VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
HO CHI MINH UNIVERSITY OF SCIENCE
FACULTY OF INFORMATION TECHNOLOGY

OBJECT ORIENTED PROGRAMMING (22_5)

ASSIGNMENT:

GROUP ASSIGNMENT WEEK 02

Advisor(s)::	Hồ Tuấn Thanh	
Student(s)::	Nguyễn Hoàng Anh	22120014
	Trần Hùng Anh	22120016 (Leader)
	Trương Tiến Anh	22120017
	Đoàn Minh Cường	22120043
	Nông Quốc Việt	22120432

Ho Chi Minh City, 12/2023

Mục lục

Phần I
Giới thiệu nhóm và tiến độ công việc

Tên nhóm: **ClownIT**

Thông tin các thành viên nhóm:

STT	Họ Tên	MSSV	Email
1	Nguyễn Hoàng Anh	22120014	hoanganhik172510@gmail.com
2	Trần Hùng Anh	22120016	anhth5659@gmail.com
3	Trương Tiến Anh	22120017	truongtienanh16@gmail.com
4	Đoàn Minh Cường	22120043	dmcuong02072004@gmail.com
5	Nông Quốc Việt	22120432	quocviett1302@gmail.com

Bảng 1: *Thông tin các thành viên trong nhóm*

Công việc được phân chia và tiến độ công việc:

STT	Họ Tên	Công việc được giao	Tiến độ công việc	Vấn đề gặp phải
1	Nguyễn Hoàng Anh	Câu 1: Constructor	Hoàn thành	Không có
2	Trần Hùng Anh	Làm Team Contract	Hoàn thành	Không có
3	Trương Tiến Anh	Câu 1: Constructor	Hoàn thành	Không có
4	Đoàn Minh Cường	Câu 2: Destructor	Hoàn thành	Không có
5	Nông Quốc Việt	Câu 2: Destructor	Hoàn thành	Không có

Bảng 2: *Công việc nhóm*

Phần II
Báo cáo bài tập

1 Constructor

1.1 Sự cần thiết khởi tạo Constructor bên trong đối tượng

Việc khởi tạo đối tượng thông qua các hàm, chẳng hạn như **KhoiTao()** sẽ gặp phải nhiều vấn đề và có thể bị lỗi. Cụ thể người lập trình có thể quên gọi hàm và không khởi tạo đối tượng cần dùng. Trong trường hợp này, các thành phần thuộc tính của lớp đối tượng có thể nhận các giá trị là dữ liệu rác còn sót lại trong bộ nhớ (Hay các giá trị ngẫu nhiên) nào đó mà ta không biết trước.

=> Do đó **Constructo**r ra đời để giải quyết vấn đề trên

1.2 Khái niệm Constructor

Trong lập trình hướng đối tượng (OOP) bằng C++, hàm constructor là một loại hàm đặc biệt được sử dụng để khởi tạo đối tượng khi nó được tạo ra. Mục đích chính của hàm constructor là cấp phát bộ nhớ cho các thành phần của đối tượng và thiết lập các giá trị ban đầu cho chúng.

Đặc điểm của Constructor:

- Phương thức có tên trùng với tên class và không có kiểu trả về.

- Tự động chạy ngay khi đối tượng được tạo, có nhiệm vụ khởi tạo dữ liệu và chuẩn bị cho những công việc cần thiết để bắt đầu chu kỳ sống của đối tượng.
- Có thể có nhiều phương thức tạo lập chồng nhau (overloading). Chúng được phân biệt theo quy tắc như việc chồng các hàm (Phân biệt nhờ danh sách tham số truyền vào)

VDMH: Trong ví dụ trên, Khi ta khởi tạo đối tượng **SinhVien sv** trong hàm **main()**, thì **Constructor SinhVien()** sẽ tự động chạy và dòng chữ "Ham khoi tao duoc goi" sẽ được in ra

```
class SinhVien
{
private:
    string id, name, date;
    double gpa;
public:
    SinhVien();//Constructor
};

//Constructor
SinhVien::SinhVien()
{
    cout << "Ham khoi tao duoc goi" << endl;
}

int main()
{
    SinhVien sv;
}
```

Hình 1: *VDMH*

1.3 Các phương thức tạo lập

1.3.1 Phương thức tạo lập mặc định (default constructor)

Là phương thức tạo lập không có tham số đầu vào. Trong tường hợp người lập trình không định nghĩa bất kì phương thức tạo lập nào thì trình biên dịch tự động tạo ra một phương thức tạo lập mặc định cho lớp này.
VDMH: Trong ví dụ này, lớp **SinhVien** chỉ định một **constructor** mặc định không có tham số, và nó được sử dụng để tạo một đối tượng **SinhVien** trong hàm **main()**. **Lưu ý:** Khi người lập trình khai báo

```
class SinhVien {
private:
    std::string id, name, date;
    double gpa;

public:
    // Default constructor
    SinhVien() {
        id = "";
        name = "";
        date = "";
        gpa = 0.0;
    }

    // Display student information
    void displayInfo() {
        std::cout << "ID: " << id << std::endl;
        std::cout << "Name: " << name << std::endl;
        std::cout << "Date of Birth: " << date << std::endl;
        std::cout << "GPA: " << gpa << std::endl;
    }
};

int main() {
    // Using default constructor
    SinhVien sv1;
    sv1.displayInfo();

    return 0;
}
```

Hình 2: *VDMH*

bất kỳ một phương thức tạo lập nào cho class thì phương thức mặc định sẽ không được phát sinh bởi trình biên dịch cho class này nữa. Trong trường hợp này, nếu muốn sử dụng phương thức tạo lập mặc định, người lập tình phải tự viết thêm vào.

1.3.2 Phương thức tạo lập có tham số đầu vào

Là phương thức tạo lập khác do người dùng định nghĩa với các tham số đầu vào khác nhau để khởi tạo dữ liệu cho đối tượng.
VDMH: Trong ví dụ này, chỉ có một **constructor** duy nhất được định nghĩa trong lớp **SinhVien**, là

constructor với các đối số. Điều này làm cho **constructor** mặc định (không có đối số) không được tự động tạo ra. Ta có thể tạo đối tượng **SinhVien** bằng cách cung cấp các đối số cho **constructor** này.

```cpp
class SinhVien {
private:
    string id, name, date;
    double gpa;

public:
    // Constructor with parameters
    SinhVien(string newID, string newName, string newDate, double newGPA) {
        id = newID;
        name = newName;
        date = newDate;
        gpa = newGPA;
    }

    // Display student information
    void displayInfo() const {
        std::cout << "ID: " << id << std::endl;
        std::cout << "Name: " << name << std::endl;
        std::cout << "Date of Birth: " << date << std::endl;
        std::cout << "GPA: " << gpa << std::endl;
    }

    //Accestor...
    //....
};

int main() {
    // Using constructor with parameters
    SinhVien sv1("22120017", "Tien Anh", "21/06/2004", 3.8);
    sv1.displayInfo();

    return 0;
}
```

Hình 3: *VDMH*

1.3.3 Phương thức tạo lập sao chép (copy constructor)

Được dùng để tạo ra một đối tượng từ một đối tượng có sẵn.
VDMH: Trong ví dụ này, chúng ta đã định nghĩa một phương thức tạo lập sao chép **SinhVien(SinhVien& sv)** để sao chép các thuộc tính từ một đối tượng **SinhVien** đã tồn tại. Khi chúng ta khởi tạo **sv2** bằng cách sử dụng sv1, phương thức tạo lập sao chép được gọi tự động, và **sv2** sẽ chứa các giá trị giống với **sv1**.

```cpp
class SinhVien {
private:
    string id, name, date;
    double gpa;

public:
    // Constructor with parameters
    SinhVien(string newID, string newName, string newDate, double newGPA) {
        id = newID;
        name = newName;
        date = newDate;
        gpa = newGPA;
    }

    // Copy constructor
    SinhVien( SinhVien& sv) {
        id = sv.id;
        name = sv.name;
        date = sv.date;
        gpa = sv.gpa;
    }

    // Display student information
    void displayInfo() const {
        cout << "ID: " << id << endl;
        cout << "Name: " << name << endl;
        cout << "Date of Birth: " << date << endl;
        cout << "GPA: " << gpa << endl;
    }

    // Accessor methods
    //...
};
```

(a)

```cpp
int main() {
    // Using constructor with parameters to create a SinhVien object
    SinhVien sv1("22120017", "Tien Anh", "21/06/2004", 3.8);

    // Using copy constructor to create another SinhVien object
    SinhVien sv2 = sv1;

    // Display information of the first student
    cout << "Student 1:" << endl;
    sv1.displayInfo();

    // Display information of the second student
    cout << "\nStudent 2 (Copy of Student 1):" << endl;
    sv2.displayInfo();

    return 0;
}
```

(b)

Hình 4: *VDMH*

1.4 References

- ChatGpt
- TechAcademy, ngày truy cập: 13/03/2024
- cuuduongthancong, giáo trình OOP ngày truy cập 13/03/2024

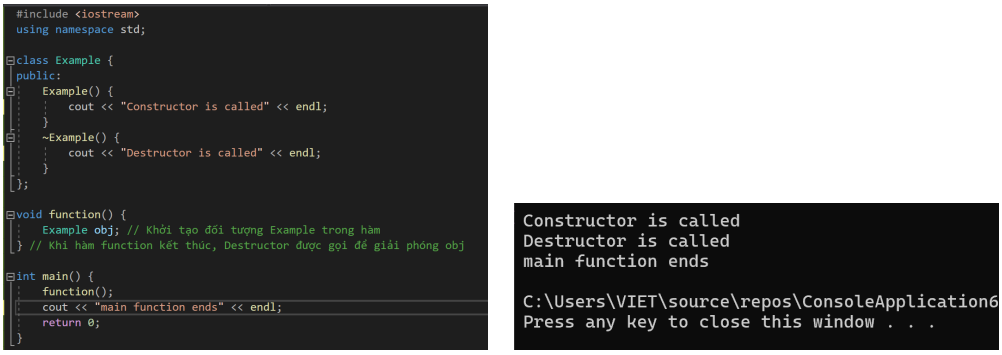
2 Destructor

2.1 Khái niệm

Destructor (hàm hủy) là một hàm đặc biệt của class, nó không có kiểu dữ liệu trả về giống như Constructor và có thêm dấu ~ở phía trước, nhưng nó không có tham số truyền vào.

2.2 Đặc điểm

- Một lớp có thể không có destructor. Trong trường hợp này, trình biên dịch sẽ tạo một destructor mặc định (default destructor).
- Destructor sẽ được gọi khi một đối tượng bị hủy hoặc đối tượng đó không cần thiết nữa. Nghĩa là nó được dùng để giải phóng tài nguyên mà một đối tượng đã chiếm giữ khi đối tượng bị hủy
- Một số ví dụ:
  - Khi đối được khai báo trong hàm và hàm đó đã thực thi xong thì Destructor được gọi để xóa đối tượng đó



(a) (b)

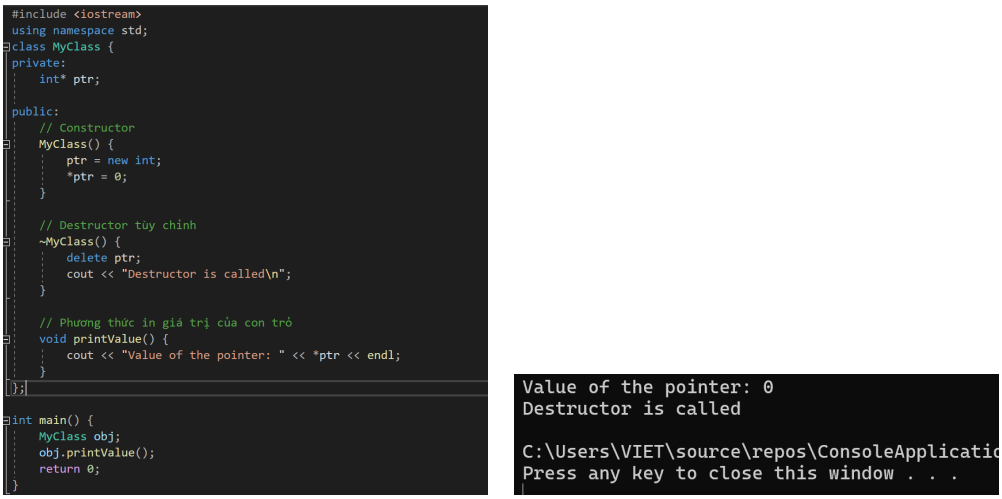
Hình 5: *VD*

  - Khi một biến cục bộ ra khỏi phạm vi hoạt động của nó.

(a) (b)

Hình 6: *VD*

  - Khi một vùng nhớ động được giải phóng bởi toán tử delete,...



```cpp
#include <iostream>
using namespace std;
class MyClass {
private:
    int* ptr;

public:
    // Constructor
    MyClass() {
        ptr = new int;
        *ptr = 0;
    }

    // Destructor tùy chỉnh
    ~MyClass() {
        delete ptr;
        cout << "Destructor is called\n";
    }

    // Phương thức in giá trị của con trỏ
    void printValue() {
        cout << "Value of the pointer: " << *ptr << endl;
    }
};

int main() {
    MyClass obj;
    obj.printValue();
    return 0;
}
```

(a)

```
Value of the pointer: 0
Destructor is called

C:\Users\VIET\source\repos\ConsoleApplicatio
Press any key to close this window . . .
```

(b)

Hình 7: *VD*