**ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»**

**(СПбГУТ)**

**Кафедра безопасности информационных систем**

**ОТЧЁТ**

по практической работе работе №3 на тему:   
**«Понятие рекурсии. Рекурсивные функции»**

по дисциплине «Алгоритмы и структуры данных»

Выполнил: студент группы ИСТ-114,Константинов А.Д.

«19» октября 2022 г. \_\_\_\_\_\_\_\_\_\_\_/А.Д. Константинов/

Принял: к.ф.-м.н., доцент, И.А. Моисеев

«20» октября 2022 г. \_\_\_\_\_\_\_\_\_\_\_/ И.А. Моисеев /

**Содержание:**

[Цель работы: 3](#_Toc114692552)

[Листинг кода: 3](#_Toc114692553)

[Результат работы программы: 8](#_Toc114692554)

[Вывод: 8](#_Toc114692555)

[Приложение: 22](#_Toc114692556)

Цель работы:

# Изучить понятие рекурсии, рекурсивные функции в программировании, приемы по- строения рекурсивной функции при решении задач, научиться применять рекурсивные методы в решении задач на языке С++.

# Листинг кода:

#include <iostream>  
#include <cmath>  
#include <ctime>  
///Функция рекурсивно вычисляет значение n-ного члена подпоследовательности  
///Функция по которой расчитывается значение: F(n) = F(n-1) + F(n-2)  
int task1(int n)  
{  
 if(n <= 0) return 0;  
 if(n == 1) return 1;  
 return task1(n-1) + task1(n-2);  
}  
int counter = 0;  
int task2\_req(int n, int m)  
{  
 counter++;  
 if(m == 0 || n == m) return 1;  
 return task2\_req(n-1,m-1) + task2\_req(n-1,m);  
}  
int task2\_dyn(int n, int m)  
{  
 int M[n+1][m+1];  
 for (int i = 0; i <= n; ++i) {  
 M[i][0] = 1;  
 M[i][i] = 1;  
 for (int j = 0; j < i; ++j) {  
 counter++;  
 M[i][j] = M[i-1][j-1] + M[i-1][j];  
 }  
 }  
 return M[n][m];  
}  
int task2()  
{  
 counter =0;  
 srand(time(0));  
 std::cout << "Task 2: Combinations" << std::endl << "Recursion C(n,m) = " << task2\_req(10,4) << std:: endl;  
 std::cout << "Steps: " << counter << " Time:" << clock()/1000.0 << std::endl;  
 counter =0;  
 srand(time(0));  
 std::cout << "Dynamic C(n,m) = " << task2\_dyn(10,4) << std::endl;  
 std::cout << "Steps: " << counter << " Time:" << clock()/1000.0 << std::endl;  
 std::cout << std::endl;  
}  
  
int task3\_req(int n, int m)  
{  
 counter++;  
 if(n == m) return n;  
 int a = std::min(std::max(n,m) - std::min(n,m),m);  
 int b = std::min(n,m);  
 return task3\_req(a,b);  
}  
int task3\_dyn(int n, int m)  
{  
 while(n!=0 && m!=0)  
 {  
 counter++;  
 if(n>m)  
 {  
 n%=m;  
 }  
 else  
 m%=n;  
 }  
 return m+n;  
}  
int task3()  
{  
 counter =0;  
 srand(time(0));  
 std::cout << "Task 3: NOD" << std::endl << "Recursion NOD = " << task3\_req(500,400) << std:: endl;  
 std::cout << "Steps: " << counter << " Time:" << clock()/1000.0 << std::endl;  
 counter =0;  
 srand(time(0));  
 std::cout << "Dynamic NOD = " << task3\_dyn(500,400) << std::endl;  
 std::cout << "Steps: " << counter << " Time:" << clock()/1000.0 << std::endl;  
 std::cout << std::endl;  
}  
  
  
int task4(int n)  
{  
 int res = 0;  
 while(n>0)  
 {  
 res+= std::pow(n%10,3);  
 n/=10;  
 }  
 if(res == 153) return 1;  
 else return task4(res)+1;  
}  
int task5(int n)  
{  
 if(n%2)  
 {  
 n/=2;  
 }  
 else  
 {  
 n--;  
 }  
 if(n == 0) return 0;  
 else return task5(n) + 1;  
}  
int main() {  
 std::cout << "Task 1: Mystery subsequence" << std::endl << "Let`s count F(6):" << task1(6) << std::endl<< "Done!" << std::endl<< std::endl;  
 task2();  
 task3();  
 std::cout << "Task 4: Find ^3:" << std::endl << "Check path length from number 21: " << task4(21) << std::endl<< "Done!" << std::endl<< std::endl;  
 std::cout << "Task 5: Find min length:" << std::endl << "Check path length to number 100: " << task5(100) << std::endl<< "Done!" << std::endl<< std::endl;  
  
 return 0;  
}

# Результат работы программы:

# /Users/andrew/Projects/Recursion/cmake-build-debug/Recursion

# Task 1: Mystery subsequence

# Let`s count F(6):8

# Done!

# Task 2: Combinations

# Recursion C(n,m) = 210

# Steps: 419 Time:35.522

# Dynamic C(n,m) = 210

# Steps: 45 Time:35.552

# Task 3: NOD

# Recursion NOD = 100

# Steps: 4 Time:35.562

# Dynamic NOD = 100

# Steps: 2 Time:35.567

# Task 4: Find ^3:

# Check path length from number 21: 5

# Done!

# Task 5: Find min length:

# Check path length to number 100: 8

# Done!

# Process finished with exit code 0

# Вывод:

В ходе выполнения практической работы была написана программа, состоящая из 10-ти функций, решающих определенную часть задания. Для решения задания 1 была написана функция task1() вычисляющая значение n-ного члена последовательности по формуле выведенной из примера последовательности F(n) = F(n-1) + F(n-2). Задание решено при помощи рекурсии.

Во втором задании требовалось вычислить число комбинаций по формуле

Задача была решена в рекурcионном и динамическом виде. Для рекурсии потребовалось 419 вызовов функции, а для динамического расчета потребовалось 45 шагов. Что в 10 раз меньше, а значит эффективнее. По времени обе задачи заняли равное количество времени.

Для третьего задания решение рекурсией потребовало 4 шага, а динамический метод решил за 2 шага

Для четвертого задания была написана функция которая считала сумму кубов цифр заданного числа, и если сумма равна 153, то выводила длину пути подсчетов(сколько раз потребовалось вызвать функцию). Для числа 21 этот путь 5 шагов.

Решение 5-й задачи весьма простое – в рекурсии ищем наиболее выгодный вариант действия(умножение на 2 быстрее двигает 1 к 100, нежели +1). Минимальный путь для превращения 1 в 100 – 8 шагов.

Подводя итоги можно утверждать, что рекурсией можно решать многие задачи, но нужно принимать к сведению, что динамический подход предпочтительнее, поскольку требует меньше ресурсов.