ChatGPT

Лабораторная работа №22. Перегрузка операций

1. Постановка задачи

Общая: - Определить пользовательский класс. - Реализовать конструкторы (по умолчанию, с параметрами, копирования). - Определить деструктор. - Создать селекторы и модификаторы. - Перегрузить оператор присваивания. - Перегрузить операции ввода и вывода. - Перегрузить операции, указанные в варианте. - Написать программу, демонстрирующую работу с объектами.

Вариант №3: Создать класс `Money` с двумя полями: `long` для рублей и `int` для копеек. Реализовать: - сложение суммы и дробного числа; - операции сравнения `>`, `<`, `==`; - перегрузку оператора присваивания `=`; - ввод/вывод через потоки `>>` и `<<`.

2. Анализ задачи

Функции: - `Money()` — конструктор по умолчанию. - `Money(long, int)` — конструктор с параметрами. - `Money(const Money&)` — конструктор копирования. - `~Money()` — деструктор. - `getRubles()`, `getKopecks()` — селекторы. - `setRubles(long)`, `setKopecks(int)` — модификаторы. - `operator=`, `operator+`, `operator>`, `operator<`, `operator==` — перегрузки. - `friend operator<<`, `operator>>` — дружественные функции для ввода/вывода.

Функция main(): - Вывод текстового меню. - Обработка выбора пользователя через `switch-case`. - Демонстрация работы всех операций.

3. Блок-схема

См. прилагаемое изображение: `money_flowchart.png`.

4. Текст программы

(Файлы: `Money.h`, `Money.cpp`, `main.cpp`)

// Money.h

```
#ifndef MONEY_H
#define MONEY_H
#include <iostream>
class Money {
private:
    long rubles;
    int kopecks;
    void normalize();
public:
```

```
    Money();
    Money(long r, int k);
    Money(const Money& other);
    ~Money();
    long getRubles() const;
    int getKopecks() const;
    void setRubles(long r);
    void setKopecks(int k);
    Money& operator=(const Money& other);
    Money operator+(double value) const;
    bool operator==(const Money& other) const;
    bool operator>(const Money& other) const;
    bool operator<(const Money& other) const;
    friend std::ostream& operator<<(std::ostream& os, const Money& money);
    friend std::istream& operator>>(std::istream& is, Money& money);
    friend Money operator+(double value, const Money& money);
};
#endif
```

// Money.cpp

```
#include "Money.h"
#include <cmath>
void Money::normalize() {
    if (kopecks >= 100 || kopecks < 0) {
        rubles += kopecks / 100;
        kopecks = kopecks % 100;
    }
    if (kopecks < 0) {
        rubles--;
        kopecks += 100;
    }
}
Money::Money() : rubles(0), kopecks(0) {}
Money::Money(long r, int k) : rubles(r), kopecks(k) { normalize(); }
Money::Money(const Money& other) { rubles = other.rubles; kopecks =
other.kopecks; }
Money::~Money() {}
long Money::getRubles() const { return rubles; }
int Money::getKopecks() const { return kopecks; }
void Money::setRubles(long r) { rubles = r; }
void Money::setKopecks(int k) { kopecks = k; normalize(); }
Money& Money::operator=(const Money& other) {
    if (this != &other) {
        rubles = other.rubles;
        kopecks = other.kopecks;
```

```
    }
    return *this;
}
Money Money::operator+(double value) const {
    long r = (long)value;
    int k = round((value - r) * 100);
    Money temp(rubles + r, kopecks + k);
    temp.normalize();
    return temp;
}
bool Money::operator==(const Money& other) const {
    return rubles == other.rubles && kopecks == other.kopecks;
}
bool Money::operator>(const Money& other) const {
    return (rubles > other.rubles) || (rubles == other.rubles && kopecks >
other.kopecks);
}
bool Money::operator<(const Money& other) const {
    return !(*this > other || *this == other);
}
std::ostream& operator<<(std::ostream& os, const Money& money) {
    os << money.rubles << "," << (money.kopecks < 10 ? "0" : "") <<
money.kopecks;
    return os;
}
std::istream& operator>>(std::istream& is, Money& money) {
    char comma;
    is >> money.rubles >> comma >> money.kopecks;
    money.normalize();
    return is;
}
Money operator+(double value, const Money& money) {
    long r = (long)value;
    int k = round((value - r) * 100);
    Money temp(money.rubles + r, money.kopecks + k);
    temp.normalize();
    return temp;
}
```

// main.cpp

```
#include <iostream>
#include "Money.h"
using namespace std;
int main() {
    Money m1, m2;
```

```
    double val;
    int choice;
    do {
        cout << "\nМеню:\n";
        cout << "1. Ввести первую сумму\n";
        cout << "2. Ввести вторую сумму\n";
        cout << "3. Прибавить к первой дробное число\n";
        cout << "4. Сравнить суммы\n";
        cout << "5. Вывести суммы\n";
        cout << "0. Выход\n";
        cout << "Выбор: ";
        cin >> choice;
        switch (choice) {
            case 1:
                cout << "Введите первую сумму (формат: рубли,копейки): ";
                cin >> m1;
                break;
            case 2:
                cout << "Введите вторую сумму (формат: рубли,копейки): ";
                cin >> m2;
                break;
            case 3:
                cout << "Введите дробное число для прибавления: ";
                cin >> val;
                m1 = m1 + val;
                cout << "Результат: " << m1 << endl;
                break;
            case 4:
                cout << "m1 > m2: " << (m1 > m2 ? "Да" : "Нет") << endl;
                cout << "m1 < m2: " << (m1 < m2 ? "Да" : "Нет") << endl;
                cout << "m1 == m2: " << (m1 == m2 ? "Да" : "Нет") << endl;
                break;
            case 5:
                cout << "Первая сумма: " << m1 << endl;
                cout << "Вторая сумма: " << m2 << endl;
                break;
        }
    } while (choice != 0);
    return 0;
}
```

5. Тесты

Ввод m1	Ввод m2	Операция	Результат
10,50	5,25	m1 + 2.75	13,25

Ввод m1	Ввод m2	Операция	Результат
10,50	5,25	m1 > m2	Да
5,25	5,25	m1 == m2	Да
5,00	10,00	m1 < m2	Да

Готово.