МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

АЭРОКОСМИЧЕСКОГО ПРИБО6РОСТРОЕНИЯ»

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Старший преподаватель |  |  |  | С.А. Рогачев |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ |
| «*Простой генетический алгоритм*» |
| по курсу: ЭВОЛЮЦИОННЫЕ МЕТОДЫ ПРОЕКТИРОВАНИЯ ПРОГРАММНО-ИНФОРМАЦИОННЫХ СИСТЕМ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4136 |  | 13.09.24 |  | А.С. Вилюмсон |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2024

**1. Цель работы**

Необходимо преобразовать заданную арифметическую функцию в рекурсивную, используя операторы примитивной рекурсии.

**2. Основные сведения из теории**

Функция – форма отображения. Функцию для которой существует вычисляющей её алгоритм называют вычисляемой.  
Рекурсия (лат. recurso) — есть такой способ задания вычислимой функции, когда каждое её значение задаётся через значение этой функции или како-либо другой вычислимой функции, через меньшие или ранее определённые значения аргумента.

В Теории Рекурсивных Функций все функции строятся на 1) элементарных (базисных) функциях и 2) операциях для операторов арифметических преобразований функции.

Базисные функции это:

1. Нулевая функция *O*, всегда возвращающая 0
2. Функция проекции тождества
3. Функция непосредственного следования

Операции:

1. Оператор суперпозиции
2. Оператор примитивной рекурсии

**3. Постановка задачи;**

Необходимо преобразовать заданную арифметическую функцию в рекурсивную, используя операторы примитивной рекурсии. Во второй части лабораторной работы требуется создать программу на языке высокого уровня содержащую две функции. Первая функция должна вычислять заданную арифметическую функцию итеративным способом. Вторая функция должна вычислять заданную арифметическую функцию рекурсивным способом.

Функция 36.

**4. Преобразование арифметической функции в рекурсивную с использованием**

**операторов примитивной рекурсии**

Для начала преобразуем функцию сложения, её можно представить как примитивно рекурсивную функцию от двух переменных:

Тогда функцию умножения можно также представить как рекурсивную:

**5. Листинг программы на языке высокого уровня с комментариями**

#include <iostream>

#include <functional>

#include <vector>

#include <chrono>

/// *@brief* Класс с функциями рекурсии

*class* PrimitiveReqursion{

*public:*

    //элементарные функции

*int* o(){ return *int*(0);}

*int* n(const *int*& *n*){return {*n*+1};}

*int* i(const std::vector<*int*>& *n*,size\_t *i*){

        if(*i*>=1 || *i*<=*n*.size()){

            return *n*.at(*i*-1);

        }else{

            throw std::out\_of\_range("Index out of range");

        }

    }

    //элементарные операторы

*int* S(std::function<*int*(std::vector<*int*>)> *f*, std::vector< std::function<*int*(const std::vector<*int*>)>> *g*,const std::vector<*int*>& *xn*){

        std::vector<*int*> xm = {};

        for(const *auto*& gi: *g*){

            xm.push\_back( gi(*xn*));

        }

    return  *f*(xm);

    }

*int* R(std::function<*int*(const std::vector<*int*>&)> *f*, std::function<*int*(const std::vector<*int*>&,*int*,*int*)> *g*, std::vector<*int*> *xn*, *int* *y*){

        if(*y* == 0){

            return *f*(*xn*);

        }

        else{

            return *g*(*xn*,*y*-1,R(*f*,*g*,*xn*,*y*-1));

        }

    }

    //оператор сложения на основе базовых

*int* sum(*int* *x*, *int* *y*) {

*auto* base\_function = [this](const std::vector<*int*>& *xn*) -> *int* {

            return this->i(*xn*,1);

        };

*auto* I3\_3 = [this](const std::vector<*int*>& *xn*) -> *int* {

            return this->i(*xn*,3);

        };

*auto* step\_up = [this](const std::vector<*int*>& *xn*) -> *int* {

            return n(*xn*[0]);

        };

*auto* recursive\_function = [this, step\_up,I3\_3](const std::vector<*int*>& *xn*, *int* *y*, *int* *r*) -> *int* {

            return S(step\_up, {I3\_3}, {*xn*[0],*y*,*r*});

        };

        std::vector<*int*> xn = {*x*};

        return R(base\_function, recursive\_function, xn, *y*);

    }

    //оператор умножения на основе базовых

*int* mul(*int* *x*,*int* *y*){

        std::vector<*int*> xn = {*x*};

*auto* base\_function = [this](const std::vector<*int*>& *xn*) -> *int* {

            return this->o();

        };

*auto* I3\_3 = [this](const std::vector<*int*>& *xn*) -> *int* {

            return this->i(*xn*,3);

        };

*auto* I1\_3 = [this](const std::vector<*int*>& *xn*) -> *int* {

            return this->i(*xn*,1);

        };

*auto* sum = [this](const std::vector<*int*>& *xn*) -> *int* { return this->sum(*xn*[0],*xn*[1]);};

*auto* recursive\_function = [this, I1\_3, I3\_3, sum](const std::vector<*int*>& *xn*, *int* *y*, *int* *r*) -> *int* {

            return S(sum, {I1\_3,I3\_3}, {*xn*[0],*y*,*r*});

        };

        return R(base\_function, recursive\_function, xn, *y*);

    }

};

//рекурсивное выполение функции

*int* fun(*int* *x1*,*int* *x2*, *int* *y*){

    PrimitiveReqursion pr;

        return pr.sum(pr.mul(*x1*,*x2*),*y*);

    }

//итеративное выполение функции

*int* iterFun(*int* *x1*,*int* *x2*, *int* *y*){

*int* result = 0;

    for(*int* i=0;i<*x1*;i++){

        result+=1;

    }

    for(*int* i=0;i<*x2*;i++){

        result+=*x1*;

    }

    for(*int* i=0;i<*y*;i++){

        result+=1;

    }

    return result;

}

*int* main(*int* *argc*, *char* \* *argv*[]){

*int* x1 = 0;

*int* x2 = 0;

*int* y = 0;

*int* algo=0;

    std::cout<<"Enter x1: ";

    std::cin>>x1;

    std::cout<<"Enter x2: ";

    std::cin>>x2;

    std::cout<<"Enter y: ";

    std::cin>>y;

    std::cout<<"Enter algo\n 1)Recursive\n 2)Iterative\n";

    std::cin>>algo;

    PrimitiveReqursion pr;

*int* answer = 0;

*auto* start = std::chrono::high\_resolution\_clock::now();

*auto* end = std::chrono::high\_resolution\_clock::now();

    std::chrono::duration<*double*,std::milli> duration;

    switch(algo){

        case 1:

            start = std::chrono::high\_resolution\_clock::now();

            answer = fun(x1,x2,y);

            end = std::chrono::high\_resolution\_clock::now();

            duration = end - start;

            std::cout << "ReqursiveFun x1\*x2+y of " << x1 <<" and " << x2 <<" and " << y << " is: " << answer <<" at " << duration.count()<<"ms"<<std::endl;

        break;

        case 2:

             start = std::chrono::high\_resolution\_clock::now();

             answer = iterFun(x1,x2,y);

             end = std::chrono::high\_resolution\_clock::now();

             duration = end - start;

            std::cout << "IterFun x1\*x2+y of " << x1 <<" and " << x2 <<" and " << y << " is: " << answer <<" at " << duration.count()<<"ms"<<std::endl;

        break;

        default:

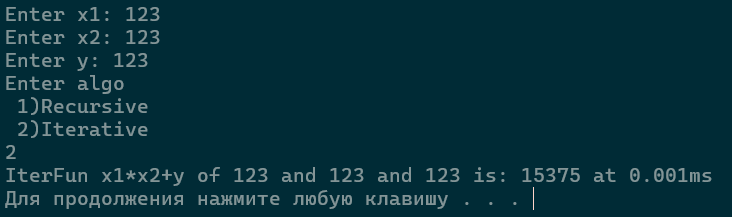
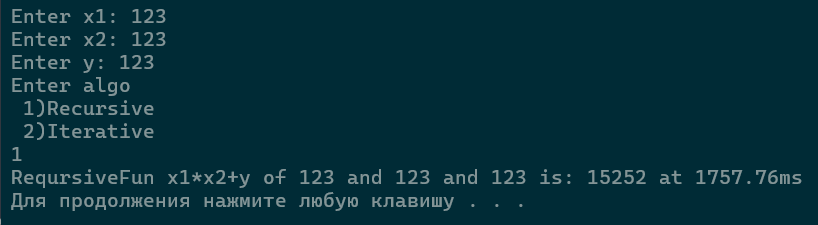
        return -1;

    }

    return 0;

}

**6. Пример результата выполнения**

****

**7. Вывод.**

В ходе данной работы я детально изучил функции и операции рекурсии в теории рекурсивных функций, написал программу для выполнения задания – вычисление формулы с помощью примитивных рекурсивных функций.