Министерство образования и науки Российской Федерации   
Федеральное государственное бюджетное образовательное   
учреждение высшего образования   
«Алтайский государственный технический университет им. И.И. Ползунова»

Факультет информационных технологий

Кафедра прикладной математики

Отчёт защищён с оценкой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Преподаватель Астахова Е.В.

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2019 г.

Отчёт

по лабораторной работе № 7

«Эффективность алгоритмов»

по дисциплине «Основы программной инженерии»

Студенты группы ПИ 92:

В.М. Шульпов, И.В. Вильк, А. Н. Гулин, Ю.П. Пирязев

Преподаватель доцент, к. п. н. Астахова Е.В.

Барнаул 2019

**Задание 1**

// runtime.cpp: определяет точку входа для консольного приложения.

#include "stdafx.h"

#include <iostream>

#include <ctime>

using namespace std;

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

{

srand(time(0));

const int array\_size = 200000; // размер массива

int a[array\_size]; // объявление массива

for (int counter = 0; counter < array\_size; counter++)

{

a[counter] = rand() % 50; // или 1, или rand() %50-rand()%50

cout << a[counter] << " "; // печать элементов массива

}

int min = a[0]; // переменная для хранения минимального значения

for (int counter = 1; counter < array\_size; counter++)

{

if ( min > a[counter] ) // поиск минимального значения в массиве

min = a[counter];

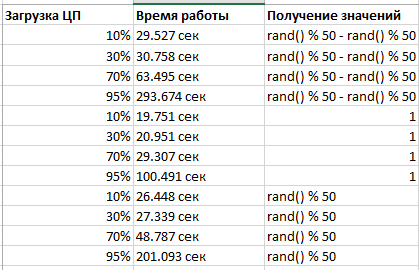
}

cout << "\nmin = " << min << endl;

cout << "runtime = " << clock()/1000.0 << endl; // время работы программы

system("pause"); return 0;

}

**Задание 2 Вариант 2. Наибольший общий делитель (2 с)**

// Программа на 5, ОПИ Лаб.7

#include "stdafx.h"

#include <stdlib.h>

#include <locale.h>

#include <conio.h>

#include <time.h>

#include <Windows.h>

#define N 1000 //размер основного массива

int \_tmain(int argc, \_TCHAR\* argv[])

{

system("color F0"); //белый цвет консоли

setlocale(LC\_ALL, "Rus");

long int nod[N]; //основной массив чисел, для которых ищем НОД

double prtct; //prtct используется для защиты ввода от символов, кроме цифр

int n; //n- кол-во чисел, вводится пользователем

do{ //зацикливание программы

printf("Программа находит НОД для введённых чисел.\n");

do{

printf("Введите кол-во чисел n (n<=1000): ");

prtct=scanf\_s("%d", &n); //ввод n и хранение результата scanf в отдельной переменной

fflush(stdin); //очистить выходной буфер, чтобы программа работала правильно при вводе чего-то неподходящего

} while ( n<1 || n>1000 || prtct!=1 ); //защита от ввода отрицательного или слишком большого n или символов вместо числа

printf("Введите %d ваших чисел:\n", n);

for(int i=0; i<n; i++) //ввод чисел в массив поэлементно

do{

prtct=scanf("%ld", &nod[i]); //ввод данного числа и хранение результата scanf в отдельной переменной

fflush(stdin); //очистить выходной буфер, чтобы программа работала правильно при вводе чего-то неподходящего

} while (prtct!=1 && nod[i]<=0 ); //защита от ввода символов вместо числа

unsigned int start\_time = clock(); // начальное время работы программы

for(int i=0; i<n-1; i++) //перебираем все введённые числа

{

while(nod[i]!=0 && nod[i+1]!=0)//находим НОД для двух чисел (данного числа и следующего за ним) с помощью алгоритма Евклида

{

if(nod[i]>nod[i+1])

nod[i]=nod[i]-nod[i+1];

else nod[i+1]=nod[i+1]-nod[i];

}

/\*по итогу алгоритма записать в следующее за данным число итоговый НОД для этих чисел, затем именно это число будет использоваться в алгоритме со следующим\*/

if(nod[i]!=0)

nod[i+1]=nod[i];

else nod[i+1]=nod[i+1];

}

printf("НОД для введённых чисел: %d\n", nod[n-1]);

unsigned int end\_time = clock(); // конечное время

unsigned int search\_time = end\_time - start\_time; // искомое время

printf("Время в милисекундах затраченное на программу: %d\n",search\_time);

printf("Продолжить? (Да - любая клавиша, нет - Esc)\n \n");

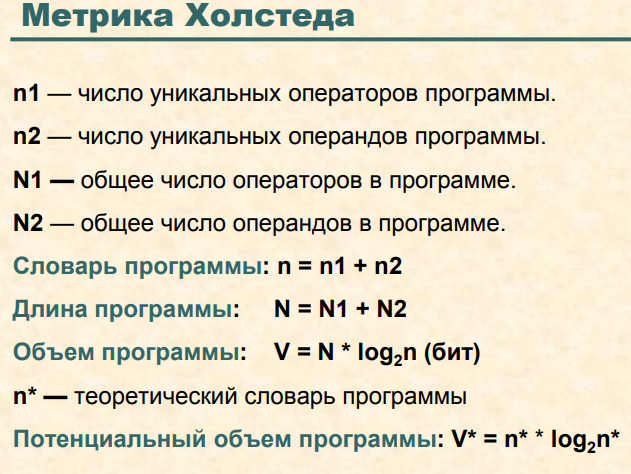
} while (\_getch()!=27); //запустить программу заново, если нажато что угодно, кроме esc

return 0;

}

****

Трудоемкость – квадратичная

**n1 = 10**

**n2 = 12**

**N1 = 43**

**N2 = 44**

**n = n1 + n2 = 22**

**N = N1 + N2 = 87**

**V =87\*4,5=392 бит**

**V\*=1022\*10= 10220**

**cl = CL / N1 = 4/43 =0,093**

**F = Nком/Nстр = 21/46=0,45 > 0.1**

Элементы защитного программирования:   
Минимизация области действия переменных, защита ввода.

**Задание 3 Вариант 1. Числа Фибоначчи (2 с)**

// exeercise3\_Fibonachi.cpp

//Программа выводит на экран только числа Фибоначчи из последовательности чисел

#include "stdafx.h"

#include <stdio.h>

#include <locale.h>

#include <ctime>

#include <windows.h>

#define MAX 2000000000 //максимальное вводимое число 2 млрд

#define NUM 10000 //максимальное количество чисел в последовательности

#define NUM\_BIG\_FIB 32 //количество чисел Фиббоначи, которые меньше 2 млрд

int \_tmain(int argc, \_TCHAR\* argv[])

{

srand(time(0));

setlocale(LC\_ALL, "Rus");

system("color F0");

int list[NUM], //список чисел в последовательности

n, //количество чисел в последовательности

fib[NUM\_BIG\_FIB], //последовательность чисел Фибоначчи

f=1; //флаг на случай, когда чисел Ф. нет

//заполнение массива числами Фибоначчи

fib[0] = 0;

fib[1] = 1;

for (int i = 2; i < NUM\_BIG\_FIB; i++) fib[i] = fib[i - 1] + fib[i - 2];

//вывод чисел Фибоначчи

printf("Числа Фибоначчи (до числа %d): ", MAX);

for (int i = 0; i < NUM\_BIG\_FIB; i++) printf("%d ", fib[i]);

puts("");

//ввод количества чисел в последовательности

do {

printf("Введите количество чисел в последовательности: ");

scanf("%d", &n);

fflush(stdin);

} while (n<1 || n>NUM);

//ввод чисел последовательности

for (int i = 0; i < n; i++)

{

do {

printf("Введите %d-е число: ", i + 1);

scanf("%d", &list[i]);

fflush(stdin);

} while (list[i]>MAX);

}

int start\_time = clock();

//вывод чисел последовательности

printf("Числа последовательности: ");

for (int i = 0; i < n; i++) printf("%d ", list[i]);

puts("");

//вывод только чисел Фиббоначи последовательности

printf("Числа Фибоначчи: ");

for (int i = 0; i < n; i++)

for (int j = 0; j < NUM\_BIG\_FIB; j++)

if (list[i] == fib[j])

{

printf("%d ", list[i]);

f = 0;

break;

}

if (f) printf("в вашей последовательности нет чисел Фибоначчи");

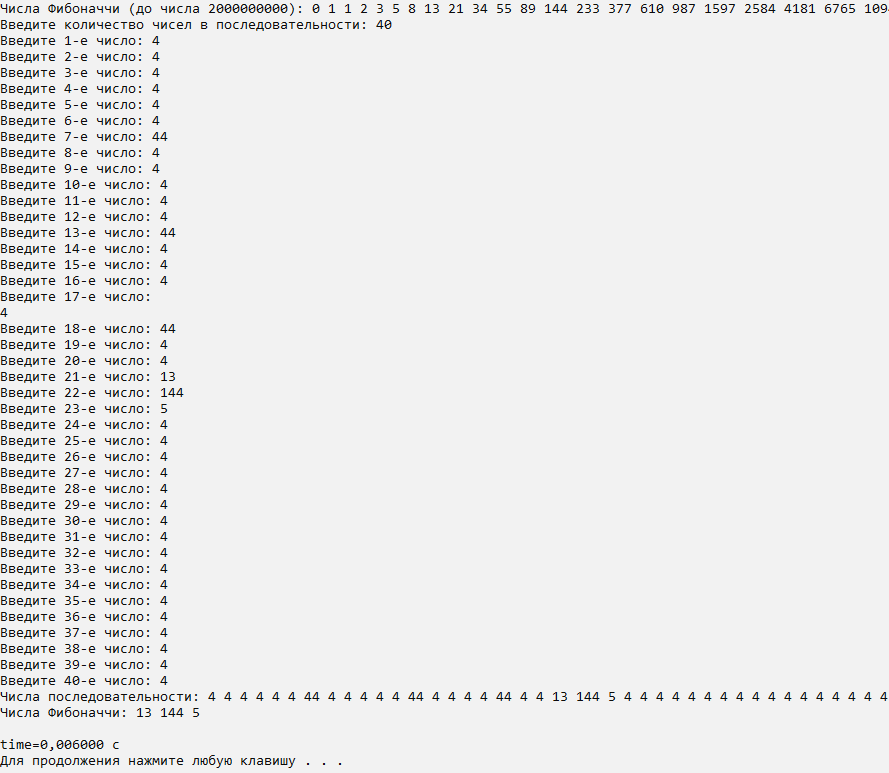
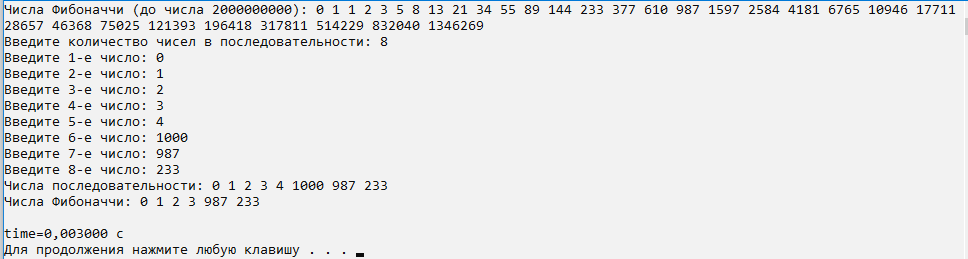
puts("");

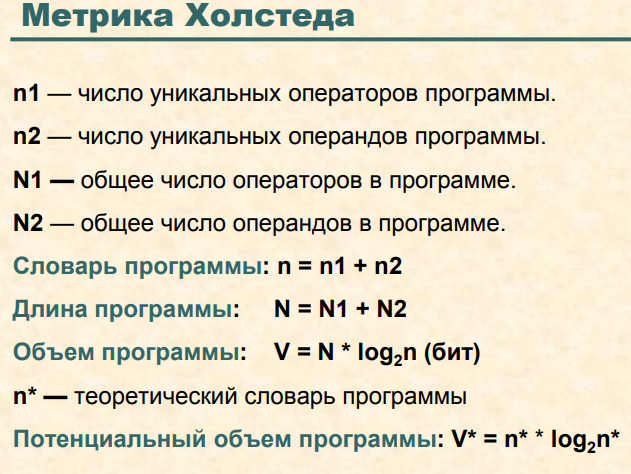
int end\_time = clock();

double search\_time = end\_time - start\_time;

printf("\ntime=%f с\n", search\_time / 1000.);

return 0; }



**n1 = 8**

**n2 = 16**

**N1 = 37**

**N2 = 67**

**n = n1 + n2 = 24**

**N = N1 + N2 = 104**

**V =104\*4.6=479 бит**

**V\*=1023\*9=9171**

**cl = CL / N = 2/37 =0,05**

**F = Nком/Nстр = 15/77=0,19 > 0.1**

элементы защитного программирования: Минимизация области действия переменных

**Задача на «4» Частота встречаемости букв (2 с)**

// na4\_repeats.cpp:

// программа определяет, какие буквы встречаются в тексте наиболее часто

#include "stdafx.h"

#include <stdio.h>

#include <locale.h>

#include <conio.h>

#include <windows.h>

#include <ctime>

#include <string.h>

#define MIN\_NUM\_WORD 97 //10чный код символа "a" в ASKII

#define MAX\_NUM\_WORD 122 //10чный код символа "z" в ASKII

#define LEN 1000

int main() {

srand(time(0));

setlocale(LC\_ALL, "Rus");

system("color F0");

char str[LEN]; // строка

char ch[LEN]; // массив разных символов строки

int num\_ch[LEN];// количество разных символов строки (например, в num\_ch[1] содержится кол-во элементов ch[1])

int len; // длина строки

int k = 0;// счётчик

int f = 1; // флаг

char tmp\_ch; // вспомогательная для сортировки массива символов (ch)

int tmp\_n; // вспомогательная для сортировки массива чисел (num\_ch)

int n; // кол-во частовстречающихся элементов (для вопроса пользователю)

printf("Введите строку: ");

gets(str);

int start\_time = clock();

len = strlen(str);

for (int i = 0; i < len; i++) printf("%c", str[i]);

ch[len+1] = '\n';

// заполнение массива неповторяющимися символами

for (int i = 0; i < len; i++) {

for (j = 0; ch[j] != '\n'; j++) {

/\*если символ - пробел или такой уже записан\*/

if (str[i] == ' ' || str[i] == ch[j]) {

f = 0;

break;

}

}

/\*если символ не пробел и ещё не записан, записать его\*/

if (f) {

ch[k] = str[i];

k++;

}

else f = 1;

}

// посчёт кол-ва каждого символа в тексте

for (int i = 0; i < k; i++) num\_ch[i] = 0; /\*инициализация нулями\*/

for (int i = 0; i < len; i++)

for (int j = 0; j < k; j++)

if (str[i] == ch[j]) {

num\_ch[j]++;

break;

}

// сортировка по убыванию массива num\_ch (одновременно с ним сортируем массив ch)

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

for (int j = 0; j < k-i; j++)

if (num\_ch[j] < num\_ch[j + 1]) {

/\*сортировка массива с числами\*/

tmp\_n = num\_ch[j];

num\_ch[j] = num\_ch[j + 1];

num\_ch[j + 1] = tmp\_n;

/\*сортировка массива со знаками\*/

tmp\_ch = ch[j];

ch[j] = ch[j + 1];

ch[j + 1] = tmp\_ch;

}

// вывод n частовстречающихся элементов

for (int i = 0; i < 2; i++) printf("%c(%d)\n", ch[i], num\_ch[i]);

printf("\nколичество уникальных=%d\n", k);

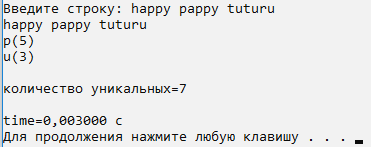
int end\_time = clock();

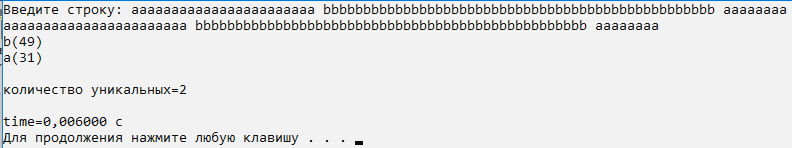
double search\_time = end\_time - start\_time;

printf("\ntime=%f с\n", search\_time/1000.);

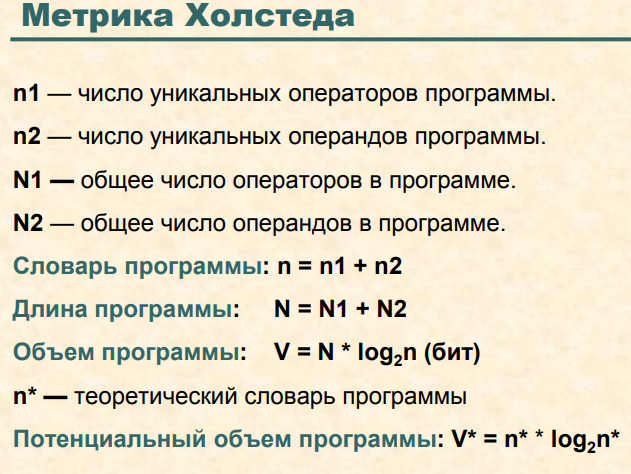
return 0;

}





Трудоемкость – квадратичная

**n1 = 8**

**n2 = 18**

**N1 = 53**

**N2 = 114**

**n = n1 + n2 = 26**

**N = N1 + N2 = 167**

**V = 167\*4,7=784 бит**

**V\*=1026\*9=9171**

****

**cl = CL / N = 4/53= 0,075**

 **F = Nком/Nстр = 22/98=0,224 > 0.1**

элементы защитного программирования: Минимизация области действия переменных

**Задача на «5» Скобочная последовательность (2 с)**

// Программа для определения правильность скобочной последовательности

#include "stdafx.h"

#define N 1000

int \_tmain(int argc, \_TCHAR\* argv[])

{

int open,close, len;

char a[N];

setlocale(LC\_ALL, "RUS");

system("color F0");

do

{

open = 0;// Переменная для открытых скобок

close = 0;// Переменная для закрытых скобок

printf("Программа определяет правильность скобочной последовательности, введите скобки(<%d), учитываются только круглые скобки\n",N);

fgets(a, N, stdin);

unsigned int start\_time = clock(); // начальное время

len = strlen(a); // Переменная для длины массива со скобками

for(int i = 0; i<len; i++)// Посимвольный перебор в поиске открытых и закрытых скобок с их учётом в счётчике

{

if(a[i] == '(' ) open = open + 1;

if(a[i] == ')' ) close = close + 1;

}

if(open==close) printf("YES!\n");

else printf("NO!\n");

unsigned int end\_time = clock(); // конечное время

unsigned int search\_time = end\_time - start\_time; // искомое время

printf("Время в милисекундах затраченное на программу: %d\n",search\_time);

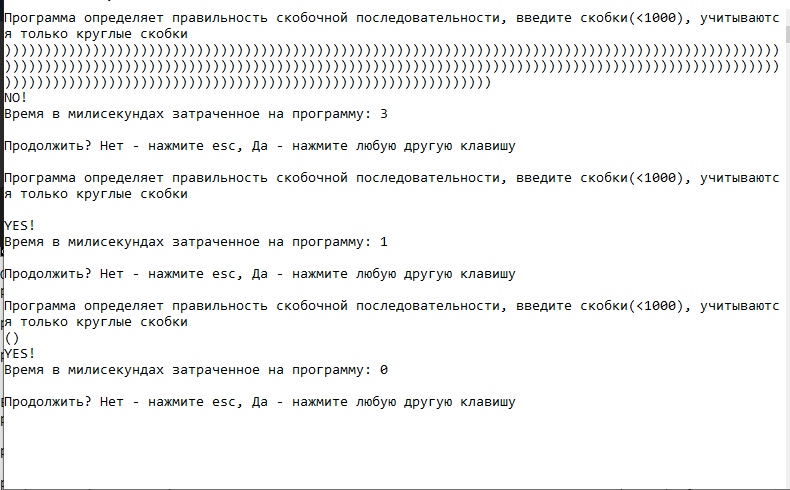
printf("\nПродолжить? Нет - нажмите esc, Да - нажмите любую другую клавишу\n\n");

}

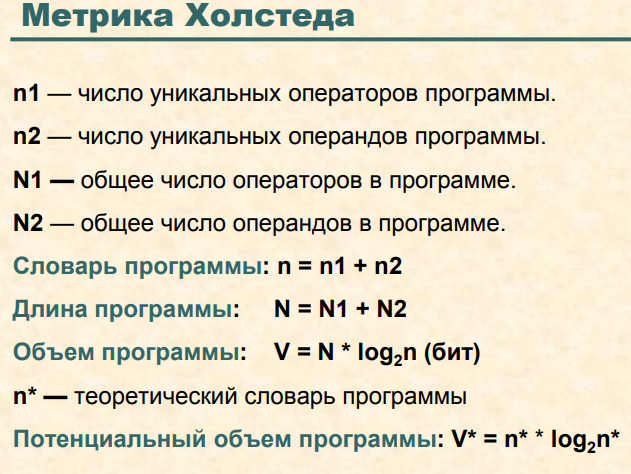
while(getch()!=27);

return 0;

}



Трудоемкость – квадратичная



**n1 = 6**

**n2 = 13**

**N1 = 18**

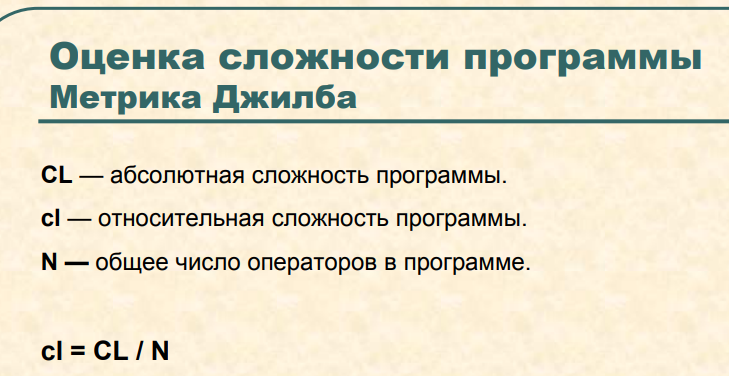
**N2 = 28**

**n = n1 + n2 = 19**

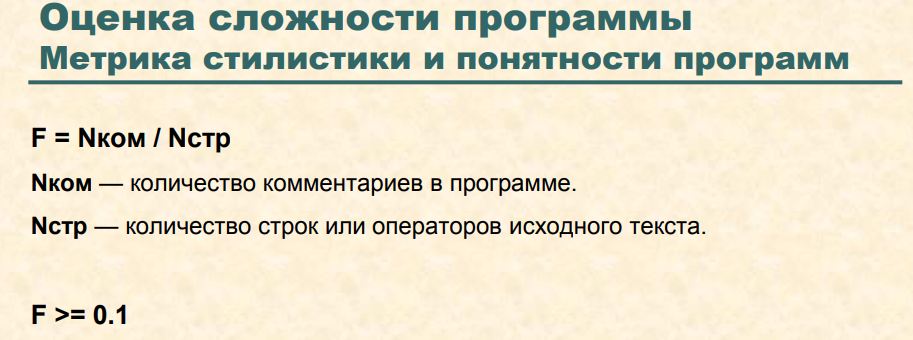
**N = N1 + N2 = 46**

**V = 46\*4=184**

**V\*=1019\*9=9171**



**cl = CL / N = 4/18= 2/9**



**F = Nком/Nстр = 8/32=0,25  
F > 0,1**

элементы защитного программирования: Минимизация области действия переменных