МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное

учреждение высшего образования

"Томский государственный университет систем управления и

радиоэлектроники" (ТУСУР)

Кафедра телевидения и управления (ТУ)

ОБЪЕКТЫ И КЛАССЫ

Лабораторная работа по объектно-ориентированному программированию

Выполнил студент гр.112

\_\_\_\_\_\_\_К.Д.Нигматуллин

Дата выполнения работы

«\_\_\_» «\_\_\_\_\_\_\_\_» 2023 г.

Преподаватель

Ассистент кафедры ТУ

\_\_\_\_\_\_\_К.А.Рылов

Дата проверки работы

«\_\_\_» «\_\_\_\_\_\_\_\_» 2023 г.

Томск 2023

Оглавление

[**ВВЕДЕНИЕ** 3](#_Toc148994583)

[**2 ХОД РАБОТЫ** 4](#_Toc148994584)

[2.1 Решение задания 1: 4](#_Toc148994587)

[2.2 Решение задания 2: 5](#_Toc148994588)

[2.3 Решение задания 3: 7](#_Toc148994589)

[2.4 Решение задания 4: 9](#_Toc148994590)

[**ЗАКЛЮЧЕНИЕ** 12](#_Toc148994591)

**ВВЕДЕНИЕ**

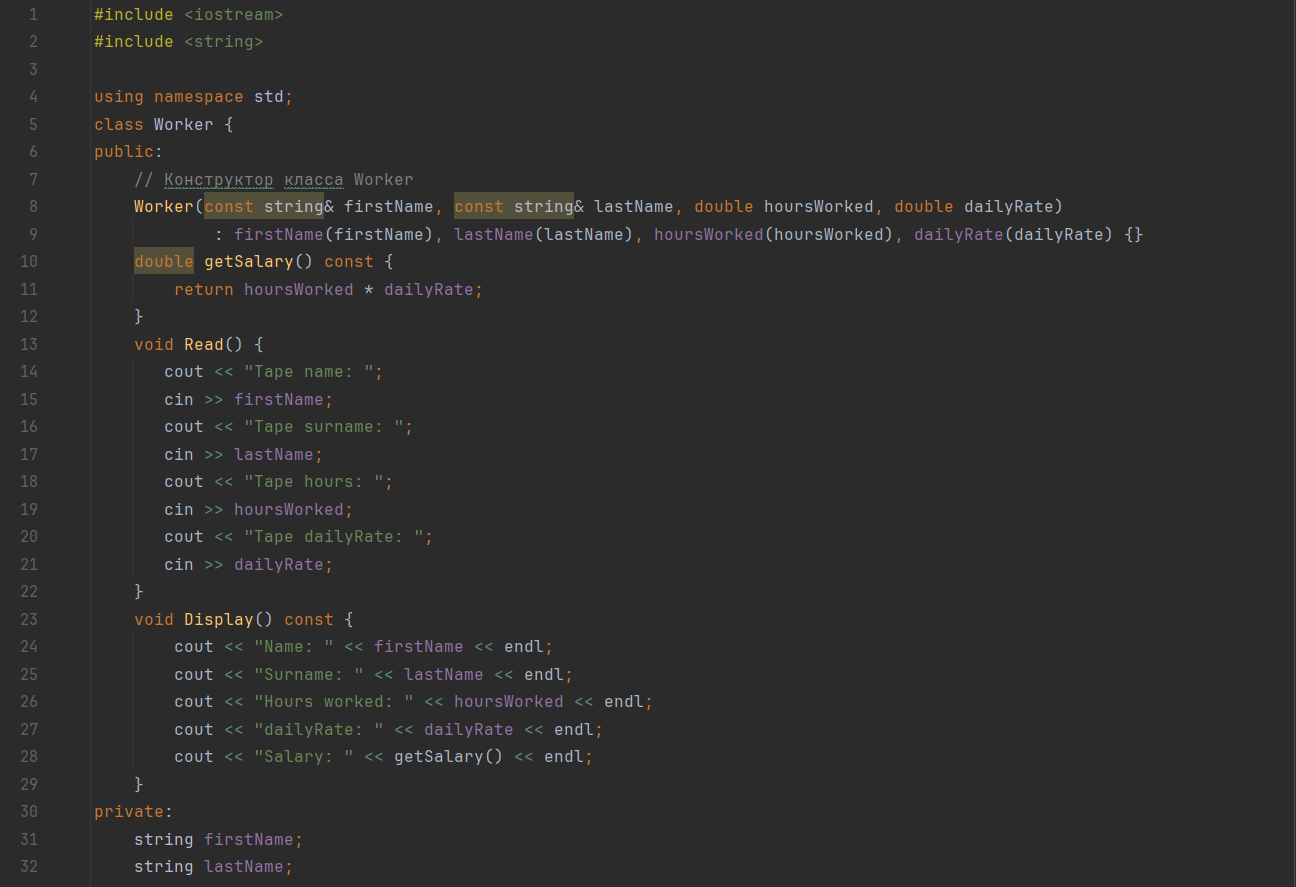
Цель работы – изучить структуру классов, особенности реализации методов класса и создания объектов; приобрести навыки в работе с классами на языке C++.Объектно-ориентированное программирование является одним из наиболее эффективных и востребованных подходов в современном программировании. Объектно-ориентированное программирование представляет собой парадигму программирования, в основе которой лежат понятия объектов и классов. На данной лабораторной работе предстояло решить приведенные ниже задачи:

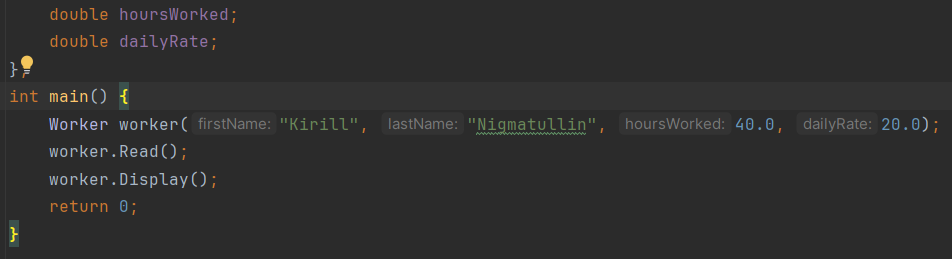
1. Реализовать класс Worker с методами: имя, фамилия, число отработанных часов и ставка за день работы. Реализовать метод getSalary, который будет вычислять зарплату работника.
2. Реализовать класс ResistanceCalculator с полями: сопротивление первого резистора, сопротивление второго резистора и способ соединения (последовательно / параллельно). Разработать метод вычисления сопротивления соединения.
3. Реализовать класс Date с тремя полями: день, месяц, год. Реализовать два конструктора инициализации: с тремя аргументами и одним целым числом вида ггггммдд. В классе реализовать методы получения и изменения отдельных полей даты и метод вычисления даты через заданное число дней.
4. Реализуйте класс Money с двумя полями: рубли и копейки. Реализуйте методы сложения, вычитания и деления двух денежных сумм, деления и умножение сумм на дробное число. Реализованные методы в качестве входного и возвращаемого аргументов должны использовать объекты класса Money.

**2 ХОД РАБОТЫ**

2. 1. Решение задания 1:

Текст программы для реализации первого задания представлен ниже:





Блок-схема программы (рисунок 1.1):

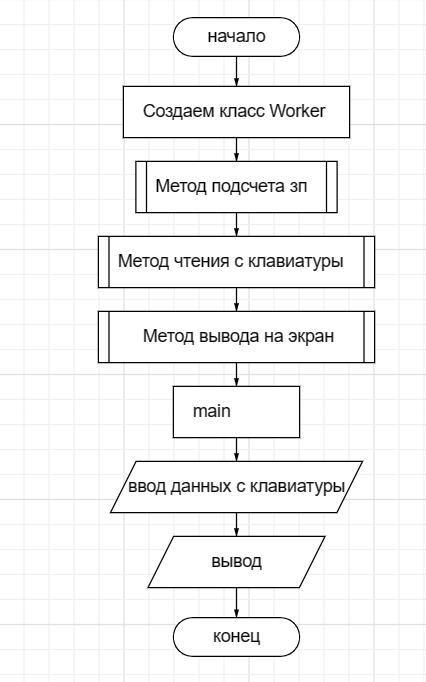


Рисунок 1.1 – Блок-схема программы.

Результат тестирования программы (рисунок 1.2):

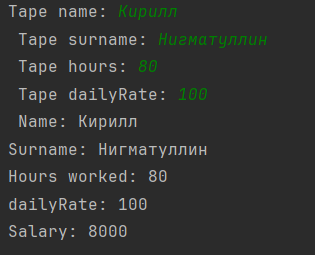
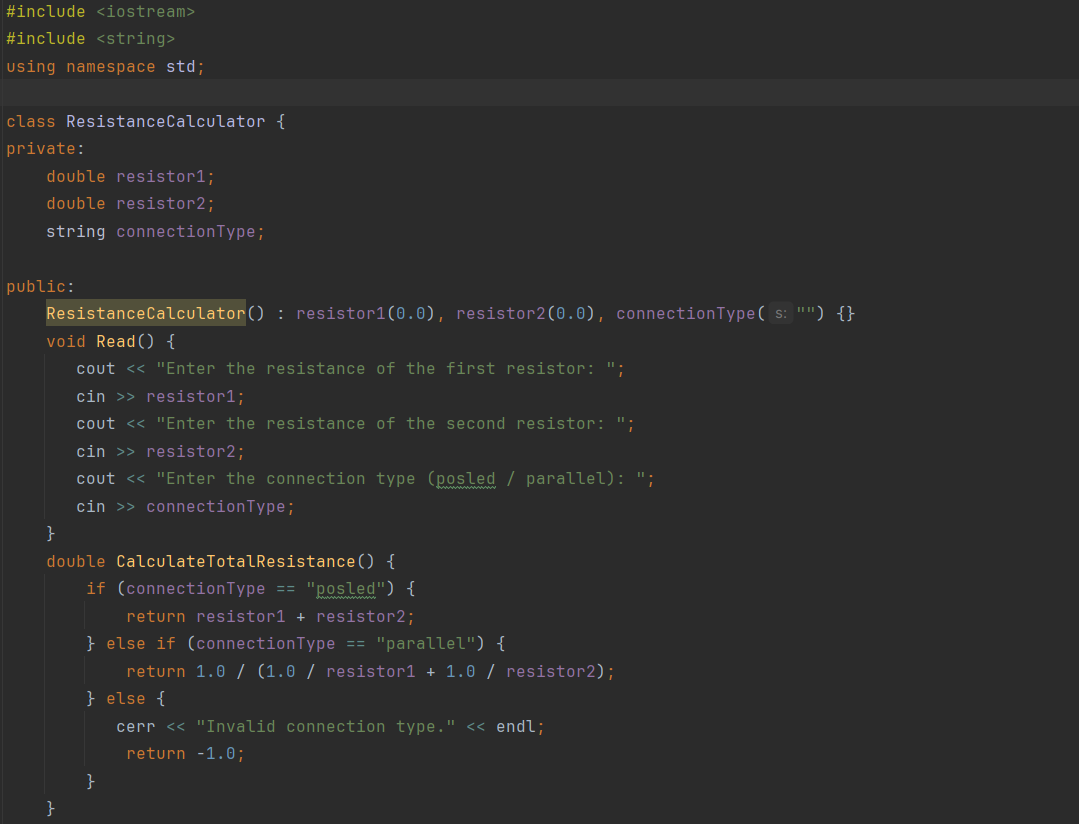
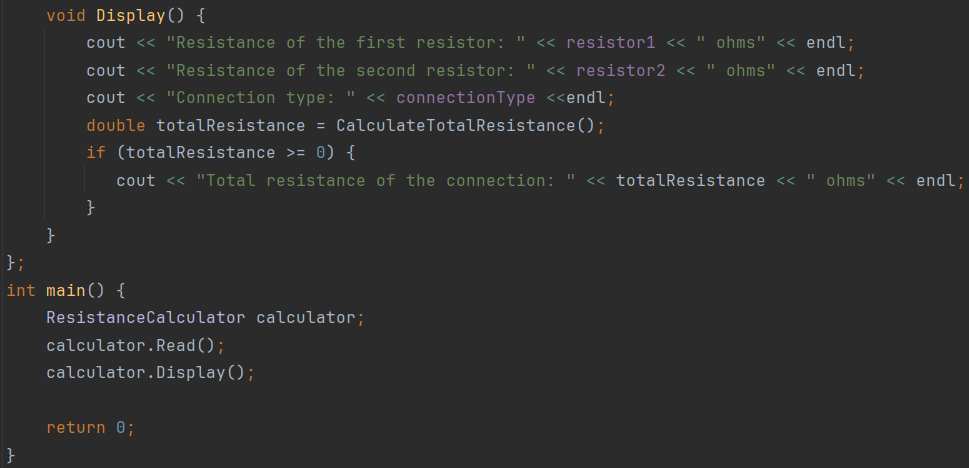


Рисунок 1.2 – Результат работы программы

* 1. Решение задания 2:

Текст программы 2:





Блок-схема программы (рисунок 2.1)



Рисунок 2.1 – Блок-схема программы.

Результат тестирования программы (рисунок 2.2):

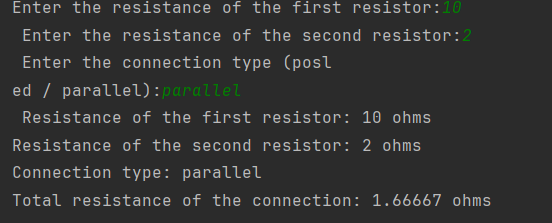
**

Рисунок 2.2 – Результат работы программы

* 1. Решение задания 3:

Текст программы 3:

#include <iostream>  
using namespace std;  
class Date {  
private:  
 int year;  
 int month;  
 int day;  
  
public:  
 Date(int y, int m, int d) : year(y), month(m), day(d) {}  
  
 Date(int dateInt) {  
 year = dateInt / 10000;  
 month = (dateInt % 10000) / 100;  
 day = dateInt % 100;  
 }  
 int getYear() const {  
 return year;  
 }  
 int getMonth() const {  
 return month;  
 }  
 int getDay() const {  
 return day;  
 }  
 void setYear(int y) {  
 year = y;  
 }  
 void setMonth(int m) {  
 month = m;  
 }  
 void setDay(int d) {  
 day = d;  
 }  
 void addDays(int days) {  
 day += days;  
 while (day > daysInMonth()) {  
 day -= daysInMonth();  
 if (month == 12) {  
 month = 1;  
 year++;  
 } else {  
 month++;  
 }  
 }  
 }  
 int daysInMonth() const {  
 if (month == 2) {  
 return 28;  
 } else if (month == 4 || month == 6 || month == 9 || month == 11) {  
 return 30;  
 } else {  
 return 31;  
 }  
 }  
 void Read() {  
 cout << "Enter the year: ";  
 cin >> year;  
 cout << "Enter the month: ";  
 cin >> month;  
 cout << "Enter the day: ";  
 cin >> day;  
 int daysToAdd;  
 cout << "Enter the number of days to add: ";  
 cin >> daysToAdd;  
 addDays(daysToAdd);  
 }  
 void Display() {  
 cout << "Date: " << year << " year, " << month << " month, " << day << " day" << endl;  
 }  
};  
int main() {  
 Date date1(2023, 10, 17);  
 date1.Read();  
 date1.Display();  
 return 0;  
}

Результат тестирования программы (рисунок 3.1):

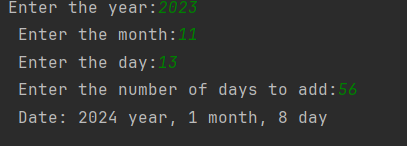


Рисунок 3.1 – Результат работы программы

* 1. Решение задания 4:

Текст программы 4:

#include <iostream>  
#include <string>  
using namespace std;  
class Money {  
private:  
 int rubles;  
 int kopecks;  
  
public:  
 Money() : rubles(0), kopecks(0) {}  
 Money(int rubles, int kopecks) : rubles(rubles), kopecks(kopecks) {}  
 void Read() {  
 string action;  
 cout << "Enter the action (+, -, \*, /): ";  
 cin >> action;  
 if (action == "+" || action == "-" || action == "\*" || action == "/") {  
 cout << "Enter your amount in rubles: ";  
 cin >> rubles;  
 cout << "Enter your amount in kopecks: ";  
 cin >> kopecks;  
 }  
 if (action == "+") {  
 int rubles2, kopecks2;  
 cout << "Enter the second amount in rubles: ";  
 cin >> rubles2;  
 cout << "Enter the second amount in kopecks: ";  
 cin >> kopecks2;  
 Money other(rubles2, kopecks2);  
 \*this = Add(other);  
 } else if (action == "-") {  
 int rubles2, kopecks2;  
 cout << "Enter the amount to subtract in rubles: ";  
 cin >> rubles2;  
 cout << "Enter the amount to subtract in kopecks: ";  
 cin >> kopecks2;  
 Money other(rubles2, kopecks2);  
 \*this = Subtract(other);  
 } else if (action == "\*") {  
 double multiplier;  
 cout << "Enter a number to multiply your money by: ";  
 cin >> multiplier;  
 \*this = Multiply(multiplier);  
 } else if (action == "/") {  
 double divisor;  
 cout << "Enter a number to divide your money by: ";  
 cin >> divisor;  
 \*this = Divide(divisor);  
 } else {  
 cerr << "Unknown action: " << action <<endl;  
 }  
 }  
 void Display() const {  
 cout << "Result: " << rubles << " rubles and " << kopecks << " kopecks" << endl;  
 }  
 Money Add(const Money& other) const {  
 int totalRubles = rubles + other.rubles;  
 int totalKopecks = kopecks + other.kopecks;  
 if (totalKopecks >= 100) {  
 totalRubles += totalKopecks / 100;  
 totalKopecks %= 100;  
 }  
 return Money(totalRubles, totalKopecks);  
 }  
 Money Subtract(const Money& other) const {  
 int totalRubles = rubles - other.rubles;  
 int totalKopecks = kopecks - other.kopecks;  
 if (totalKopecks < 0) {  
 totalRubles -= 1;  
 totalKopecks += 100;  
 }  
 return Money(totalRubles, totalKopecks);  
 }  
 Money Multiply(double factor) const {  
 int totalKopecks = static\_cast<int>((rubles \* 100 + kopecks) \* factor);  
 int totalRubles = totalKopecks / 100;  
 totalKopecks %= 100;  
 return Money(totalRubles, totalKopecks);  
 }  
 Money Divide(double divisor) const {  
 if (divisor == 0.0) {  
 cerr << "Error: Division by zero." << endl;  
 return Money(0, 0);  
 }  
 double totalAmount = rubles \* 100 + kopecks;  
 totalAmount /= divisor;  
 int totalRubles = static\_cast<int>(totalAmount / 100);  
 int totalKopecks = static\_cast<int>(totalAmount) % 100;  
 return Money(totalRubles, totalKopecks);  
 }  
};  
int main() {  
 Money;  
 money.Read();  
 money.Display();  
  
 return 0;  
}

Результат тестирования программы (рисунок 4.1):

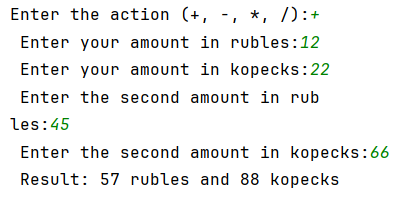


Рисунок 4.1 – Результат работы программы

**ЗАКЛЮЧЕНИЕ**

Изучив структуру классов, особенности реализации методов класса и создания объектов на языке C++, мы получаем возможность создавать и использовать пользовательские типы данных, что позволяет абстрагироваться от сложности программы и упростить управление данными и функциональностью. Изучение классов на C++ позволяет организовать код в более логичные и поддерживаемые структуры, что облегчает разработку и обслуживание программ. Это мощный инструмент для создания объектно-ориентированных приложений и обеспечения модульности и повторного использования кода.