OOP244 Cheat Sheet

by nakul2645 via cheatography.com/200923/cs/42453/

Name and student ID

Name: Krishang jignesh patel

Student ID: 147964225

Encapsulation

Bundling data and methods into a single unit (class)

Constructors and Destructors

The special member function that any object invokes at creation-time is called its class' constructor. We use the default constructor to execute any preliminary logic and set the object to an empty - state.

The special member function that every object invokes before going out of scope is called its class' destructor. We code all of the terminal logic in this special member function.

Unary operators

A unary operation consists of one operator and one operand. The left operand of a unary member operator is the current object. The operator does not take any explicit parameters (with one exception - see post-fix operators below).

Friendship Functions

Friendship grants helper functions access to the private members of a class. By granting friendship status, a class lets a helper function access to any of its private members: data members or member functions. Friendship minimizes class bloat.

Function overloading

C++ supports function overloading, where multiple functions with the same name but different parameter lists are defined, and the compiler selects the appropriate one based on the argument types in a function call.

Con- Des- in array

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

const int NG = 3;

class Student {
    int no;
    float grade[NG];
public:
    Student(int sn = 0, const float* g = nullptr) : no(sn) {
        if (sn > 0 && g) {
            for (int i = 0; i < NG; i++)
                grade[i] = (g[i] >= 0.0f && g[i] <= 100.0f) ? g[i] : 0.0f;
        }
        cout << "In constructor" << endl;
    }

    ~Student() {
        cout << "In destructor for " << no << endl;
    }

    void display() const {
        cout << no << ":\n";
        cout.setf(ios::fixed);
        cout.precision(2);
        for (int i = 0; i < NG; i++) {
            cout.width(8);
            cout << grade[i] << endl;
        }
        cout.unsetf(ios::fixed);
        cout.precision(6);
    }
};

int main() {
    Student student(1234, {89.4f, 67.8f, 45.5f});
    student.display();

    return 0;
}
```

output

```
In constructor
1234:
 89.40
 67.80
 45.50
In destructor for 1234
```

Friendship Functions EG

```
#include <iostream>

class MyClass {
private:
    int privateData;

public:
    MyClass(int data) : privateData(data) {}

    // Declaration of a friend function
    friend void displayPrivateData(const MyClass& obj);
};

// Definition of the friend function
void displayPrivateData(const MyClass& obj) {
    std::cout << "Accessing private data from friend function: " << obj.privateData << std::
}

int main() {
    MyClass myObject(42);

    // Call the friend function from main
    displayPrivateData(myObject);

    return 0;
}
```

Dynamic Memory

'new' and 'delete' operators: Dynamically allocate and deallocate memory.

The memory that an application obtains from the operating system during execution is called dynamic - memory.

Dynamic memory is distinct from the static memory. While the operating system allocates static memory for an application at load time, the system reserves dynamic memory, allocates it and deallocates it at - run-time.

Current object (this()) EG

```
Student Student::display() const {

    // ...

    return *this;
}
int main() {
    float gh[] = {89.4f, 67.8f, 45.5f};
    Student harry(1234, gh, 3), backup;
    backup = harry.display();
    backup.display();
}
```

```
Entering 3-arg constructor
1234:
 89.40
 67.80
 45.50
Entering destructor for 1234
Entering destructor for 1234
Entering destructor for 1234
Entering destructor for 1234
```

Binary Operators

A binary operation consists of one operator and two operands. In a binary member operator, the left operand is the current object and the member function takes one explicit parameter: the right operand.

Dynamic Memory

```
int* dynamicInt = new int;
*dynamicInt = 5;
delete dynamicInt;
```

Constructors and Destructors EG

```
#include <iostream>

class ResourceManager {
private:
    int* resource;

public:
    // Constructor allocates dynamic memory
    ResourceManager() {
        std::cout << "Constructor - Allocating dynamic memory" << st
        resource = new int[5];
    }

    // Destructor deallocates dynamic memory
    ~ResourceManager() {
        std::cout << "Destructor - Deallocating dynamic memory" << s
        delete[] resource;
    }
};

int main() {
    // Default constructor is called
    ResourceManager defaultResource;

    // Destructor is automatically called when defaultResource goes

    // Creating a new object with parameterized constructor
    {
        ResourceManager parameterizedResource;

        // Destructor is automatically called when parameterizedReso
    }

    return 0;
}
```

keyword

The this keyword in C++ returns the address of the current object, representing the memory region containing all instance variables. *this refers to the current object itself, encompassing its complete set of instance variables, and is used within a member function to access these variables through implicit - parameters.

Helper Fucntions

In object-oriented programming, helper functions provide external support to a class by accepting explicit parameters. These functions access class objects solely through their parameters, often including at least one parameter of the class type. Well-encapsulated classes may utilize helper functions for additional logic.

By **nakul2645**
cheatography.com/nakul2645/

Not published yet.
Last updated 20th February, 2024.