1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский государственный политехнический университет
3. —
4. Институт кибербезопасности и защиты информации
5. **ЛАБОРАТОРНАЯ РАБОТА №2**
6. **Числовой ребус**  
   по дисциплине «Языки программирования»
7. Выполнил
8. студент гр. 4851004/90002 С.В. Логачева

Я.Э. Янбарисова

1. <*подпись*>
2. Руководитель
3. acc. Малышев Е. В.
4. <*подпись*>
5. Санкт-Петербург
6. 2020

**ОГЛАВЛЕНИЕ**

[1 Формулировка задания 3](#_Toc63181805)

[2 Результаты работы 4](#_Toc63181806)

[2.1 Исходная программа 4](#_Toc63181807)

[2.2 Оптимизация программы 4](#_Toc63181808)

[2.3 Сравнение программ 5](#_Toc63181809)

[3 ВЫВОДЫ ПО РАБОТЕ 6](#_Toc63181810)

[Приложение А 7](#_Toc63181811)

[Приложение Б 12](#_Toc63181812)

# Формулировка задания

Цель работы -– реализовать программу для решения числовых ребусов, оптимизировать предложенное решение, после чего проанализировать влияние предложенных оптимизаций на быстродействие программы.

Задачи:

1. Реализовать программу, решающую числовой ребус
2. Оптимизировать программу
3. Сделать вывод о влиянии оптимизации за скорость работы

# Результаты работы

## Исходная программа

Разработана программа, отвечающая требованиям лабораторной работы. Листинг программы в Приложении А.

Идея начального алгоритма:

1. Входная строка разбивается на слова, все данные записываются в специальную структуру
2. Происходит поиск используемых букв
3. Запускается подбор чисел: генерируется значения, заменяются ими буквы и считает итоговое значение

Алгоритм плох тем, что нет проверок на более ранний выход, лишние проходы циклов, сложная функция подбора с 10 циклами. В худшем случае сложность программы будет равна примерно 10!, так как перебор цифр зависит от количества слов, их длины, проверки правильности примера.

## **Оптимизация программы**

Код оптимизированной программы представлен в приложении Б.

Изменения и оптимизации:

1. Происходит подсчёт используемых букв, который в дальнейшем используется как алфавит, что позволяет убрать огромные циклы.
2. В функции подбора были заменены циклы на рекурсивные, что позволило упростить программу.
3. Добавлена функция проверки, которая запускает подсчет значений. Сопоставляют алфавиту букв их цифровые значения, осуществляются проверки подсчёта суммы последних цифр и старта числа с 0.

Также в программе выли вынесены всевозможные инварианты, оптимизированы циклы, проверки, включена оптимизация на уровне компилятора О2.

Итоговую сложность удалось снизить до N!, так как было сокрушено количество итераций до 10!/(10-N)!. И теперь алгоритм больше зависит от количества различных используемых букв.

## Сравнение программ

Сравним начальную и оптимизированную программы.

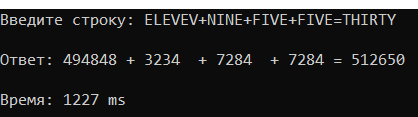
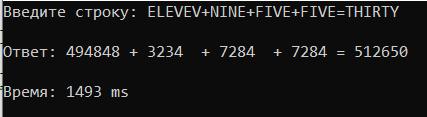


Рисунок 1 - работа начальной и оптимизированной программы

|  |  |  |
| --- | --- | --- |
| Тест | Время выполнения, миллисекунд | |
| Начальная | Оптимизированная |
| SEND+MORE=MONEY | 1461 | 632 |
| BE+BE=MOO | 927 | 1 |
| BIG+CAT=LION | 94 | 147 |
| ELEVEN+NINE+FIVE+ +FIVE=THIRTY | 1496 | 1253 |

Таблица 1 - Результаты сравнения программ.

Можно сделать вывод, что оптимизация дала хорошие результаты для многих тестов. В этом помогли использование рекурсии и использование проверок, чтобы не производить лишний подбор. Время работы зависит действительно от количества используемых букв, так что настоящая сложность оптимизированной программы стремится именно к N!.

# ВЫВОДЫ ПО РАБОТЕ

В результате данной работы были освоены базовые принципы оптимизации программ. Результаты данной работы показали, что далеко не всегда самый очевидный вариант оптимизации – самый оптимальный. Были отработаны навыки улучшения и оптимизации имеющегося алгоритма, проведено множество различных тестов. Очень важно попробовать различные подходы и среди них выбрать лучший.

Исходный алгоритм работает с подбором через 10 вложенных циклов for(). Оптимизированный работает алгоритм так же. Всяческие проверки позволили не перебирать варианты неадекватной подстановки. Поэтому использован оптимизированный и лучший из вариантов кода.

Бесспорно, оптимизацию кода необходима использовать, так как она позволяет добиться двузначного прироста производительности, однако, важно помнить, что самая эффективная оптимизация – это оптимизация алгоритма, так как компилятор чаще всего способен выполнять только базовые оптимизации, что данная работа демонстрирует, так как рекурсивная и нерекурсивная оптимизации отличаются только алгоритмом, но имеют двузначную разницу в скорости работы.

Приложение А

Начальная программа

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <time.h>

#include <locale.h>

void used\_letters();

int check\_numbers\_on\_letters();

int calc();

void replace\_on\_number(int\* Mas);

void selection();

struct about\_rebus {

char mass\_word[5][9];

int mass\_w\_int[5][9];

int word\_count;

int len[8];

int row = 0;

int column = 0;

};

char eng\_letters[] = { 'A', 'B', 'C' , 'D' , 'E' , 'F' , 'G' , 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S' , 'T' , 'U' , 'V' , 'W' , 'X', 'Y', 'Z' };

int eng\_letter\_int[] = { -1, -1, -1 , -1 , -1, -1 ,-1 , -1, -1, -1, -1, - 1, -1, -1, -1, -1, -1, -1, -1, -1 , -1 ,-1 , -1 , -1, -1, -1 };

unsigned int timee;

struct about\_rebus mass\_reb;

int main()

{

setlocale(LC\_ALL, "Rus");

int i = 0;

char str[100];

printf("Введите строку: ");

fgets(str, 100, stdin);

while (str[i-1] != '=')

{

mass\_reb.column = 0;

while (str[i] != '+' && str[i] != '=') {

mass\_reb.mass\_word[mass\_reb.row][mass\_reb.column] = str[i];

i++;

mass\_reb.column++;

}

mass\_reb.mass\_word[mass\_reb.row][mass\_reb.column] = '\0';

mass\_reb.len[mass\_reb.row] = mass\_reb.column;

mass\_reb.row++;

i++;

}

mass\_reb.column = 0;

while (str[i] != '\n') {

mass\_reb.mass\_word[mass\_reb.row][mass\_reb.column] = str[i];

i++;

mass\_reb.column++;

}

mass\_reb.mass\_word[mass\_reb.row][mass\_reb.column] = '\0';

mass\_reb.len[mass\_reb.row] = mass\_reb.column;

mass\_reb.word\_count = mass\_reb.row+1;

timee = clock();

selection();

}

void selection()

{

used\_letters();

int mass\_number[10];

int i0, i1, i2, i3, i4, i5, i6, i7, i8, i9;

for (int i0 = 0; i0 < 10; i0++)

{

mass\_number[0] = i0;

for (i1 = 0; i1 < 10; i1++)

{

if (i1 != i0)

{

mass\_number[1] = i1;

for (i2 = 0; i2 < 10; i2++)

{

if (i2 != i0 && i2 != i1)

{

mass\_number[2] = i2;

for (i3 = 0; i3 < 10; i3++)

{

if (i3 != i0 && i3 != i1 && i3 != i2)

{

mass\_number[3] = i3;

for (i4 = 0; i4 < 10; i4++)

{

if (i4 != i0 && i4 != i1 && i4 != i2 && i4 != i3)

{

mass\_number[4] = i4;

for (i5 = 0; i5 < 10; i5++)

{

if (i5 != i0 && i5 != i1 && i5 != i2 && i5 != i3 && i5 != i4)

{

mass\_number[5] = i5;

for (i6 = 0; i6 < 10; i6++)

{

if (i6 < 10 && i6 != i0 && i6 != i1 && i6 != i2 && i6 != i3 && i6 != i4 && i6 != i5)

{

mass\_number[6] = i6;

for (i7 = 0; i7 < 10; i7++)

{

if (i7 != i0 && i7 != i1 && i7 != i2 && i7 != i3 && i7 != i4 && i7 != i5 && i7 != i6)

{

mass\_number[7] = i7;

for (i8 = 0; i8 < 10; i8++)

{

if (i8 != i0 && i8 != i1 && i8 != i2 && i8 != i3 && i8 != i4 && i8 != i5 && i8 != i6 && i8 != i7)

{

mass\_number[8] = i8;

for (i9 = 0; i9 < 10; i9++)

{

if (i9 != i0 && i9 != i1 && i9 != i2 && i9 != i3 && i9 != i4 && i9 != i5 && i9 != i6 && i9 != i7 && i9 != i8)

{

mass\_number[9] = i9;

replace\_on\_number(mass\_number);

if (check\_numbers\_on\_letters())

{

if (calc())

{

return;

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

return;

}

void used\_letters()

{

int i, j, k;

for (i = 0; i < 26; i++)

{

for (j = 0; j < mass\_reb.word\_count; j++)

{

for (k = 0; k < mass\_reb.len[j]; k++)

{

if (mass\_reb.mass\_word[j][k] == eng\_letters[i])

{

eng\_letter\_int[i] = -2;

mass\_reb.mass\_w\_int[j][k] = eng\_letter\_int[i];

}

}

}

}

}

int check\_numbers\_on\_letters()

{

int i, j, k;

for (i = 0; i < 26; i++)

{

if (eng\_letter\_int[i] != -1)

{

for (j = 0; j < mass\_reb.word\_count; j++)

{

for (k = 0; k < mass\_reb.len[j]; k++)

{

if (mass\_reb.mass\_word[j][k] == eng\_letters[i])

{

mass\_reb.mass\_w\_int[j][k] = eng\_letter\_int[i];

}

}

}

}

}

return 1;

}

int calc()

{

int mass\_sum\_word[5];

int final\_sum = 0;

int j, i = 0;

int ten;

int summa;

for (i = 0; i < mass\_reb.word\_count; i++)

{

ten = 1;

summa = 0;

for (j = mass\_reb.len[i] - 1; j >= 0; j--) {

summa += mass\_reb.mass\_w\_int[i][j] \* ten;

ten \*= 10;

}

mass\_sum\_word[i] = summa;

final\_sum += mass\_sum\_word[i];

}

final\_sum -= mass\_sum\_word[i - 1];

if (final\_sum == mass\_sum\_word[i - 1]) {

printf("\nОтвет: %d", mass\_sum\_word[0]);

for (j = 1; j < mass\_reb.word\_count - 1; j++)

{

printf(" + %d ", mass\_sum\_word[j]);

}

printf("= %d \n", mass\_sum\_word[mass\_reb.word\_count - 1]);

printf("\nВремя: %u ms\n\n", (clock() - timee));

return 1;

}

else {

return 0;

}

}

void replace\_on\_number(int\* mass\_number)

{

int i, j = 0;

for (i = 0; i < 26; i++)

{

if (eng\_letter\_int[i] != -1)

{

eng\_letter\_int[i] = mass\_number[j];

j++;

}

}

}

# Приложение Б

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <time.h>

#include <locale.h>

int calc();

int replace\_on\_number(int\* Mas);

void selection(int\* Mas, int step = 0);

void checking\_exit(int CHIS, int\* Mas);

void used\_letter(char a);

struct about\_table {

char mass\_word[5][9];

int mass\_w\_int[5][9];

int word\_count;

int len[8];

int row = 0;

int column = 0;

int count\_letter = 0;

};

char eng\_letters[26];

int eng\_letter\_int[26];

unsigned int timee;

int check\_decisions = 0;

struct about\_table mass\_reb;

int main()

{

setlocale(LC\_ALL, "Rus");

int i = 0;

int mass\_numbers[10];

char str[100];

printf("Введите строку: ");

fgets(str, 100, stdin);

while (str[i - 1] != '=')

{

mass\_reb.column = 0;

while (str[i] != '+' && str[i] != '=') {

used\_letter(str[i]);

mass\_reb.mass\_word[mass\_reb.row][mass\_reb.column] = str[i];

i++;

mass\_reb.column++;

}

mass\_reb.mass\_word[mass\_reb.row][mass\_reb.column] = '\0';

mass\_reb.len[mass\_reb.row] = mass\_reb.column;

mass\_reb.row++;

i++;

}

mass\_reb.column = 0;

while (str[i] != '\n') {

used\_letter(str[i]);

mass\_reb.mass\_word[mass\_reb.row][mass\_reb.column] = str[i];

i++;

mass\_reb.column++;

}

mass\_reb.mass\_word[mass\_reb.row][mass\_reb.column] = '\0';

mass\_reb.len[mass\_reb.row] = mass\_reb.column;

mass\_reb.word\_count = mass\_reb.row + 1;

if (mass\_reb.count\_letter > 10)

{

printf("\nОшибка в выражении. Много букв.\n");

exit(0);

}

timee = clock();

for (i = 0; i <= 9; i++)

{

mass\_numbers[i] = i;

}

selection(mass\_numbers);

if (check\_decisions == 0)

{

printf("\nНет решения.\n");

}

}

int calc()

{

int mass\_sum\_word[5];

int final\_sum = 0;

int j, i = 0;

int ten = 1;

int summa = 0;

for (i = 0; i < mass\_reb.word\_count; i++)

{

ten = 1;

summa = 0;

for (j = mass\_reb.len[i] - 1; j >= 0; j--) {

summa += mass\_reb.mass\_w\_int[i][j] \* ten;

ten \*= 10;

}

mass\_sum\_word[i] = summa;

final\_sum += mass\_sum\_word[i];

}

final\_sum -= mass\_sum\_word[i - 1];

if (final\_sum == mass\_sum\_word[i - 1]) {

printf("\nОтвет: %d", mass\_sum\_word[0]);

for (j = 1; j < mass\_reb.word\_count - 1; j++)

{

printf(" + %d ", mass\_sum\_word[j]);

}

printf("= %d \n", mass\_sum\_word[mass\_reb.word\_count - 1]);

printf("\nВремя: %u ms\n\n", (clock() - timee));

check\_decisions = 1;

return 1;

}

else {

return 0;

}

}

int replace\_on\_number(int\* mass\_number)

{

int i, j = 0, ii, jj;

for (i = 0; i < mass\_reb.count\_letter; i++)

{

eng\_letter\_int[i] = mass\_number[j];

j++;

for (ii = 0; ii < mass\_reb.word\_count; ii++)

{

for (jj = 0; jj < mass\_reb.len[ii]; jj++)

{

if (mass\_reb.mass\_word[ii][jj] == eng\_letters[i])

{

mass\_reb.mass\_w\_int[ii][jj] = eng\_letter\_int[i];

}

}

}

}

//0

int sum = 0, num = 0, k;

for (k = 0; k < mass\_reb.word\_count; k++)

{

if (mass\_reb.mass\_w\_int[k][0] == 0)

{

return 0;

}

num = mass\_reb.mass\_w\_int[k][mass\_reb.len[k] - 1];

sum += num;

}

//1 и послед

sum -= num;

if (sum > 9)

{

sum %= 10;

}

if (sum != num) {

return 0;

}

return 1;

}

void checking\_exit(int step, int\* mass\_number) {

if (step == mass\_reb.count\_letter)

{

if (replace\_on\_number(mass\_number))

{

if (calc())

{

exit(0);

}

}

}

}

void selection(int\* mass\_number, int step)

{

int i, a;

if (step >= mass\_reb.count\_letter)

{

checking\_exit(step, mass\_number);

return;

}

for (i = step; i < 10; ++i)

{

a = mass\_number[i];

mass\_number[i] = mass\_number[step];

mass\_number[step] = a;

selection(mass\_number, step + 1);

a = mass\_number[i];

mass\_number[i] = mass\_number[step];

mass\_number[step] = a;

}

return;

}

void used\_letter(char letter)

{

for (int i = 0; i < mass\_reb.count\_letter; i++)

{

if (letter == eng\_letters[i])

{

return;

}

}

eng\_letters[mass\_reb.count\_letter] = letter;

eng\_letter\_int[mass\_reb.count\_letter] = -2;

mass\_reb.count\_letter++;

}