CSCE 1101

LAB 7

COVERED IN THIS WEEK'S LAB:

- VTables
- Casting
- Linked Lists

WHAT IS A VTABLE?

A **vtable** (short for **virtual table**) is an internal mechanism used by C++ compilers to implement **runtime polymorphism** via **virtual functions**.

*When you declare a **virtual function** in a class, the compiler automatically:*

1. Creates a **vtable** for the class.
2. Inserts a hidden **vptr (virtual pointer)** inside each object of that class.
3. At runtime, uses the **vptr** to call the correct function via the vtable.

EXAMPLE

Imagine you have a class like this:

```
class Animal {
public:
    virtual void speak() {
        cout << "Animal speaks\n";
    }
    virtual void move() {
        cout << "Animal moves\n";
    }
};
```

The compiler creates a **vtable** behind the scenes that might look like this:

Animal vtable:

Function Name	Function Address
speak()	&Animal::speak
move()	&Animal::speak

EXAMPLE

Now if you have a derived class:

```
class Dog : public Animal {
public:
    void speak() override {
        cout << "Dog barks\n";
    }
};
```

Then Dog will have its own vtable:

Dog vtable:

Function Name	Function Address
speak() move()	&Dog::speak &Animal::move

HOW THE PROGRAM USES THE VTABLE

```
Animal* a = new Dog();
a->speak();  // What happens here?
```

This happens under the hood:

1. The Dog object has a hidden pointer inside (called **vptr**) pointing to **Dog's vtable**.
2. The code a->speak():
    1. Looks up the speak() entry in the vtable.
    2. Finds &Dog::speak.
    3. Calls Dog::speak().

That's how even though a is of type Animal*, the correct version (Dog::speak) is called.

STATIC CAST

🔷 What Is Static Cast?

static_cast is a compile-time cast used to convert between related types, such as:

- Upcasting or downcasting in inheritance hierarchies.
- Converting between built-in types (e.g., int to float).

⚠ **Static cast does not check types at runtime, so if you downcast incorrectly, the behavior is undefined.**

DYNAMIC CAST

What Is Dynamic Cast?

dynamic_cast is a runtime-safe cast used mainly for:

- **Downcasting** (casting a base class pointer to a derived class pointer).
- It works only when the base class has at least **one virtual function**.
- If the cast fails, it returns nullptr for pointers or throws an exception for references.

EXAMPLE

```
#include <iostream>
using namespace std;

class Device {
public:
    virtual void info() {
            cout << "Generic device\n";}
    };

class Laptop : public Device {
public:
    void openLid() {
           cout << "Opening laptop lid\n";}
    };
```

```
int main() {
    Device* d = new Laptop();  // Upcasting (safe)

    // Static cast (assumes d is a Laptop)
    Laptop* l1 = static_cast<Laptop*>(d);
    l1->openLid();  // ✅ Works here because d actually is a Laptop

    // Dynamic cast (safe check)
    Laptop* l2 = dynamic_cast<Laptop*>(d);
    if (l2)
        l2->openLid();
    else
        cout << "Cast failed\n";

    delete d;
    return 0;
}
```

Note:

If d pointed to something that wasn't a Laptop, like another derived class, static_cast would crash (undefined behavior), but dynamic_cast would safely return nullptr.

EXERCISE 1

- Based on what you learned about **Linked Lists** in the lecture, download the LinkedList.cpp file from Canvas and implement the following:
  - Create a function to add a node
  - Create a function to delete a node

EXERCISE 2

- Let the user enter values into a vector until they enter -1.

Then, **remove all values less than 50** using an iterator, and print the result.

EXERCISE 2 SOLUTION

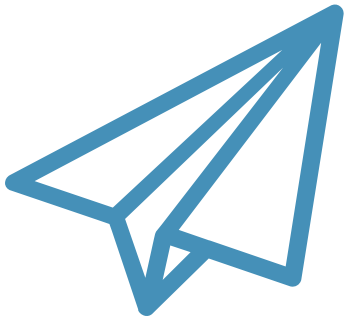
```
#include <iostream>
#include <vector>
using namespace std;

int main() {
    vector<int> scores;
    int input;
    cout << "Enter scores (-1 to stop):\n";
    while (cin >> input && input != -1) {
        scores.push_back(input);
    }

    // Remove scores < 50
    for (auto it = scores.begin(); it != scores.end(); ) {
        if (*it < 50)
            it = scores.erase(it);  // erase returns next valid iterator
        else
            ++it;
    }

    cout << "Scores >= 50: ";
    for (int s : scores)
        cout << s << " ";

    return 0;
}
```



If you have any questions,
please send me an email

lena.shamseldin@aucegypt.edu