Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики

и радиоэлектроники»

Специальность «Программная инженерия»

Кафедра инженерной психологии и эргономики

Учебная дисциплина «Основы алгоритмизации и программирования»

ОТЧЕТ

по лабораторной работе No8

«Функции»

Вариант 9

Подготовил: Грибач Н.Э.

Проверил: Усенко Ф.В.

Минск 2024

**Цель работы**: сформировать навыки и умения обработки структурированных типов данных, организованных в виде функций.

**Задание 1**: Дано натуральное число Р. Напишите программу нахождения всех натуральных чисел, не превосходящих Р, которые можно представить в виде произведения двух простых чисел.

**Задание 2**:

а) для сложения десятичных дробей;

б) для сложения обыкновенных дробей.

Задание 1.

#include <iostream>

using namespace std;

// Функция для проверки, является ли число простым

bool isPrimeNum(int num)

{

    for (int i = 2; i \* i <= num; i++)

    {

        if (num % i == 0) // Если найден делитель, число не является простым

            return false;

    }

    return true; // Если делителей нет, число простое

}

// Функция для нахождения всех простых чисел от 2 до P

void setPrimeNums(int \*&primeNums, int &size, int P)

{

    int capacity = 10;             // начальный размер массива

    primeNums = new int[capacity]; // выделяем память

    for (int i = 2; i <= P; i++)

    {

        if (isPrimeNum(i)) // Проверяем, является ли число простым

        {

            if (size == capacity) // если массив заполнен

            {

                capacity \*= 2;                 // увеличиваем размер

                int \*temp = new int[capacity]; // создаем новый массив большего размера

                for (int j = 0; j < size; j++) // копируем данные

                {

                    temp[j] = primeNums[j];

                }

                delete[] primeNums; // освобождаем старый массив

                primeNums = temp; // указываем указатель со старого массива на новый массив

            }

            primeNums[size++] = i; // добавляем простое число

        }

    }

}

// Функция для вывода всех произведений пар простых чисел, не превышающих P

void getNaturalNums(int \*&primeNums, int &size, int P)

{

    for (int i = 0; i < size; i++)

    {

        for (int j = i; j < size; j++)

        {

            int numberValue = primeNums[i] \* primeNums[j];

            if (numberValue <= P)

            {

                cout << primeNums[i] << " \* " << primeNums[j] << " = " << numberValue << endl;

            }

        }

    }

}

int main()

{

    int P;

    cout << "Введите натуральное число: ";

    cin >> P;

    int \*primeNums;           // Указатель на массив простых чисел

    int size = 0;             // текущий размер массива простых чисел

    setPrimeNums(primeNums, size, P);

    getNaturalNums(primeNums, size, P);

    delete[] primeNums; // освобождаем память

    return 0;

}

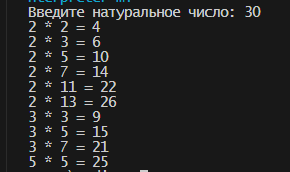


Рисунок 1 – Снимок результата работы программы 1

Блок-схема работы программы представлена на рисунке 2.

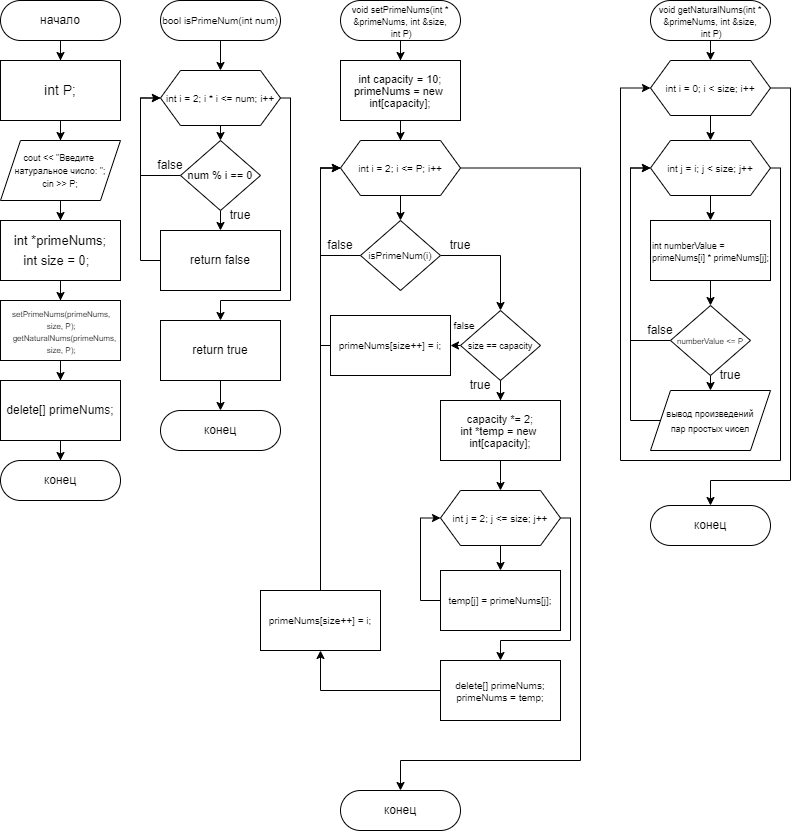


Рисунок 2

Задание 2.

#include <iostream>

#include <numeric> // Для функции lcm (НОК)

using namespace std;

class FractionCalculator

{

public:

    // Метод для сложения десятичных дробей

    void solutionDouble(double num1, double num2, double &result)

    {

        result = num1 + num2;

    }

    // Метод для сложения обыкновенных дробей

    void solutionDouble(int nmtr1, int dmtr1, int nmtr2, int dmtr2)

    {

        int nok = std::lcm(dmtr1, dmtr2); // Находим наименьшее общее кратное знаменателей

        if (dmtr1 == dmtr2)

        { // Если знаменатели одинаковы

            if ((nmtr1 + nmtr2) == nok)

            { // Если сумма числителей равна НОК знаменателей

                int resultNmtr = (nmtr1 + nmtr2) / nok;

                cout << "Результат: " << resultNmtr;

            }

            else

            {

                int resultNmtr = nmtr1 + nmtr2;

                cout << resultNmtr << "/" << dmtr1;

            }

        }

        else

        {

            // Приводим дроби к общему знаменателю

            int newNmtr1 = (nok / dmtr1) \* nmtr1;

            int newNmtr2 = (nok / dmtr2) \* nmtr2;

            int resultNmtr = newNmtr1 + newNmtr2;

            if (resultNmtr % nok == 0)

            { // Если сумма числителей делится без остатка

                int result = resultNmtr / nok;

                cout << result;

                return;

            }

            if (resultNmtr >= nok)

            { // Если числитель больше знаменателя

                int chastnoe = resultNmtr / nok;

                int raznost = resultNmtr - (nok \* chastnoe); // Остаток

                cout << "Не преобразованная дробь: " << resultNmtr << "/" << nok << endl;

                cout << "Преобразованная дробь: " << chastnoe << "+" << raznost << "/" << nok;

                return;

            }

            // Если числитель < знаменателя => выводим десятичную и обыкновенную

            double result = (double)resultNmtr / (double)nok;

            cout << result << " или " << resultNmtr << "/" << nok;

        }

    }

};

int main()

{

    FractionCalculator calculator;

    int selectedNum;

    cout << "Выберите, какие дроби будете складывать: 1 - десятичные, 2 - обыкновенные: ";

    cin >> selectedNum;

    if (selectedNum == 1)

    {

        double num1;

        double num2;

        double result;

        cout << "Введите первую десятичную дробь: ";

        cin >> num1;

        cout << "Введите вторую десятичную дробь: ";

        cin >> num2;

        calculator.solutionDouble(num1, num2, result); // Вызываем метод сложения десятичных дробей

        cout << "Ответ: " << result;

    }

    else

    {

        int nmtr1, nmtr2; // Числители

        int dmtr1, dmtr2; // Знаменатели

        cout << "Введите числитель первой дроби: ";

        cin >> nmtr1;

        cout << "Введите знаменатель первой дроби: ";

        cin >> dmtr1;

        cout << "Введите числитель второй дроби: ";

        cin >> nmtr2;

        cout << "Введите знаменатель второй дроби: ";

        cin >> dmtr2;

        calculator.solutionDouble(nmtr1, dmtr1, nmtr2, dmtr2); // Вызываем метод сложения обыкновенных дробей

    }

    return 0;

}

Рисунки 3-6 – Снимки результата работы программы 2

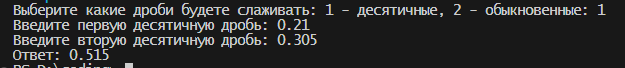


Рисунок 3 – результат сложения десятичных дробей

Рисунки 4-6 – результаты сложения обыкновенных дробей при разных значениях

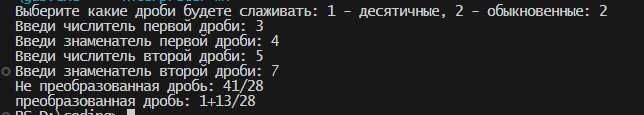


Рисунок 4

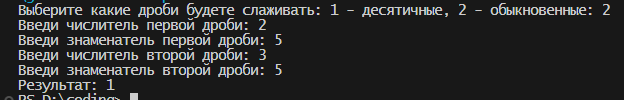


Рисунок 5

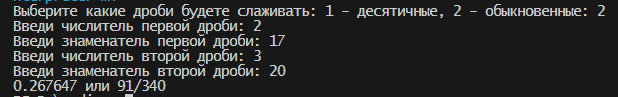


Рисунок 6

Блок-схема работы программы представлена на рисунке 7

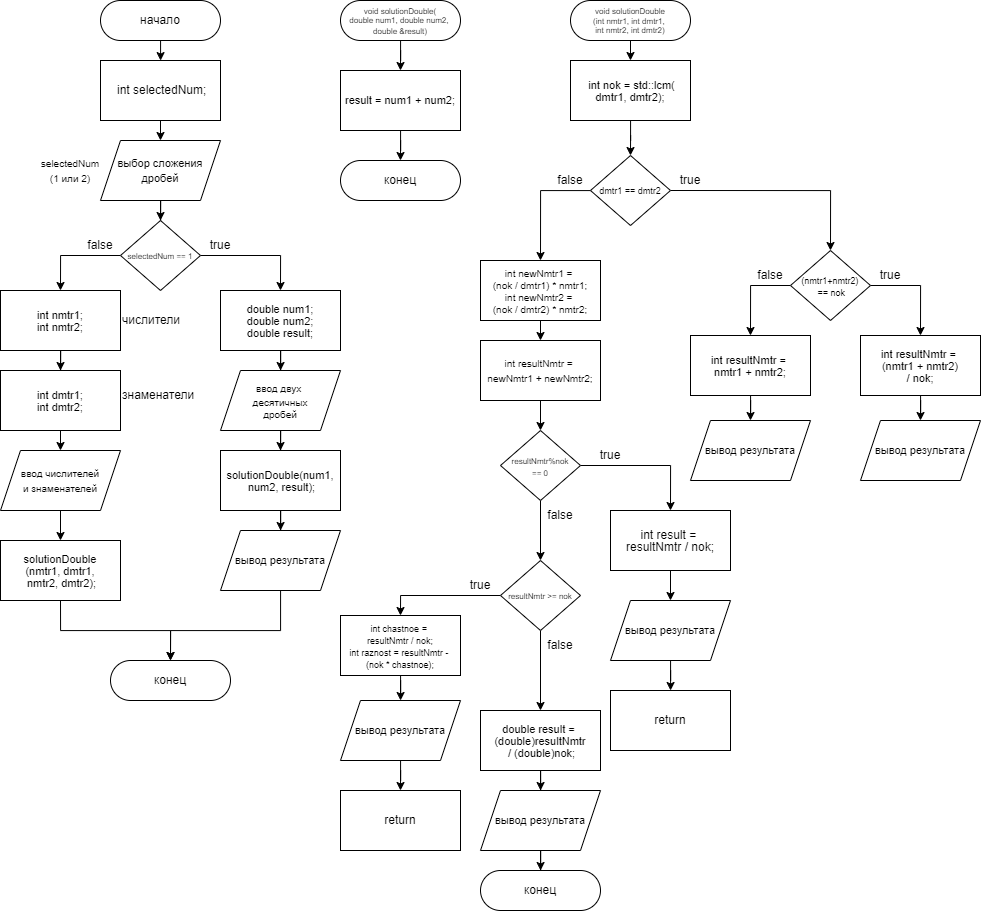


Рисунок 7

**Вывод**: В результате работы были созданы две программы содержащие функции. Были использованы функции типа void, которые не возвращают значений и типа bool, которые возвращают true or false. Во второй программе были созданы 2 перегруженные функции. Использование функций позволило облегчить понимание основной структуры программы.