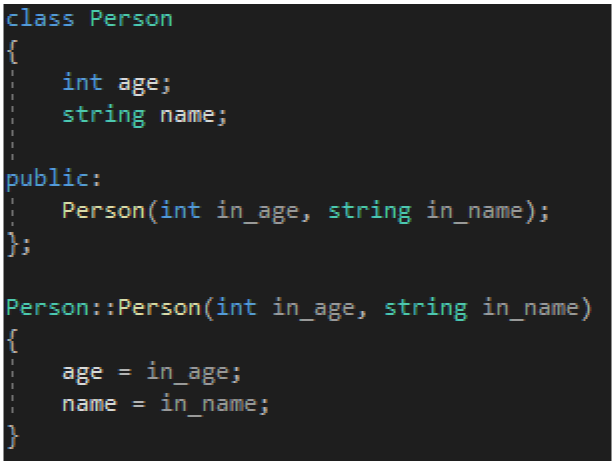
Object Oriented programming and software engineering – Lab 6

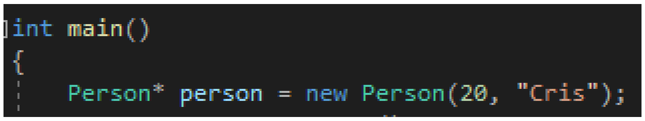
Adam Korytowski – 2025

1. Constructors

Constructor is a special (class) member function, which has the same name as the class. In the constructors body, instructions used to set the initial values of the object elements

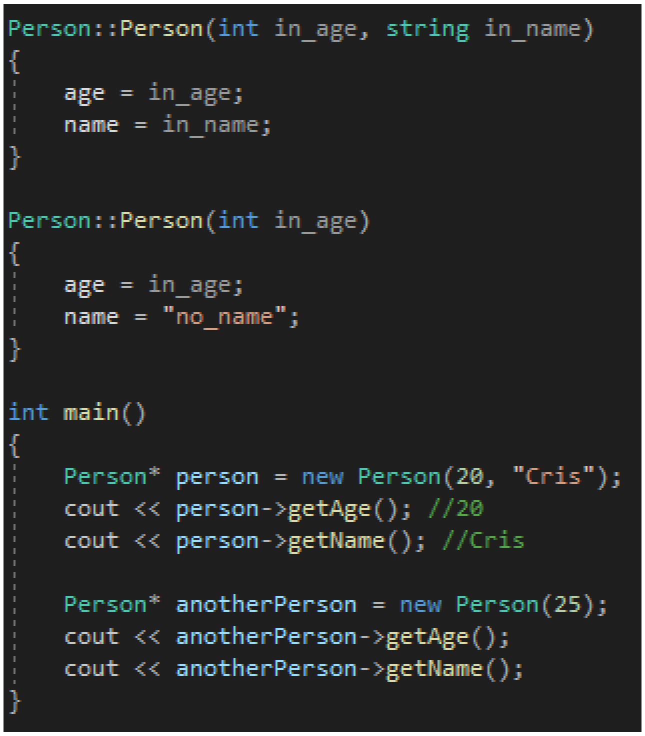


```
class Person
{
    int age;
    string name;

public:
    Person(int in_age, string in_name);
};

Person::Person(int in_age, string in_name)
{
    age = in_age;
    name = in_name;
}
```



```
int main()
{
    Person* person = new Person(20, "Cris");
```

A constructor can be overloaded. It is used so in one class can be more than one version of a constructor (with different number of parameters)



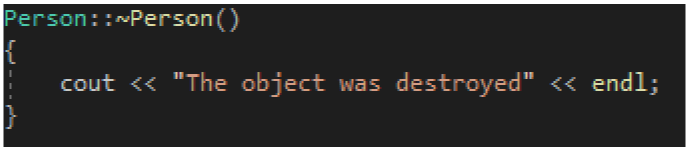
```
Person::Person(int in_age, string in_name)
{
    age = in_age;
    name = in_name;
}

Person::Person(int in_age)
{
    age = in_age;
    name = "no_name";
}

int main()
{
    Person* person = new Person(20, "Cris");
    cout << person->getAge(); //20
    cout << person->getName(); //Cris

    Person* anotherPerson = new Person(25);
    cout << anotherPerson->getAge();
    cout << anotherPerson->getName();
}
```

2. Destructors

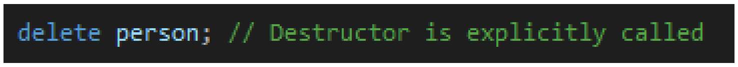
A destructor in C++ is a special member function of a class that is automatically called when an object goes out of scope or is explicitly deleted. It is used to release resources (like memory, file handles, etc.) that the object may have acquired during its lifetime.

Why Use a Destructor?

- To release dynamically allocated memory (delete).
- To close file handles or network connections.
- To clean up resources before an object is destroyed.



```
Person::~Person()
{
    cout << "The object was destroyed" << endl;
}
```



```
delete person; // Destructor is explicitly called
```

3. Create a program with classes interacting with one another, examples:

- classes: Character + Enemy, methods: dealDamage(), spawn(), die(), getHit(), attack(), etc.
- classes: Car + Obstacle, methods: start(), shiftGear(), stop(), getHit(), changeSpeed(), etc.

Requirements – at least:

- 2 classes
- 2 enums
- 4 methods (each class)
- 5 class properties (each class)
- 2 constructors (each class) + a destructor

Demonstrate how the code works by displaying the result of the implemented methods on the screen.