Федеральное агентство по образованию Российской Федерации

Государственное образовательное учреждение

высшего профессионального образования

Нижегородский государственный университет им. Н.И. Лобачевского

Факультет Вычислительной математики и кибернетики

Отчёт по лабораторной работе

Реализация динамической структуры данных типа «стек» на основе линейных односвязных списков   
и её практическое применение для задания локального оператора перехода для двумерных клеточных автоматов

Выполнил:

студент ф-та ВМК гр. 82-01

Савичев М. Р.

Проверил:

ассистент каф. МО ЭВМ, ВМК

Кустикова В. Д.

Нижний Новгород

2012

Оглавление

[Введение 3](#_Toc344144256)

[Постановка задачи 4](#_Toc344144257)

[Руководство пользователя 5](#_Toc344144258)

[Руководство программиста 8](#_Toc344144259)

[Описание структуры программы 8](#_Toc344144260)

[Описание структур данных 9](#_Toc344144261)

[Описание классов 9](#_Toc344144262)

[Описание алгоритмов 12](#_Toc344144263)

[Реализация стека на списках 12](#_Toc344144264)

[Алгоритм перевода формулы в постфиксную форму 12](#_Toc344144265)

[Алгоритм использования формулы в постфиксном виде для вычисления 12](#_Toc344144266)

[Заключение 13](#_Toc344144267)

[Литература 14](#_Toc344144268)

[Приложение. Исходные коды 15](#_Toc344144269)

[Приложение А. Модуль Draw.cpp 15](#_Toc344144270)

[Приложение Б. Модуль main.cpp 18](#_Toc344144271)

[Приложение В. Заголовочный файл Configuration.h 19](#_Toc344144272)

[Приложение Г. Модуль Configuration.cpp 19](#_Toc344144273)

[Приложение Д. Заголовочный файл CellAuto.h 20](#_Toc344144274)

[Приложение Е. Модуль CellAuto.cpp 20](#_Toc344144275)

[Приложение Ж. Заголовочный файл CellAuto2D.h 20](#_Toc344144276)

[Приложение З. Модуль CellAuto2D.cpp 21](#_Toc344144277)

[Приложение И. Заголовочный файл CellAuto2DPersonal.h 23](#_Toc344144278)

[Приложение К. Модуль CellAuto2DPersonal.cpp 23](#_Toc344144279)

[Приложение Л. Заголовочный файл TDataCom.h 24](#_Toc344144280)

[Приложение М. Заголовочный файл List.h 25](#_Toc344144281)

[Приложение Н. Модуль List.cpp 25](#_Toc344144282)

[Приложение О. Заголовочный файл ListStack.h 27](#_Toc344144283)

[Приложение П. Модуль ListStack.cpp 27](#_Toc344144284)

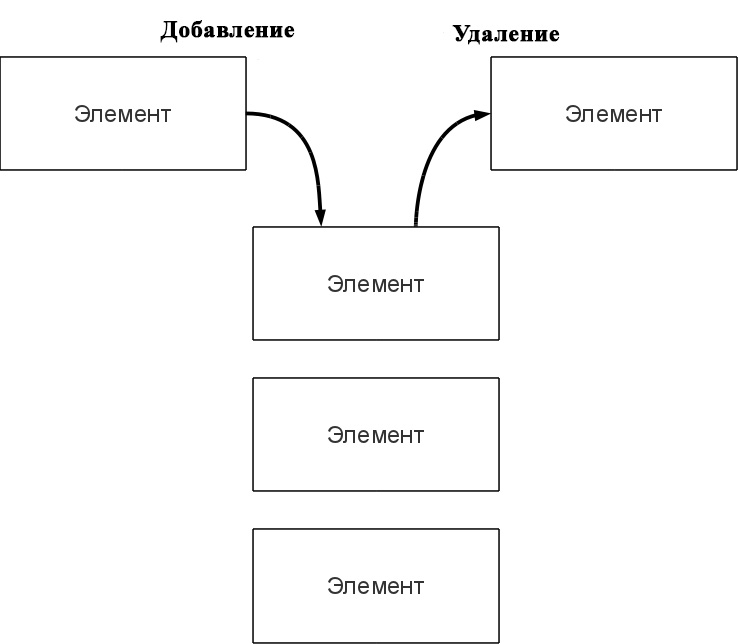
[Приложение Р. Заголовочный файл Postfix.h 28](#_Toc344144285)

[Приложение С. Модуль Postfix.cpp 28](#_Toc344144286)

# Введение

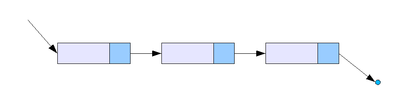
Стек (англ. stack — стопка) — структура данных, в которой доступ к элементам организован по принципу LIFO(англ. last in — first out, «последним пришёл — первым вышел»). Чаще всего принцип работы стека сравнивают со стопкой тарелок: чтобы взять вторую сверху, нужно снять верхнюю.

Добавление элемента возможно только в вершину стека (добавленный элемент становится первым сверху). Удаление элемента тоже возможно только из вершины стека, при этом второй сверху элемент становится верхним.



1. Принцип LIFO

Односвязный список — базовая динамическая структура данных, состоящая из узлов, каждый из которых содержит как собственно данные, так и одну ссылку на следующий узел списка. Принципиальным преимуществом перед массивом является структурная гибкость: порядок элементов связного списка может не совпадать с порядком расположения элементов данных в памяти компьютера, а порядок обхода списка всегда явно задаётся его внутренними связями.



1. Односвязный список

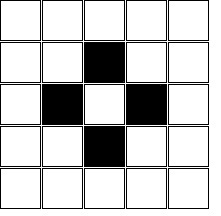
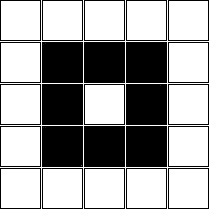
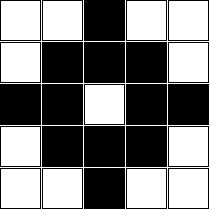
Здесь ссылка в каждом узле указывает на следующий узел в списке. В односвязном списке можно передвигаться только в сторону конца списка. Узнать адрес предыдущего элемента, опираясь на содержимое текущего узла, невозможно.

Приведенные выше структуры данных мы и будем использовать в нашей лабораторной работе.

# Постановка задачи

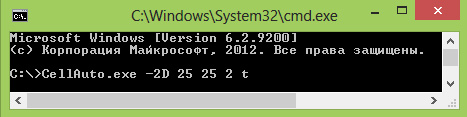
Целью данной лабораторной работы является разработка программной системы, которая визуализирует поведение двумерного клеточного автомата (ДКА) с заданной размерностью с помощью библиотеки компьютерного зрения «OpenCV». Разработать три способа задания начальной конфигурации: тестовая, случайная (с помощью генератора псевдослучайных чисел) и пользовательская (кликом мышкой). Включить в программу выбор окрестности клеточного автомата, по которой будет пересчитываться следующая конфигурация. Разработать возможность задания собственной окрестности.

Исходные данные:

* n – размер строки
* m – размер столбца
* Тип окрестности:
  + 0 – Неймана  
    
  + 1 –Мура   
    
  + 2 – Мвона  
    
  + 3- пользовательская
* Тип начальной конфигурации:
  + t - тестовая
  + r - случайная
  + p - пользовательская

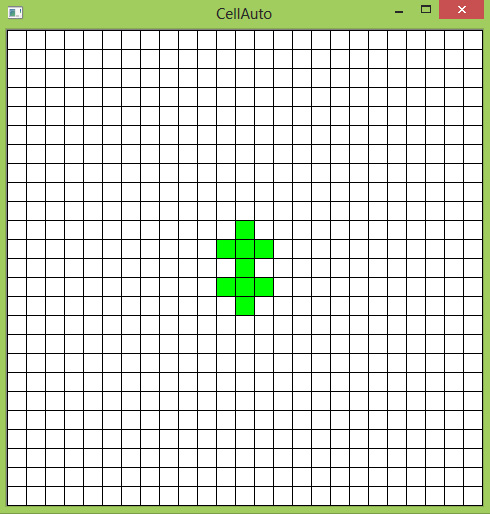
# Руководство пользователя

Для того чтобы воспользоваться программой необходимо запустить из консоли исполняемый файл CellAuto.exe с параметрами – исходными данными. Первый параметр указывает, что мы хотим создать ДКА, второй – размер строки, третий – размер столбца, четвертый – тип окрестности, пятый – тип начальной конфигурации. Пример с возможной комбинацией параметром на рис. 3.



1. Возможная комбинация параметров

При вводе такой комбинации по нажатию клавиши «Enter» создастся ДКА с окрестностью Мвона с тестовой конфигурацией (рис. 4).

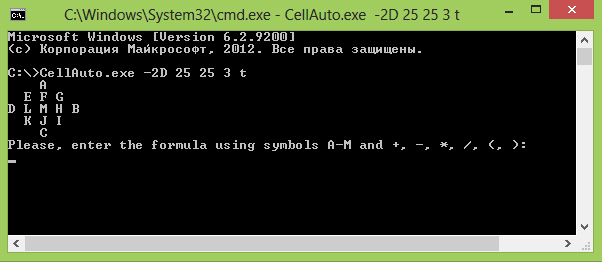


1. ДКА с тестовой конфигурацией

Чтобы задать пользовательскую конфигурацию, нужно заменить последний параметр на «p» и, с помощью кликов мышью, задать конфигурацию.

Подробное описание возможных исходных данных приведено выше, в пункте «Постановка задачи».

Чтобы задать пользовательскую окрестность необходимо изменить четвертый параметр на «3». После нажатия клавиши «Enter» программа выдаст сообщения с просьбой ввести формулу пересчета состояния клетки (рис. 5), также показаны соседние клетки и символы, которые можно использовать в формуле.



1. Просьба ввести формулу пересчета состояния клетки

При вводе неверных параметров программа выдаст сообщение о неверных параметрах: «Incorrect parameters» или «Incorrect formula».

Далее по нажатию клавиши «Enter» запустится слайд-шоу, нажатие клавиши «Space» останавливает слайд-шоу. При повторном нажатии будет появляться одна следующая конфигурация. Клавиша «Enter» продолжает слайд-шоу.

Выход из программы осуществляется нажатием клавиши «Esc».

# Руководство программиста

## Описание структуры программы

Данная программа разрабатывалась в среде Visual Studio. Решение состоит из трех проектов:

1. CellAuto – основная программа
2. CellAutoLib – статическая библиотека
3. TestApp – тестовая программа

Основная программа использует статическую библиотеку CellAutoLib, которая содержит все объявления и реализации классов, и библиотеку компьютерного зрения OpenCV для визуализации клеточного автомата на экране.

Тестовая программа, используя Google Testing Framework, тестирует методы классов из библиотеки CellAutoLib.

CellAutoLib состоит из восьми модулей:

* Configuration – класс конфигурации клеточного автомата (КА);
* CellAuto – абстрактный класс КА;
* CellAuto2D – классы ДКА:
  + CellAuto2D – абстрактный класс ДКА;
  + CellAuto2DNeumann – класс ДКА с окрестностью Неймана;
  + CellAuto2DMoore – класс ДКА с окрестностью Мура;
  + CellAuto2DMvon – класс ДКА с окрестностью Мвона;
* CellAuto2DPersonal – класс ДКА с пользовательской окрестностью;
* List – классы линейного односвязного списка и элемента списка;
* ListStack – класс стека, основанного на линейном односвязном списке;
* Postfix – класс для перевода формулы в постфиксную форму;
* TDataCom – класс ошибок структуры данных.

CellAuto состоит из двух модулей: Draw.cpp и main.cpp. Первый содержит функции, которые, работая с классами из CellAutoLib и с OpenCV, создают визуализацию клеточного автомата.

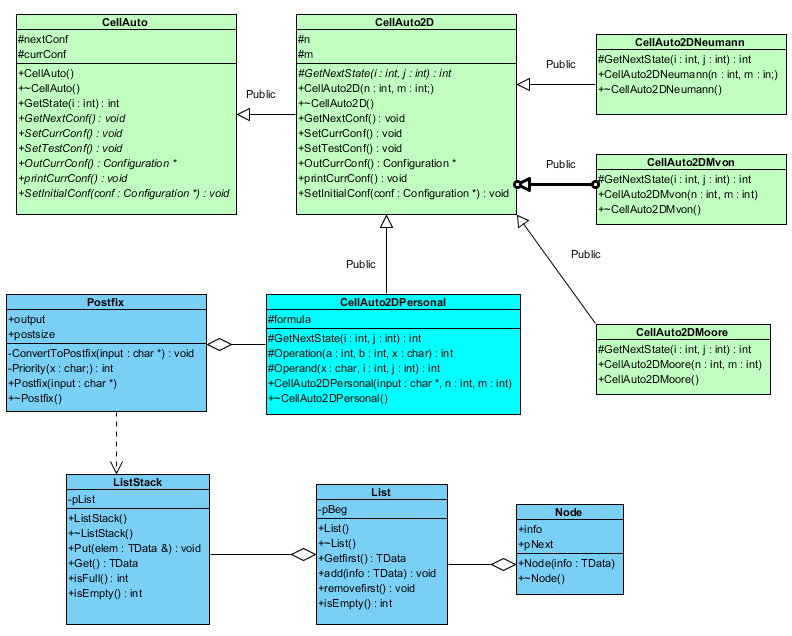
Состав модуля Draw.cpp:

* void RandomConfiguring(Configuration\* conf, int n, int m) – функция создания случайной конфигурации КА;
* void onMouseClick(int eventId, int x, int y, int flags, void \*userdata) – функция, которая вызывается при клике мышкой;
* void Lines(int n, int m) – функция отрисовки линий;
* int PersonalConfiguring2D(int n, int m, Configuration\* conf) – функция формирования пользовательской конфигурации КА;
* void Draw2D(CellAuto2D\* object, int n, int m) – функция отрисовки КА;
* void ConfiguringAndDrawing2D(int n, int m, int OptionSurround, int OptionConf, char \*formula)– функция, которая конфигурирует и отрисовывает КА.

Модуль main.cpp cодержит основную функцию, которая разбирает полученные параметры, проверяет их на корректность, а потом вызывает функцию ConfiguringAndDrawing2D, которая использует все остальные функции из Draw.cpp.

## Описание структур данных

Схема наследования классов в проекте CellAutoLib (рис. 6):



1. Схема наследования классов библиотеки CellAutoLib

### Описание классов

1. Класс CellAuto2D

|  |  |
| --- | --- |
| Поле | Описание |
| int n | Размер строки |
| int m | Размер столбца |

|  |  |
| --- | --- |
| Метод | Описание |
| CellAuto1D(int, int) | Конструктор с параметрами: размерность КА, размер очереди |
| ~CellAuto1D() | Деструктор |
| virtual int GetNextState(int)=0 | Возвращает следующее состояние клетки, входной параметр – номер клетки |

1. Класс CellAuto2DNeumann

|  |  |
| --- | --- |
| Метод | Описание |
| CellAuto1DNeumann(int, int) | Конструктор с параметрами: размер строки и размер столбца |
| ~CellAuto1DNeumann() | Деструктор |

1. Класс CellAuto2DMoore

|  |  |
| --- | --- |
| **Метод** | **Описание** |
| CellAuto1DMoore (int, int) | Конструктор с параметрами: размер строки и размер столбца |
| ~CellAuto1DMoore () | Деструктор |

1. Класс CellAuto2DMvon

|  |  |
| --- | --- |
| **Метод** | **Описание** |
| CellAuto1DMvon (int, int) | Конструктор с параметрами: размер строки и размер столбца |
| ~CellAuto1DMvon () | Деструктор |

1. Класс TDataCom

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| int RetCode | Код состояния данных |

|  |  |
| --- | --- |
| **Метод** | **Описание** |
| void SetRetCode(int) | Устанавливает код состояния, входной параметр – код |
| int GetRetCode() | Возвращает код состояния |
| TDataCom() | Конструктор |
| virtual ~TDataCom() | Деструктор |

1. Класс Node

|  |  |
| --- | --- |
| Поле | Описание |
| TData info | Данные |
| Node \*pNext | Указатель на следующий элемент |

|  |  |
| --- | --- |
| Метод | Описание |
| Node(TData) | Конструктор, который создает список из одного элемента с данными, переданными через параметр |
| ~Node() | Деструктор |

1. Класс List

|  |  |
| --- | --- |
| Поле | Описание |
| Node \*pBeg | Указатель на первый элемент списка |

|  |  |
| --- | --- |
| Метод | Описание |
| List() | Конструктор |
| ~List() | Деструктор |
| void add(TData) | Добавляет элемент из параметра в начало списка |
| TData getfirst() | Возвращает первый элемент |
| void removefirst() | Удалят первый элемент |
| int isEmpty() | Проверка на пустоту |

1. Класс ListStack

|  |  |
| --- | --- |
| Поле | Описание |
| List \*pList | Указатель |

|  |  |
| --- | --- |
| Метод | Описание |
| ListStack() | Конструктор |
| ~ListStack() | Деструктор |
| void Put(TData) | Положить элемент в стек |
| TData Get() | Вынуть |
| int isFull() | Проверка на полноту |
| int isEmpty() | Проверка на пустоту |

1. Класс Postfix

|  |  |
| --- | --- |
| Поле | Описание |
| char \*output | Формула в постфиксной форме |
| int postsize | Количество символов в постфиксной форме |

|  |  |
| --- | --- |
| Метод | Описание |
| Postfix(char\*) | Конструктор, входной параметр – формула в обычном виде |
| ~Postfix() | Деструктор |
| int Priority(char) | Возращает приоритет операции, входной параметр – символ операции |
| void ConvertToPostfix(char\*) | Функция, которая переводит формулу в постфиксную форму, входной параметр – формула в обычном виде |

1. Класс CellAuto2DPersonal

|  |  |
| --- | --- |
| Поле | Описание |
| Postfix \*formula | Указатель на объект формулы в постфиксном виде |

|  |  |
| --- | --- |
| Метод | Описание |
| CellAuto2DPersonal(char\*, int, int) | Конструктор, входные параметры: формула в обычном виде, размер строки и размер столбца |
| ~CellAuto2DPersonal() | Деструктор |
| int Operand(char, int, int) | Функция, ставящая в соответствие букве-операнду значение нужной клетки из окрестности, входные параметры: символ клекти из окрестности и координаты клетки, пересчет состояния которой происходит |
| int Operation(int, int, char) | Возвращает результат операции, которая передана как символ |

Если в приведенных выше таблицах в каком-то классе не был описан метод, то, значит, он был описан выше, в классе предка. «=0» после прототипа функции означает, что функция чисто виртуальная и реализация находится в классе-наследнике.

## Описание алгоритмов

Большинство функций, реализованных в процессе выполнения лабораторной работы, являются простыми, алгоритм которых ясен сразу же из их описания. Поэтому в этом разделе будут описаны наиболее сложные алгоритмы.

### Реализация стека на списках

Стек на списках был реализован на основе односвязных списков на указателях. То есть класс ListStack агрегирует класс List. В последнем классе есть функции удаления и вставки в начало списка и функции проверки на полноту и пустоту. С помощью них и были реализованы соответствующие функции в классе ListStack.

### Алгоритм перевода формулы в постфиксную форму

1. Создаем два стека: первый для хранения операндов с последующим добавлением операций, второй для операций и скобок
2. Просматриваем формулу слева направо
3. Если пришел операнд, кладем его в стек 1
4. Если приоритет пришедшей операции меньше приоритета операции, лежащей на вершине стека 2, то перекладываем операцию из стека 2 в стек 1. Пришедшую операцию кладем в стек 2
5. Если пришла правая скобка, перекладываем из стека 2 в стек 1 всё, что идет до левой скобки
6. В конце перекладываем всё, что осталось в стеке 2, в стек 1

### Алгоритм использования формулы в постфиксном виде для вычисления

1. Создаем стек для хранения операндов
2. Просматриваем формулу слева направо
3. Если пришел операнд, кладем его в стек
4. Если пришла операция, изымаем два операнда из стека и производим операцию, а результат в стек
5. В конце в стеке окажется один элемент – результат

# Заключение

В ходе выполнения лабораторной работы была разработана программа, позволяющая визуализировать поведение двумерного клеточного автомата. Реализована возможность выбора окрестности, в том числе и задаваемой вручную формулой. Проведенные эксперименты показали, что программа работает корректно.

# Литература

1. *Страуструп Б*. Программирование: принципы и практика использования C++.: Пер. с англ. – М.: ООО «И. Д. Вильямс», 2011.
2. *Наумов Л., Шалыто А*. Клеточные автоматы. Реализация и эксперименты. Статья опубликована в журнале «Мир ПК», 2003, №8, с. 64-71.
3. Структура данных типа стек [<http://ru.wikipedia.org/wiki/Стек>]
4. Структура данных связный список [<http://ru.wikipedia.org/wiki/Связный_список>]

# Приложение. Исходные коды

## Приложение А. Модуль Draw.cpp

#include "CellAuto2D.h"

#include "CellAuto2DPersonal.h"

#include <iostream>

#include <opencv2/opencv.hpp>

#include <time.h>

using namespace cv;

using namespace std;

#define ESC\_KEY 27

#define ENTER\_KEY 13

#define SPACE\_KEY 32

#define DELAY 100

int width=500, height=500;

const char\* winName = "CellAuto";

const int lineThick=1;

Mat image;

struct DataForMouseClick

{

int n;

int m;

Configuration\* conf;

};

void RandomConfiguring(Configuration\* conf, int n, int m)

{

srand(time(NULL));

for(int i=0; i<n\*m; i++)

{

conf->conf[i]=rand()%2;

}

}

void onMouseClick(int eventId, int x, int y, int flags, void \*userdata)

{

if (eventId != CV\_EVENT\_LBUTTONDOWN) return;

DataForMouseClick\* data=(DataForMouseClick\*)userdata;

int widthCell=(width-(data->n+1)\*lineThick)/data->n, heightCell=(height-(data->m+1)\*lineThick)/data->m;

int xLTAngle = (x / (widthCell + lineThick)) \* (widthCell + lineThick) + lineThick;

int yLTAngle = (y / (heightCell + lineThick)) \* (heightCell + lineThick) + lineThick;

if (data->conf->conf[y/(heightCell+lineThick)\*data->n+x/(widthCell+lineThick)]==0)

{

rectangle(image, cvRect(xLTAngle, yLTAngle, widthCell, heightCell), CV\_RGB(0, 100, 0), CV\_FILLED);

data->conf->conf[y/(heightCell+lineThick)\*data->n+x/(widthCell+lineThick)]=1;

}

else

{

rectangle(image, cvRect(xLTAngle, yLTAngle, widthCell, heightCell), CV\_RGB(255, 255, 255), CV\_FILLED);

data->conf->conf[y/(heightCell+lineThick)\*data->n+x/(widthCell+lineThick)]=0;

}

imshow(winName, image);

}

imshow(winName, image);

}

void Lines(int n, int m)

{

int widthCell=(width-(n+1)\*lineThick)/n, heightCell=(height-(m+1)\*lineThick)/m;

int begin = 0;

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

{

begin = i \* (widthCell + lineThick);

line(image, cvPoint(begin, 0), cvPoint(begin, height), CV\_RGB(0, 0, 0),

lineThick);

}

begin = 0;

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

{

begin = i \* (heightCell + lineThick);

line(image, cvPoint(0, begin), cvPoint(width, begin), CV\_RGB(0, 0, 0),

lineThick);

}

}

int PersonalConfiguring2D(int n, int m, Configuration\* conf)

{

int key=0;

namedWindow(winName,1);

DataForMouseClick data;

data.n=n;

data.m=m;

data.conf=conf;

int widthCell=(width-(n+1)\*lineThick)/n, heightCell=(height-(m+1)\*lineThick)/m;

int xLTAngle=0;

int yLTAngle=0;

cvSetMouseCallback(winName, onMouseClick, (void\*)(&data));

image.release();

width=widthCell\*n+lineThick\*(n+1);

height=heightCell\*m+lineThick\*(m+1);

image.create(width,height,CV\_32FC3);

rectangle(image, cvRect(0, 0, width, height), CV\_RGB(255, 255, 255), CV\_FILLED);

Lines(n,m);

imshow(winName,image);

while (key!=ENTER\_KEY && key!=ESC\_KEY)

{

key=waitKey();

}

if (key==ESC\_KEY) { image.release(); destroyAllWindows(); return -1; }

image.release();

return 0;

}

void Draw2D(CellAuto2D\* object, int n, int m)

{

int key=0;

namedWindow(winName,1);

int widthCell=(width-(n+1)\*lineThick)/n, heightCell=(height-(m+1)\*lineThick)/m;

int xLTAngle=0;

int yLTAngle=0;

bool show=false;

while (key!=ESC\_KEY)

{

switch (key)

{

case ENTER\_KEY: { show=true; break; }

case SPACE\_KEY: { show=false; break; }

default: { break; }

}

image.release();

width=widthCell\*n+lineThick\*(n+1);

height=heightCell\*m+lineThick\*(m+1);

image.create(width,height,CV\_32FC3);

rectangle(image, cvRect(0, 0, width, height), CV\_RGB(255, 255, 255), CV\_FILLED);

Lines(n,m);

for(int i=0; i<n\*m; i++)

{

if (object->GetState(i)==1)

{

xLTAngle = (i%n)\*(widthCell + lineThick) + lineThick;

yLTAngle = (i/n)\*(heightCell + lineThick) + lineThick;

rectangle(image, cvRect(xLTAngle, yLTAngle, widthCell, heightCell), CV\_RGB(0, 100, 0), CV\_FILLED);

}

}

imshow(winName,image);

object->GetNextConf();

object->SetCurrConf();

if (show==true) key=waitKey(DELAY);

if (show==false) key=waitKey();

}

image.release();

destroyAllWindows();

}

void ConfiguringAndDrawing2D(int n, int m, int OptionSurround, int OptionConf, char \*formula)

{

Configuration conf(n\*m);

CellAuto2D \*object;

switch (OptionSurround)

{

case 0:

{

object=new CellAuto2DNeumann(n, m);

break;

}

case 1:

{

object=new CellAuto2DMoore(n, m);

break;

}

case 2:

{

object=new CellAuto2DMvon(n, m);

break;

}

case 3:

{

object=new CellAuto2DPersonal(formula, n, m);

break;

}

default:

{

return;

}

}

switch (OptionConf)

{

case 0:

{

object->SetTestConf();

break;

}

case 1:

{

RandomConfiguring(&conf, n, m);

object->SetInitialConf(&conf);

break;

}

case 2:

{

int tmp=PersonalConfiguring2D(n,m, &conf);

if (tmp==-1) { delete object; return; }

object->SetInitialConf(&conf);

break;

}

default:

{

delete object;

cout << "Incorrect parameters\n";

return;

}

}

Draw2D(object, n, m);

delete object;

}

## Приложение Б. Модуль main.cpp

#include <iostream>

#include <conio.h>

#include "CellAuto2D.h"

#include "CellAuto1D.h"

using namespace std;

#define WAITKEY \_getch();

void ConfiguringAndDrawing1D(int, int, int, int);

void ConfiguringAndDrawing2D(int, int, int, int, char\*);

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

{

if (argc!=6)

{

cout << "Incorrect parameters\n";

WAITKEY

return -1;

}

int n=atoi(argv[2]);

int m=atoi(argv[3]);

if (n<=0 || m<=0)

{

cout << "Incorrect parameters\n";

WAITKEY

return -1;

}

int Surround=atoi(argv[4]);

int OptionConf=-1;

if (strcmp(argv[5], "t")==0) OptionConf=0;

if (strcmp(argv[5], "r")==0) OptionConf=1;

if (strcmp(argv[5], "p")==0) OptionConf=2;

if (OptionConf==-1)

{

cout << "Incorrect parameters\n";

WAITKEY

return -1;

}

if (strcmp(argv[1], "-1D")==0)

{

if (Surround<0 || Surround>1)

{

cout << "Incorrect parameters\n";

WAITKEY

return -1;

}

ConfiguringAndDrawing1D(n, Surround, OptionConf, m);

}

if (strcmp(argv[1], "-2D")==0)

{

if (Surround<0 || Surround>3)

{

cout << "Incorrect parameters\n";

WAITKEY

return -1;

}

char \*formula=0;

if (Surround==3)

{

formula=new char[100];

cout << " A\n";

cout << " E F G\n";

cout << "D L M H B\n";

cout << " K J I\n";

cout << " C\n";

cout << "Please, enter the formula using symbols A-M and +, -, \*, /, (, ):\n";

cin >> formula;

cout << endl;

int i=0;

while (formula[i]!='\0')

{

if ((formula[i]<65 || formula[i]>77) && formula[i]!='+' && formula[i]!='-'

&& formula[i]!='\*' && formula[i]!='/' && formula[i]!='(' && formula[i]!=')')

{

cout << "Incorrect formula\n";

WAITKEY

return -1;

}

i++;

}

}

ConfiguringAndDrawing2D(n, m, Surround, OptionConf, formula);

}

return 0;

}

## Приложение В. Заголовочный файл Configuration.h

#ifndef \_\_CONFIGURATION\_H\_\_

#define \_\_CONFIGURATION\_H\_\_

class Configuration

{

protected:

int size;

public:

Configuration(int);

~Configuration();

int \*conf;

};

#endif

## Приложение Г. Модуль Configuration.cpp

#include <iostream>

using namespace std;

#include "Configuration.h"

Configuration::Configuration(int \_size)

{

size=\_size;

conf=new int[size];

memset(conf, 0, sizeof(int)\*size);

}

Configuration::~Configuration()

{

delete[] conf;

}

## Приложение Д. Заголовочный файл CellAuto.h

#ifndef \_\_CELLAUTO\_H\_\_

#define \_\_CELLAUTO\_H\_\_

#include "Configuration.h"

class CellAuto

{

protected:

Configuration \*nextConf, \*currConf;

public:

CellAuto();

virtual ~CellAuto();

int GetState(int);

virtual void SetCurrConf()=0;

virtual Configuration\* OutCurrConf()=0;

virtual void SetInitialConf(Configuration\*)=0;

virtual void GetNextConf()=0;

virtual void SetTestConf()=0;

virtual void printCurrConf()=0;

};

#endif

## Приложение Е. Модуль CellAuto.cpp

#include <iostream>

using namespace std;

#include "CellAuto.h"

CellAuto::CellAuto()

{

currConf=0;

nextConf=0;

}

CellAuto::~CellAuto()

{

currConf=0;

nextConf=0;

}

int CellAuto::GetState(int k)

{

return currConf->conf[k];

}

## Приложение Ж. Заголовочный файл CellAuto2D.h

#ifndef \_\_CELLAUTO2D\_H\_\_

#define \_\_CELLAUTO2D\_H\_\_

#include "CellAuto.h"

class CellAuto2D: public CellAuto

{

protected:

int n;

int m;

virtual int GetNextState(int, int)=0;

public:

CellAuto2D(int, int);

virtual ~CellAuto2D();

virtual void GetNextConf();

virtual void SetCurrConf();

virtual void SetTestConf();

virtual void printCurrConf();

virtual void SetInitialConf(Configuration\*);

virtual Configuration\* OutCurrConf();

};

class CellAuto2DNeumann: public CellAuto2D

{

protected:

virtual int GetNextState(int, int);

public:

virtual ~CellAuto2DNeumann();

CellAuto2DNeumann(int, int);

};

class CellAuto2DMoore: public CellAuto2D

{

protected:

virtual int GetNextState(int, int);

public:

virtual ~CellAuto2DMoore();

CellAuto2DMoore(int, int);

};

class CellAuto2DMvon: public CellAuto2D

{

protected:

virtual int GetNextState(int, int);

public:

virtual ~CellAuto2DMvon();

CellAuto2DMvon(int, int);

};

#endif

## Приложение З. Модуль CellAuto2D.cpp

#include <iostream>

using namespace std;

#include "CellAuto2D.h"

CellAuto2D::CellAuto2D(int \_n, int \_m)

{

n=\_n;

m=\_m;

currConf=new Configuration(n\*m);

nextConf=new Configuration(n\*m);

}

CellAuto2D::~CellAuto2D()

{

m=0;

n=0;

delete currConf;

delete nextConf;

}

void CellAuto2D::SetCurrConf()

{

Configuration \*tmp;

tmp=currConf;

currConf=nextConf;

nextConf=tmp;

}

CellAuto2DNeumann::CellAuto2DNeumann(int \_n, int \_m)

: CellAuto2D(\_n, \_m)

{

}

CellAuto2DNeumann::~CellAuto2DNeumann()

{

}

void CellAuto2D::GetNextConf()

{

int k=0;

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

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

{

nextConf->conf[k++]=GetNextState(i,j);

}

}

void CellAuto2D::SetTestConf()

{

for(int i=0; i<m\*n; i++)

currConf->conf[i]=0;

currConf->conf[m/2\*n+n/2]=1;

currConf->conf[(m/2-1)\*n+n/2]=1;

currConf->conf[(m/2+1)\*n+n/2]=1;

currConf->conf[(m/2-1)\*n+n/2-1]=1;

currConf->conf[(m/2-1)\*n+n/2+1]=1;

currConf->conf[(m/2-2)\*n+n/2]=1;

currConf->conf[(m/2+2)\*n+n/2]=1;

currConf->conf[(m/2+1)\*n+n/2-1]=1;

currConf->conf[(m/2+1)\*n+n/2+1]=1;

}

void CellAuto2D::printCurrConf()

{

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

{

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

cout << currConf->conf[j\*n+i] << ' ';

cout << endl;

}

cout << endl;

}

int CellAuto2DNeumann::GetNextState(int i, int j)

{

return (currConf->conf[j\*n+(i-1+n)%n]+currConf->conf[j\*n+(i+1)%n]

+currConf->conf[((j-1+m)%m)\*n+i]+currConf->conf[((j+1)%m)\*n+i])%2;

}

int CellAuto2DMoore::GetNextState(int i, int j)

{

return (currConf->conf[j\*n+(i-1+n)%n]+currConf->conf[j\*n+(i+1)%n]+currConf->conf[((j-1+m)%m)\*n+i]+currConf->conf[((j+1)%m)\*n+i]

+currConf->conf[((j-1+m)%m)\*n+(i-1+n)%n]+currConf->conf[((j-1+m)%m)\*n+(i+1)%n]+currConf->conf[((j+1)%m)\*n+(i-1+n)%n]

+currConf->conf[((j+1)%m)\*n+(i+1)%n])%2;

}

CellAuto2DMoore::CellAuto2DMoore(int \_n, int \_m)

: CellAuto2D(\_n, \_m)

{

}

CellAuto2DMoore::~CellAuto2DMoore()

{

}

int CellAuto2DMvon::GetNextState(int i, int j)

{

return (currConf->conf[j\*n+(i-1+n)%n]+currConf->conf[j\*n+(i+1)%n]+currConf->conf[((j-1+m)%m)\*n+i]+currConf->conf[((j+1)%m)\*n+i]

+currConf->conf[((j-1+m)%m)\*n+(i-1+n)%n]+currConf->conf[((j-1+m)%m)\*n+(i+1)%n]+currConf->conf[((j+1)%m)\*n+(i-1+n)%n]

+currConf->conf[((j+1)%m)\*n+(i+1)%n]+currConf->conf[j\*n+(i-2+n)%n]+currConf->conf[j\*n+(i+2)%n]

+currConf->conf[((j-2+m)%m)\*n+i]+currConf->conf[((j+2+m)%m)\*n+i])%2;

}

CellAuto2DMvon::CellAuto2DMvon(int \_n, int \_m)

: CellAuto2D(\_n, \_m)

{

}

CellAuto2DMvon::~CellAuto2DMvon()

{

}

void CellAuto2D::SetInitialConf(Configuration\* conf)

{

memcpy(currConf->conf, conf->conf, sizeof(int)\*n\*m);

}

Configuration\* CellAuto2D::OutCurrConf()

{

Configuration\* tmp=new Configuration(n\*m);

memcpy(tmp->conf, currConf->conf, sizeof(int)\*n\*m);

return tmp;

}

## Приложение И. Заголовочный файл CellAuto2DPersonal.h

#ifndef \_\_CELLAUTO2DPERSONAL\_H\_\_

#define \_\_CELLAUTO2DPERSONAL\_H\_\_

#include "CellAuto2D.h"

#include "Postfix.h"

class CellAuto2DPersonal: public CellAuto2D

{

protected:

Postfix \*formula;

virtual int GetNextState(int, int);

int Operation(int, int, char);

int Operand(char, int, int);

public:

CellAuto2DPersonal(char\*, int, int);

virtual ~CellAuto2DPersonal();

};

#endif

## Приложение К. Модуль CellAuto2DPersonal.cpp

#include "CellAuto2DPersonal.h"

#include "ListStack.h"

#include <math.h>

CellAuto2DPersonal::CellAuto2DPersonal(char \*input, int \_n, int \_m)

:CellAuto2D(\_n, \_m)

{

formula=new Postfix(input);

}

CellAuto2DPersonal::~CellAuto2DPersonal()

{

delete formula;

}

int CellAuto2DPersonal::Operand(char x, int i, int j)

{

if (x=='A') return currConf->conf[((j-2+m)%m)\*n+i];

if (x=='B') return currConf->conf[j\*n+(i+2)%n];

if (x=='C') return currConf->conf[((j+2+m)%m)\*n+i];

if (x=='D') return currConf->conf[j\*n+(i-2+n)%n];

if (x=='E') return currConf->conf[((j-1+m)%m)\*n+(i-1+n)%n];

if (x=='F') return currConf->conf[((j-1+m)%m)\*n+i];

if (x=='G') return currConf->conf[((j-1+m)%m)\*n+(i+1)%n];

if (x=='H') return currConf->conf[j\*n+(i+1)%n];

if (x=='I') return currConf->conf[((j+1)%m)\*n+(i+1)%n];

if (x=='J') return currConf->conf[((j+1)%m)\*n+i];

if (x=='K') return currConf->conf[((j+1)%m)\*n+(i-1+n)%n];

if (x=='L') return currConf->conf[j\*n+(i-1+n)%n];

if (x=='M') return currConf->conf[j\*n+(i+n)%n];

if (x==0) return 0;

if (x==1) return 1;

return -1;

}

int CellAuto2DPersonal::Operation(int A, int B, char x)

{

if (x=='+') return (A+B)%2;

if (x=='-') return abs(A-B)%2;

if (x=='\*') return A\*B;

if (x=='/')

if (B==0) return 0; else return A/B;

return -1;

}

int CellAuto2DPersonal::GetNextState(int i, int j)

{

ListStack \*stack=new ListStack;

for (int k=0; k<formula->postsize; k++)

{

if (formula->output[k]>=65 && formula->output[k]<=77)

stack->Put(formula->output[k]);

if (formula->output[k]=='+' || formula->output[k]=='-'

|| formula->output[k]=='\*' || formula->output[k]=='/')

{

char tmp1=stack->Get(), tmp2=stack->Get();

int operand1=Operand(tmp1, i, j);

int operand2=Operand(tmp2, i, j);

int operation=Operation(operand1, operand2, formula->output[k]);

stack->Put((char)operation);

}

}

int result=((int)stack->Get())%2;

delete stack;

return result;

}

## Приложение Л. Заголовочный файл TDataCom.h

#ifndef \_\_TDATACOM\_H\_\_

#define \_\_TDATACOM\_H\_\_

#define DataOK 0

#define DataFull 1

#define DataEmpty 2

class TDataCom

{

protected:

int RetCode;

void SetRetCode(int);

public:

TDataCom();

virtual ~TDataCom();

int GetRetCode();

};

#endif

Модуль TDataCom.cpp

*#include "TDataCom.h"*

*TDataCom::TDataCom()*

*{*

*RetCode=DataEmpty;*

*}*

*TDataCom::~TDataCom()*

*{*

*RetCode=DataEmpty;*

*}*

*int TDataCom::GetRetCode()*

*{*

*return RetCode;*

*}*

*void TDataCom::SetRetCode(int code)*

*{*

*RetCode=code;*

*}*

## Приложение М. Заголовочный файл List.h

#ifndef \_\_LIST\_H\_\_

#define \_\_LIST\_H\_\_

typedef char TData;

class Node

{

public:

TData info;

Node \*pNext;

Node(TData);

~Node();

};

class List

{

private:

Node \*pBeg;

public:

List();

~List();

int add(TData);

TData getfirst();

void remove(TData);

void removefirst();

int isEmpty();

};

#endif

## Приложение Н. Модуль List.cpp

#include "List.h"

Node::Node(TData \_info)

{

info=\_info;

pNext=0;

}

Node::~Node()

{

}

List::List()

{

pBeg=0;

}

List::~List()

{

if (pBeg!=0)

{

Node \*tmp;

while (pBeg->pNext!=0)

{

tmp=pBeg->pNext;

delete pBeg;

pBeg=tmp;

}

}

}

int List::add(TData \_info)

{

if (pBeg==0)

{

pBeg=new Node(\_info);

return 0;

}

/\*Node \*tmp=pBeg;

while (tmp->pNext!=0)

tmp=tmp->pNext;

tmp->pNext=new Node(\_info);

tmp->pNext->pNext=0;\*/

Node \*tmp=new Node(\_info);

if (tmp==0) return -1;

tmp->pNext=pBeg;

pBeg=tmp;

return 0;

}

TData List::getfirst()

{

if (pBeg==0) return '\_';

/\*Node \*tmp=pBeg;

while (tmp->pNext!=0)

tmp=tmp->pNext;

return tmp->info;\*/

return pBeg->info;

}

void List::removefirst()

{

if (pBeg==0) return;

Node \*tmp=pBeg->pNext;

delete pBeg;

pBeg=tmp;

}

void List::remove(TData \_info)

{

Node \*pCurr, \*pPrev;

pCurr=pBeg;

pPrev=0;

while (pCurr!=0 && pCurr->info!=\_info)

{

pPrev=pCurr;

pCurr=pCurr->pNext;

}

if (pCurr->pNext==0 && pCurr->info!=\_info) return;

if (pPrev==0)

{

pBeg=pBeg->pNext;

delete pCurr;

return;

}

pPrev->pNext=pCurr->pNext;

delete pCurr;

}

int List::isEmpty()

{

if (pBeg==0) return 0;

else return -1;

}

## Приложение О. Заголовочный файл ListStack.h

#ifndef \_\_LISTSTACK\_H\_\_

#define \_\_LISTSTACK\_H\_\_

#include "TDataCom.h"

#include "List.h"

class ListStack: public TDataCom

{

private:

List \*pList;

public:

ListStack();

~ListStack();

void Put(TData);

TData Get();

int isFull();

int isEmpty();

};

#endif

## Приложение П. Модуль ListStack.cpp

#include "ListStack.h"

ListStack::ListStack()

{

pList=new List;

}

ListStack::~ListStack()

{

delete pList;

}

int ListStack::isEmpty()

{

if (pList->isEmpty()==0)

{

SetRetCode(DataEmpty);

return DataEmpty;

}

return DataOK;

}

int ListStack::isFull()

{

int tmp=pList->add('~');

if (tmp!=0) return DataFull;

pList->remove('~');

return DataOK;

}

TData ListStack::Get()

{

if (isEmpty()==DataEmpty) return '\_';

TData tmp=pList->getfirst();

pList->removefirst();

return tmp;

}

void ListStack::Put(TData \_info)

{

if (isFull()==DataFull) return;

pList->add(\_info);

}

## Приложение Р. Заголовочный файл Postfix.h

#ifndef \_\_POSTFIX\_H\_\_

#define \_\_POSTFIX\_H\_\_

class Postfix

{

private:

void ConvertToPostfix(char\*);

int Priority(char);

public:

char \*output;

int postsize;

Postfix(char\*);

~Postfix();

};

#endif

## Приложение С. Модуль Postfix.cpp

#include "Postfix.h"

#include "ListStack.h"

#include <iostream>

Postfix::Postfix(char \*input)

{

postsize=0;

ConvertToPostfix(input);

}

Postfix::~Postfix()

{

delete output;

}

int Postfix::Priority(char x)

{

if (x=='/') return 3;

if (x=='\*') return 2;

//if ((x=='\*')||(x=='/')) return 2;

if ((x=='+')||(x=='-')) return 1;

if ((x=='(')||(x==')')) return 0;

return -1;

}

void Postfix::ConvertToPostfix(char \*input)

{

int size=0;

while (input[size]!='\0')

size++;

ListStack \*stack1=new ListStack;

ListStack \*stack2=new ListStack;

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

{

if (input[i]>=65 && input[i]<=77)

stack1->Put(input[i]);

if (input[i]=='+' || input[i]=='-' || input[i]=='\*' || input[i]=='/' || input[i]=='(')

{

if (stack2->isEmpty()!=DataEmpty)

{

char tmp=stack2->Get();

if (Priority(input[i])<Priority(tmp) && input[i]!='(')

stack1->Put(tmp);

else

stack2->Put(tmp);

stack2->Put(input[i]);

} else

stack2->Put(input[i]);

}

if (input[i]==')')

{

char tmp=stack2->Get();

while (tmp!='(')

{

stack1->Put(tmp);

tmp=stack2->Get();

}

}

}

while (stack2->isEmpty()!=DataEmpty)

stack1->Put(stack2->Get());

int brackets=0;

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

if (input[i]=='(' || input[i]==')') brackets++;

char \*result=new char[size-brackets];

for(int i=size-brackets-1; i>=0; i--)

result[i]=stack1->Get();

output=result;

postsize=size-brackets;

delete stack1;

delete stack2;

}