**

Write a program that:

1. Declares a string variable
2. Creates a reference to it
3. Creates a pointer to it
4. Outputs:
   * The original value
   * The reference
   * The pointer (address)
   * The dereferenced pointer
5. Modifies the value using the pointer
6. Prints the updated value using the reference