ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

**БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ**

**ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

(НИУ «БелГУ»)

ИНСТИТУТ ИНЖЕНЕРНЫХ И ЦИФРОВЫХ ТЕХНОЛОГИЙ

КАФЕДРА МАТЕМАТИЧЕСКОГО И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

**Отчет по лабораторной работе 1**

**по дисциплине: «Теория автоматов»**

студента очного отделения

4 курса 12001801 группы

Капустина Виктора Сергеевича

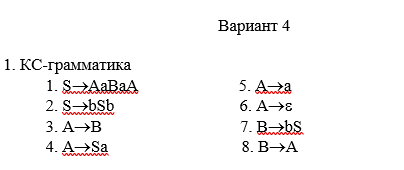
Проверил(а):

Великая Яна Геннадьевна

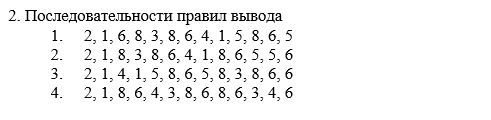
Белгород 2021

**Вариант 4**

Задание 1. Написать программу, выполняющую левый (правый) вывод в заданной КС-грамматике.

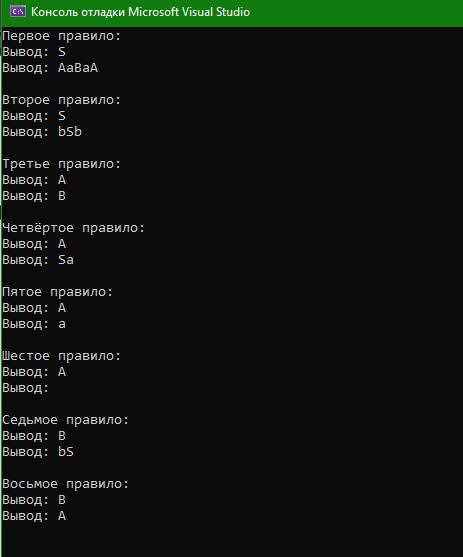


Задание 2. Написать программу, выполняющую последовательность правил вывода по КС-грамматике.



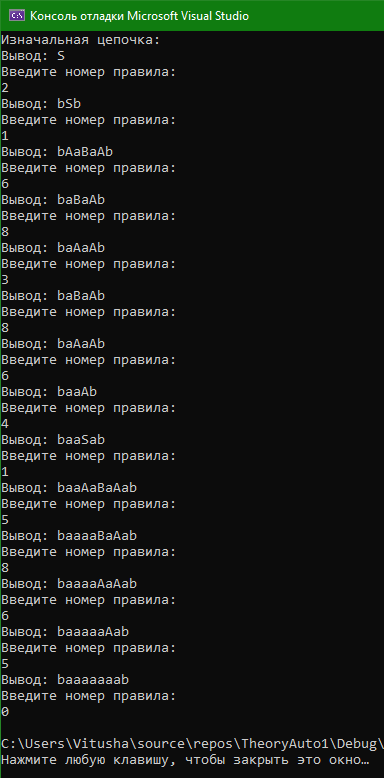
**Задание 1**

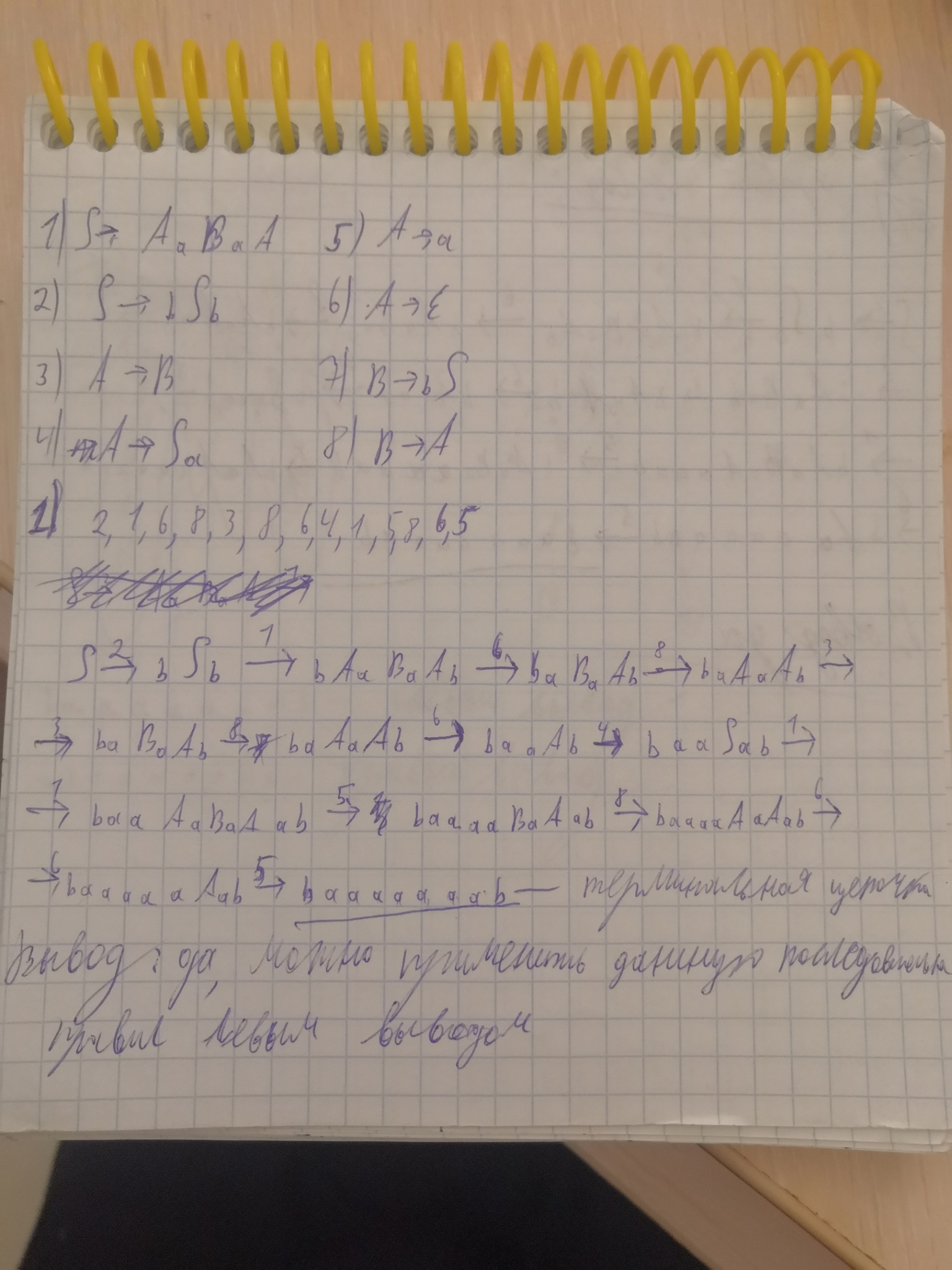
Программе задавалось каждое правило отдельно и выполняется левый вывод заданной КС-грамматике

  
Рис. 1 Результат работы программы

**Задание 2**

Программе задавалось номер каждого правила, и на выходе получаем промежуточную цепочку. Вводим последовательность по заданию

  
Рис. 2 Результат работы программы

  
Рис. 2.1 Сравнение результатов

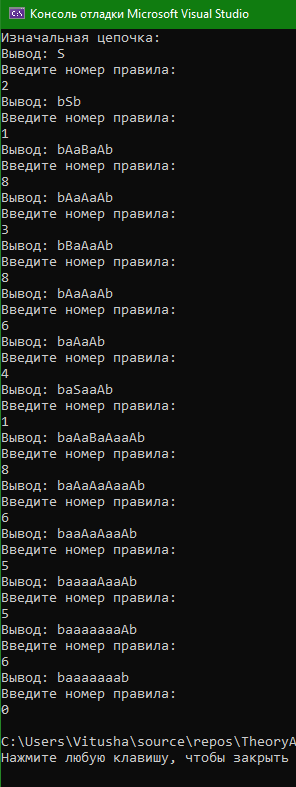


Рис. 3. Результат работы программы

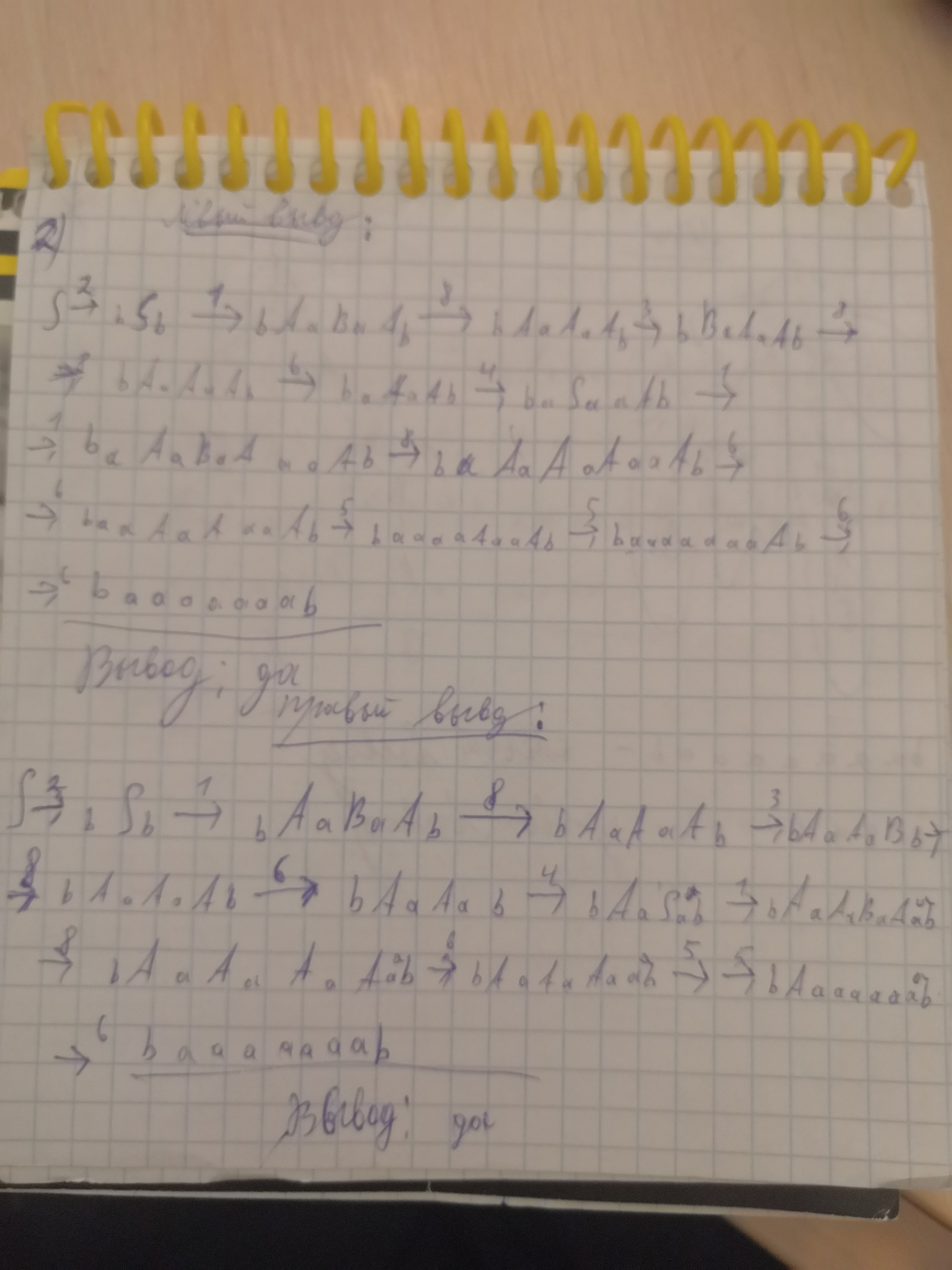
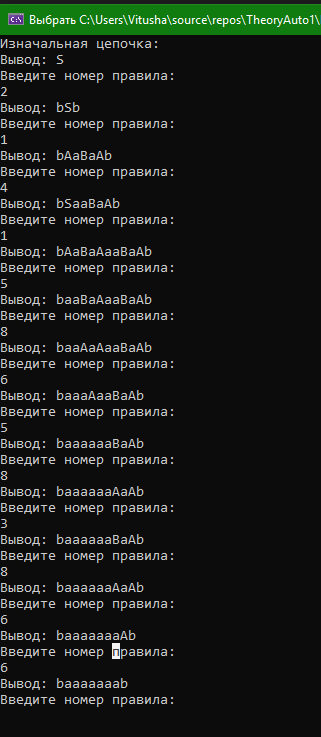
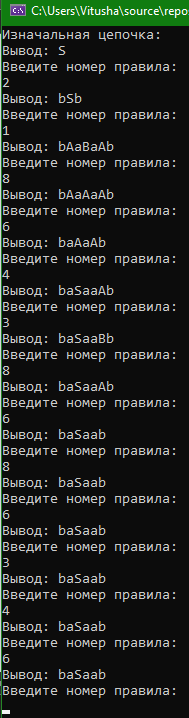


Рис. 3.1 Сравнение результатов

  
Рис. 4 Результат работы программы

  
Рис. 5 Результат работы программы

**Листинг программы 1**

#include <iostream>

#include <string>

using namespace std;

class MyClass

{

public:

string Str = "S";

void Define\_Str(string str)

{

Str = str;

}

void Out\_Str()

{

cout<<"Вывод: "<< Str<< endl;

}

string FirstRule(char ch)

{

string OutChar = "";

OutChar += ch;

if (ch == 'S') {

return("AaBaA");

}

else

{

return(OutChar);

}

}

string SecondRule(char ch)

{

string OutChar = "";

OutChar += ch;

if (ch == 'S')

{

return("bSb");

}

else

{

return(OutChar);

}

}

string ThirdRule(char ch)

{

string OutChar = "";

OutChar += ch;

if (ch == 'A')

{

return("B");

}

else

{

return(OutChar);

}

}

string FourthRule(char ch)

{

string OutChar = "";

OutChar += ch;

if (ch == 'A')

{

return("Sa");

}

else

{

return(OutChar);

}

}

string FifthRule(char ch)

{

string OutChar = "";

OutChar += ch;

if (ch == 'A')

{

return("a");

}

else

{

return(OutChar);

}

}

string SixthRule(char ch)

{

string OutChar = "";

OutChar += ch;

if (ch == 'A')

{

return("");

}

else

{

return(OutChar);

}

}

string SeventhRule(char ch)

{

string OutChar = "";

OutChar += ch;

if (ch == 'B')

{

return("bS");

}

else

{

return(OutChar);

}

}

string EigthRule(char ch)

{

string OutChar = "";

OutChar += ch;

if (ch == 'B')

{

return("A");

}

else

{

return(OutChar);

}

}

private:

};

int main() {

setlocale(LC\_ALL, "Rus");

MyClass KC{ "S" };

cout << "Первое правило: " << endl;

KC.Out\_Str();

// Устанавливаем цепочку S

if (KC.Str[0] == 'S')

{

KC.Str.replace(KC.Str.find("S"), 1, KC.FirstRule(KC.Str[0]));

}

// Применяем первое правило

KC.Out\_Str();

// Проверяем полученную цепочку

cout << endl;

cout << "Второе правило: " << endl;

KC.Define\_Str("S");

// Устанавливаем цепочку S

KC.Out\_Str();

// Проверяем полученную цепочку

if (KC.Str[0] == 'S')

{

KC.Str.replace(KC.Str.find("S"), 1, KC.SecondRule(KC.Str[0]));

}

// Применяем второе правило

KC.Out\_Str();

// Проверяем полученную цепочку

cout << endl;

cout << "Третье правило: " << endl;

KC.Define\_Str("A");

// Устанавливаем цепочку S

KC.Out\_Str();

// Проверяем полученную цепочку

if (KC.Str[0] == 'A')

{

KC.Str.replace(KC.Str.find("A"), 1, KC.ThirdRule(KC.Str[0]));

}

// Применяем второе правило

KC.Out\_Str();

// Проверяем полученную цепочку

cout << endl;

cout << "Четвёртое правило: " << endl;

KC.Define\_Str("A");

// Устанавливаем цепочку S

KC.Out\_Str();

// Проверяем полученную цепочку

if (KC.Str[0] == 'A')

{

KC.Str.replace(KC.Str.find("A"), 1, KC.FourthRule(KC.Str[0]));

}

// Применяем второе правило

KC.Out\_Str();

// Проверяем полученную цепочку

cout << endl;

cout << "Пятое правило: " << endl;

KC.Define\_Str("A");

// Устанавливаем цепочку S

KC.Out\_Str();

// Проверяем полученную цепочку

if (KC.Str[0] == 'A')

{

KC.Str.replace(KC.Str.find("A"), 1, KC.FifthRule(KC.Str[0]));

}

// Применяем второе правило

KC.Out\_Str();

// Проверяем полученную цепочку

cout << endl;

cout << "Шестое правило: " << endl;

KC.Define\_Str("A");

// Устанавливаем цепочку S

KC.Out\_Str();

// Проверяем полученную цепочку

if (KC.Str[0] == 'A')

{

KC.Str.replace(KC.Str.find("A"), 1, KC.SixthRule(KC.Str[0]));

}

// Применяем второе правило

KC.Out\_Str();

// Проверяем полученную цепочку

cout << endl;

cout << "Седьмое правило: " << endl;

KC.Define\_Str("B");

// Устанавливаем цепочку S

KC.Out\_Str();

// Проверяем полученную цепочку

if (KC.Str[0] == 'B')

{

KC.Str.replace(KC.Str.find("B"), 1, KC.SeventhRule(KC.Str[0]));

}

// Применяем второе правило

KC.Out\_Str();

// Проверяем полученную цепочку

cout << endl;

cout << "Восьмое правило: " << endl;

KC.Define\_Str("B");

// Устанавливаем цепочку S

KC.Out\_Str();

// Проверяем полученную цепочку

if (KC.Str[0] == 'B')

{

KC.Str.replace(KC.Str.find("B"), 1, KC.EigthRule(KC.Str[0]));

}

// Применяем второе правило

KC.Out\_Str();

// Проверяем полученную цепочку

cout << endl;

}

**Конец листинга 1**

**Листинг программы 2**

#include <iostream>

#include <string>

using namespace std;

class MyClass

{

public:

string Str = "S";

void Define\_Str(string str)

{

Str = str;

}

void Out\_Str()

{

cout<<"Вывод: "<< Str<< endl;

}

string FirstRule(char ch)

{

string OutChar = "";

OutChar += ch;

if (ch == 'S')

{

return("AaBaA");

}

}

string SecondRule(char ch)

{

string OutChar = "";

OutChar += ch;

if (ch == 'S')

{

return("bSb");

}

}

string ThirdRule(char ch)

{

string OutChar = "";

OutChar += ch;

if (ch == 'A')

{

return("B");

}

}

string FourthRule(char ch)

{

string OutChar = "";

OutChar += ch;

if (ch == 'A')

{

return("Sa");

}

}

string FifthRule(char ch)

{

string OutChar = "";

OutChar += ch;

if (ch == 'A')

{

return("a");

}

}

string SixthRule(char ch)

{

string OutChar = "";

OutChar += ch;

if (ch == 'A')

{

return("");

}

}

string SeventhRule(char ch)

{

string OutChar = "";

OutChar += ch;

if (ch == 'B')

{

return("bS");

}

}

string EigthRule(char ch)

{

string OutChar = "";

OutChar += ch;

if (ch == 'B')

{

return("A");

}

else

{

return(OutChar);

}

}

void DoRule(int Rule)

{

switch (Rule)

{

case 1:

for (unsigned int i = 0; i < Str.size(); i++)

{

if (Str[i] == 'S')

{

Str.replace(Str.find("S"), 1, FirstRule(Str[i]));

break;

}

}

break;

case 2:

for (unsigned int i = 0; i < Str.size(); i++)

{

if (Str[i] == 'S')

{

Str.replace(Str.find("S"), 1, SecondRule(Str[i]));

break;

}

}

break;

case 3:

for (unsigned int i = 0; i < Str.size(); i++)

{

if (Str[i] == 'A')

{

Str.replace(Str.find("A"), 1, ThirdRule(Str[i]));

break;

}

}

break;

case 4:

for (unsigned int i = 0; i < Str.size(); i++)

{

if (Str[i] == 'A')

{

Str.replace(Str.find("A"), 1, FourthRule(Str[i]));

break;

}

}

break;

case 5:

for (unsigned int i = 0; i < Str.size(); i++)

{

if (Str[i] == 'A')

{

Str.replace(Str.find("A"), 1, FifthRule(Str[i]));

break;

}

}

break;

case 6:

for (unsigned int i = 0; i < Str.size(); i++)

{

if (Str[i] == 'A')

{

Str.replace(Str.find("A"), 1, SixthRule(Str[i]));

break;

}

}

break;

case 7:

for (unsigned int i = 0; i < Str.size(); i++)

{

if (Str[i] == 'B')

{

Str.replace(Str.find("B"), 1, SeventhRule(Str[i]));

break;

}

}

break;

case 8:

for (unsigned int i = 0; i < Str.size(); i++)

{

if (Str[i] == 'B')

{

Str.replace(Str.find("B"), 1, EigthRule(Str[i]));

break;

}

}

break;

default:

break;

}

}

private:

};

int main() {

int Rule = 0;

setlocale(LC\_ALL, "Rus");

MyClass KC{ "S" };

cout << "Изначальная цепочка: " << endl;

KC.Out\_Str();

// Устанавливаем цепочку S

cout << "Введите номер правила: " << endl;

cin >> Rule;

KC.DoRule(Rule);

KC.Out\_Str();

while (Rule != 0) {

cout << "Введите номер правила: " << endl;

cin >> Rule;

KC.DoRule(Rule);

KC.Out\_Str();

}

}

**Конец листинга 2**