C++ Introduction

Object Oriented Programming

Contents

Think in objects

Just do it, the introduction isn't ready yet!

The first class – Header

```
1  //Student.h
2  //...
3  class Student
4  {
5     public:
6        Student(string name, int id);
7        ~Student();
8
9        string getName();
10       int getID();
11    protected:
12    private:
13       string name;
14       int id;
15 };
```

The first class – Source

```
1  //Student.cpp
2  //...
3  Student::Student(string name, int id)
4  {
5     this->name = name;
6     this->id = id;
7  }
8
9  Student::~Student(){} //empty destructor
10
11 string Student::getName()
12 {
13    return this->name; //same as 'return name;'
14 }
15
16 int Student::getID()
17 {
18    return this->id; //same as 'return id;'
19 }
```

Usage

```
1 Student hans = Student("Hans Vader", 2); //definition
2
3 cout << hans.getName() << endl;
4 //returns 'Hans Vader'
```

Student is a class.
franz is an instance of a student which is an object.

Functions

'this' points to the current object.

Functions

'this' points to the current object.

Example:
Imagine 'Student' has this method:

```
1 string Student::getEverything()
2 { //concatenate name with id
3   return this->getName() + " " + to_string(this->getID());
4 }
```

Functions

'this' points to the current object.

Example:
Imagine 'Student' has this method:

```
1 string Student::getEverything()
2 { //concatenate name with id
3   return this->getName() + " " + to_string(this->getID());
4 }
```

which is the same as this:

```
1 string Student::getEverything()
2 { //concatenate name with id
3   return getName() + " " + to_string(getID());
4 }
```

Attributes

Same function for attributes.

Example:
Imagine 'Student' has this method:

```
1 string Student::getEverything()
2 { //concatenate name with id
3   return this->name + " " + to_string(this->id);
4 }
```

which is the same as this:

```
1 string Student::getEverything()
2 { //concatenate name with id
3   return name + " " + to_string(id);
4 }
```

Constructor

The constructor ...

- is a special function, that is called on creation of an Object.
- is used to initialize all attributes.
- has always the same name as the class.
- can take arguments.

Example:

```
1 Student::Student(string name, int id)
2 { \\initialize name and id
3   this->name = name;
4   this->id = id;
5   this->friend = new Student("Friend", 0815);
6 }
```

Destructor

The destructor ...

- is a special function, that is called on deletion of an Object.
- is used to free memory (if it is needed).
- has always the same name as the class with a '~' in front of it.
- never takes arguments.

Example:

```
1 Student::~Student()
2 { //free memory
3   delete this->friend;
4 }
```

Assignment Operator

The assignment operator ...

- is always called 'operator='.
- is called if an object is assigned (e.g. a = b);
- always takes the other object as argument.
- returns always an object of the same type.
- overwrites the default assignment operator (more on that later).

Example:

```
1 Student& Student::operator=(const Student& other)
2 { //if it isn't called on itself ...
3   if(other != this)
4   { //copy stuff
5     this->name = other->name;
6     this->id = other->id;
7   } //return dereferences pointer on own object
8   return *this;
9 }
```

Copy Constructor

The copy constructor ...

- a special constructor.
- is called if an object is created from another (e.g. int a = int(5));
- always takes the other object as argument.
- doesn't has a return type.
- overwrites the default copy constructor (more on that later).

Example:

```
1 Student::Student(const Student &other)
2 { //copy stuff
3   this->name = other->name;
4   this->id = other->id;
5 }
```