Министерство образования и науки РФ

ФГАОУ ВПО «Уральский федеральный университет имени   
первого президента России Б.Н.Ельцина»  
Институт радиоэлектроники и информационных технологий – РТФ  
Департамент информационных технологий и автоматики

Оценка работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Преподаватель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ЧИСЛА K-БОНАЧИ  
Отчет по лабораторной работе №1  
по дисциплине «Методы исследования и моделирования  
информационных процессов и технологий»

Подпись Дата Ф.И.О.

Преподаватель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Добряк П.В.  
Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Тимошенкова Ю.С.

Группа РИМ-161203

Екатеринбург

2017

**Цель работы:** Реализовать алгоритм. Принцип алгоритма Фибоначчи состоит в том, что каждое следующее число состоит из суммы двух предыдущих. Необходимо реализовать алгоритм, который суммирует/перемножает k предыдущих чисел в числовом ряду.

**Задание:**

1. По аналогии с числами Фибоначчи напишите разные реализации:

1.1. С использованием массива размером k;

1.2. С использованием массива размером n прямым переходом;

1.3. С использованием массива размером n обратным переходом;

1.4. Рекурсия

1.5. Мемоизованная рекурсия

2. Вместо суммы предыдущих k-чисел - их произведение или любая другая функция, связывающая два аргумента. Связывающую функцию передавайте как указатель на функцию третьим аргументом.

**Содержание**

[Задание 1 простая реализация алгоритма к-боначи 3](#_Toc475286801)

[1.1 С использованием массива размером k 3](#_Toc475286802)

[1.2 С использованием массива размером n прямым переходом 3](#_Toc475286803)

[1.3 С использованием массива размером n обратным переходом 4](#_Toc475286804)

[1.4 Рекурсия 4](#_Toc475286805)

[1.5 Меморизация 5](#_Toc475286806)

[Задание 2 реализация алгоритма к-боначи с использованием функционала 5](#_Toc475286807)

[Полученные результаты работы для различных алгоритмов 6](#_Toc475286808)

[Выводы 6](#_Toc475286809)

[ПРИЛОЖЕНИЕ A КОД ПРОГРАММЫ 7](#_Toc475286810)

# Задание 1 простая реализация алгоритма к-боначи

## С использованием массива размером k

Код реализации на языке C++

//кабаначи с массивом k

int fibWithKArray(int n, int k)

{

int value = 0;

int \*fib;

fib = new int[k];

if (n <= k)

{

return 1;

}

else

{

for (int j = 1; j <= k; j++)

{

fib[j] = 1;

}

for (int i = k + 1; i <= n; i++)

{

value = 0;

for (int j = 1; j <= k; j++)

{

value = value + fib[j];

fib[j] = fib[j + 1];

}

fib[k] = value;

}

return value;

}

}

## С использованием массива размером n прямым переходом

Код реализации на языке C++

//Прямой Фибонач

int fibPryamoi(int n, int count)

{

int \*fib;

fib = new int[n + 1];

for (int i = 1; i <= n; i++)

{

if (i <= count)

{

fib[i] = 1;

}

else

{

fib[i] = 0;

for (int j = 1; j <= count; j++)

{

fib[i] = fib[i] + fib[i - j];

}

}

}

return fib[n];

}

## С использованием массива размером n обратным переходом

Код реализации на языке C++

//Обратный Фибонач

int fibObratniy(int n, int count)

{

int \*fib;

fib = new int[n + 1 + count];

for (int i = 1; i <= n; i++)

{

fib[i] = 0;

}

fib[1] = 1;

for (int i = 1; i < n; i++)

{

for (int j = 1; j <= count; j++)

{

fib[i + j] = fib[i + j] + fib[i];

if (i + j <= count)

{

fib[i + j] = 1;

}

}

}

return fib[n];

}

## Рекурсия

Код реализации на языке C++

//рекурсия

int fibRecursive(int n, int k)

{

int value = 0;

if (n <= k)

{

return 1;

}

else

{

for (int i = 1; i <= k; i++)

{

value = value + fibRecursive(n - i, k);

}

return value;

}

}

## Меморизация

Код реализации на языке C++

//Мемоизация

int fibMemos(int n, int k)

{

static int fib[1001];

static int osn = 0;

if (osn != k)

{

for (int i = 1; i <= 1001; i++)

{

fib[i] = 0;

}

}

if (fib[n] == 0)

{

fib[n] = fibRecursive(n, k);

}

return fib[n];

}

# Задание 2 реализация алгоритма к-боначи с использованием функционала

Код реализации на языке C++

//Умножить

int mult(int a, int b)

{

return a \* b;

}

//Сложить

int sum(int a, int b)

{

return a + b;

}

//Функциональное Боначи

int fibfunc(int n, int count, int(\*func)(int, int))

{

int \*fib;

fib = new int[n + 1];

for (int i = 1; i <= n; i++)

{

if (i <= count)

{

fib[i] = 1;

}

else

{

fib[i] = fib[i - 1];

for (int j = 1; j < count; j++)

{

fib[i] = func(fib[i], fib[i - j - 1]);

}

}

}

return fib[n];

}

# Полученные результаты работы для различных алгоритмов

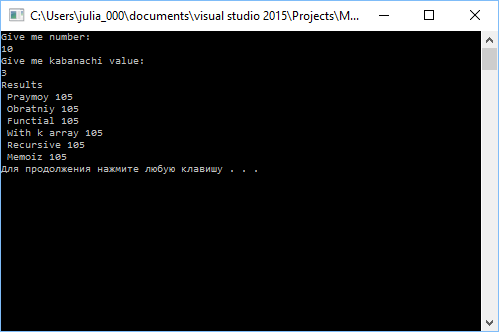


Рисунок 1 – Результат работы программы

# Выводы

В ходе выполнения работы был реализован алгоритм вычисления чисел   
k-боначи различными способами. В результате работы получена программа, которая функционирует правильно. Результат работы представлен на рисунке 1 и в приложении А.

# ПРИЛОЖЕНИЕ A КОД ПРОГРАММЫ

#include "stdafx.h"

#include <stdio.h>

#include <iostream>

using namespace std;

//Прямой Фибонач

int fibPryamoi(int n, int count)

{

int \*fib;

fib = new int[n + 1];

for (int i = 1; i <= n; i++)

{

if (i <= count)

{

fib[i] = 1;

}

else

{

fib[i] = 0;

for (int j = 1; j <= count; j++)

{

fib[i] = fib[i] + fib[i - j];

}

}

}

return fib[n];

}

//Обратный Фибонач

int fibObratniy(int n, int count)

{

int \*fib;

fib = new int[n + 1 + count];

for (int i = 1; i <= n; i++)

{

fib[i] = 0;

}

fib[1] = 1;

for (int i = 1; i < n; i++)

{

for (int j = 1; j <= count; j++)

{

fib[i + j] = fib[i + j] + fib[i];

if (i + j <= count)

{

fib[i + j] = 1;

}

}

}

return fib[n];

}

//Умножить

int mult(int a, int b)

{

return a \* b;

}

//Сложить

int sum(int a, int b)

{

return a + b;

}

//Функциональное Боначи

int fibfunc(int n, int count, int(\*func)(int, int))

{

int \*fib;

fib = new int[n + 1];

for (int i = 1; i <= n; i++)

{

if (i <= count)

{

fib[i] = 1;

}

else

{

fib[i] = fib[i - 1];

for (int j = 1; j < count; j++)

{

fib[i] = func(fib[i], fib[i - j - 1]);

}

}

}

return fib[n];

}

//кабаначи с массивом k

int fibWithKArray(int n, int k)

{

int value = 0;

int \*fib;

fib = new int[k];

if (n <= k)

{

return 1;

}

else

{

for (int j = 1; j <= k; j++)

{

fib[j] = 1;

}

for (int i = k + 1; i <= n; i++)

{

value = 0;

for (int j = 1; j <= k; j++)

{

value = value + fib[j];

fib[j] = fib[j + 1];

}

fib[k] = value;

}

return value;

}

}

//рекурсия

int fibRecursive(int n, int k)

{

int value = 0;

if (n <= k)

{

return 1;

}

else

{

for (int i = 1; i <= k; i++)

{

value = value + fibRecursive(n - i, k);

}

return value;

}

}

//Мемоизация

int fibMemos(int n, int k)

{

static int fib[1001];

static int osn = 0;

if (osn != k)

{

for (int i = 1; i <= 1001; i++)

{

fib[i] = 0;

}

}

if (fib[n] == 0)

{

fib[n] = fibRecursive(n, k);

}

return fib[n];

}

int main()

{

int n;

int k;

cout << "Give me number: \n";

cin >> n;

cout << "Give me kabanachi value: \n";

cin >> k;

cout << "Results \n Praymoy " << fibPryamoi(n, k) << "\n Obratniy ";

cout << fibObratniy(n, k) << "\n Functial ";

cout << fibfunc(n, k, sum) << "\n With k array ";

cout << fibWithKArray(n, k) << "\n Recursive ";

cout << fibRecursive(n, k) << "\n Memoiz ";

cout << fibMemos(n, k) << "\n";

system("pause");

return 0;

}