Министерство образования Российской Федерации
Государственное образовательное учреждение
Новгородский государственный университет имени Ярослава Мудрого

Политехнический колледж

РАЗРАБОТКА АЛГОРИТМА ИГРЫ «ТРУБОПРОВОД»

Пояснительная записка к курсовому проекту
по алгоритмическим языкам и программированию
по специальности 230105 «Программное обеспечение вычислительной техники и автоматизированных систем»

Студент гр.52351

Лопатин А.С.

«___»_____2008 г.

Преподаватель

Сазонова Н.В.

Оценка: _____

«___»_____2008 г.

Великий Новгород

2008

Содержание

1 Техническое задание....4
2 Математическое моделирование....4
3 Входные данные....4
4 Выходные данные....4
5 Алгоритм решения задачи....5
6 Список литературы....6
Приложение А....7
Приложение Б....10

1 Техническое задание

Дано поле заданной ширины и высоты и куски трубы, имеющие по два выхода: слева, справа, сверху или снизу (куски с противоположными концами — прямые, с соседними — угловые). Также, есть куски начала и конца имеющие один выход снизу и сверху соответственно. Куски начала и конца располагаются сверху и снизу соответственно. Обычные же куски трубы располагаются случайным образом, но так что возможен хотя бы один способ соединить их. На соединение кусков трубы дается некоторое время. Необходимо соединить куски воедино так, что бы их выходы совпадали и открыть кран до того как истекло время.

2 Математическое моделирование

Время, отведенное на решение задачи, высчитывается исходя из размера поля по следующей формуле:

timeLeft = width * height * 8 / (width + height) (1)

где timeLeft — отведенное на решение время в секундах, width — ширина поля, height — высота поля.

Данная формула была получена методом подбора при обработке полей разных размеров.

3 Входные данные

Входными данными в данной программе являются ширина и высота поля и положение кусков трубы.

4 Выходные данные

Выходными данными в данной программе являются сообщения «Удача: Вы выиграли» и «Неудача: Вы проиграли» в случае выигрыша или проигрыша соответственно. Также, выходными данными является вопрос о подтверждении выхода из программы «Подтверждение выхода: Игра еще не закончена. Выйти?».

5 Алгоритм решения задачи

При инициализации главного окна создаются все необходимые объекты и подключаются все обработчики событий. Потом запускается игра с шириной и высотой поля по-умолчанию. Запуск игры заключается в вычислении отведенного на решение времени, подготовки поля и запуска таймера времени с периодичностью одна секунда. По истечению каждой секунды таймер вызывает функцию secondPassed(), которая обновляет «табло» со временем в главном окне и проверяет не вышло ли время. Если оно еще не вышло — уменьшается переменная timeLeft на единицу, в противном случае вызывается функция fail(), которая прерывает игру и выдает сообщение о проигрыше.

Подготовка поля заключается в выделении памяти под поле, его заполнении одним из возможных вариантов соединения трубы со случайным поворотом каждого из кусков и заполнении некоторых из оставшихся свободных мест случайными кусками труб со случайными поворотами.

При нажатии левой кнопки по одному из кусков трубы она поворачивается против часовой стрелки, а при нажатии правой — по часовой. В случае если пользователь нажал на кнопку «Открыть кран» остановится таймер оставшегося времени и запустится таймер проверки трубы, который будет вызывать функцию checkStep() раз в полсекунды, проверяющую корректность соединенного куска трубы и имитирующую заполнение куска трубы водой в

случае удачи. В случае неудачи будет вызвана функция fail(). Если достигнут конец трубы — вызывается функция win(), которая прерывает игру и выдает сообщение о выигрыше. Также функция checkStep() блокирует возможность поворачивать кусок трубы, заполненного водой, при этом остальные куски еще можно будет вращать. Это сделано для того что бы пользователь смог исправить ситуацию при уже открытом кране.

В случае если пользователь не закончив игру попытается выйти из программы путем закрытия главного окна или выбора пункта «Выход» меню «Игра» ему будет выдан вопрос о подтверждении выхода, в случае же когда игра закончена — выход будет произведен без вопроса о подтверждении.

При нажатии на пункт «Новая» меню «Игра» будет показано диалоговое окно ввода ширины и высоты поля новой игры, в которых изначально заданы значения последней игры. В случае подтверждении данных (при нажатии на кнопку «OK») текущая игра будет прервана и начнется новая, с полем заданной длины и ширины.

6 Список литературы

1. Лафоре Р. «Объектно-ориентированное программирование в C++» Издательство «Питер», 2004 – 923с

2. «Qt 4.5.0: Qt Reference Documentation» [Электронный документ] (http://doc.trolltech.com/4.5/index.html) проверено 29.03.2009

Приложение А

(обязательное)

Фрагмент текста программы

```
...
inline void GameLayout::generateField()
{
    int x = 1 + qrand() % (width-2), y = 0;
    int lastX, lastY, size;

    pathBegin(x, y);
    ++y;

    while (y < height-1) {
        lastX = x; lastY = y;
        addPipe(x, y, angle);

        size = 1 + qrand() % (width/2);
        if (x > width/2) {
            if (size-x < 0)
                size = x;
            x -= size;
            addPipeLine(x+1, y, size, Qt::Horizontal);
        } else {
            if (size+x > width-1)
                size = width-1 - x;
            x += size;
            addPipeLine(lastX, lastY, size, Qt::Horizontal);
        }
        addPipe(x, y, angle);

        size = 1 + qrand() % (height/4);
        if (size+y > height-1)
            size = height-1 - y;
        y += size;
        addPipeLine(x, lastY, size, Qt::Vertical);
    }
    pathEnd(x, height-1);

    randomizeField();
}
```

```
...
void GameLayout::checkStep()
{
    static int x, y;

    if (firstCheckStep) {
        x = xBegin; y = yBegin; firstCheckStep = false;
        pipeImages[x][y]->light();
        return;
    }

    way = getNextWay(x, y);

    switch (way)
    {
        case toLeft:
            --x;
            if (x < 0 || !pipeImages[x][y]->getRight()) {
                emit fail(); return;
            }
            break;
        case toRight:
            ++x;
            if (x > width-1 || !pipeImages[x][y]->getLeft()) {
                emit fail(); return;
            }
            break;
        case toTop:
            --y;
            if (y < 0 || !pipeImages[x][y]->getBottom()) {
                emit fail(); return;
            }
            break;
        case toBottom:
            ++y;
            if (y > height-1 || !pipeImages[x][y]->getTop()) {
                emit fail(); return;
            }
            break;
    }
```

```
    pipeImages[x][y]->setRotatable(false);
    pipeImages[x][y]->light();

    if (field[x][y] == end)
        emit win();

    if (field[x][y] == free)
        emit fail();
}
...
```

Приложение Б

(обязательное)

Рисунок Б.1 — Схема функции generateField()

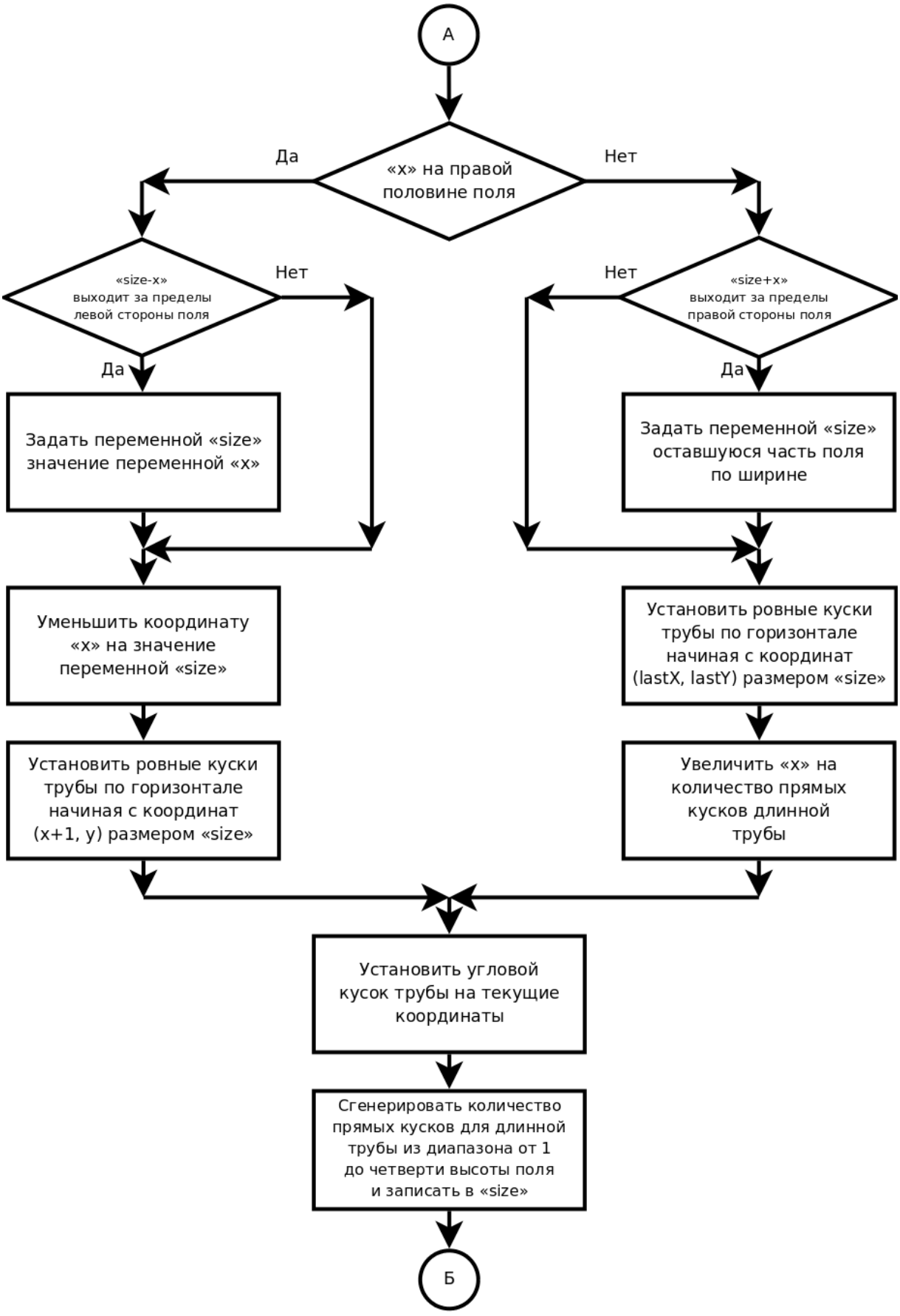

Рисунок Б.2 — Продолжение схемы функции generateField()

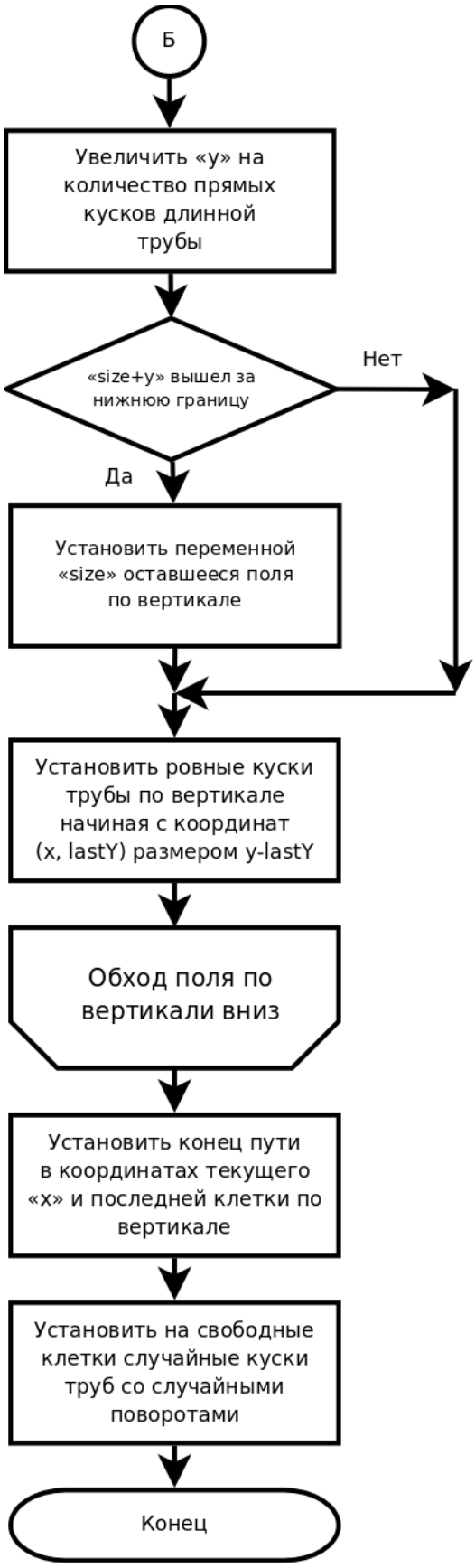

Рисунок Б.3 — Окончание схемы функции generateField()

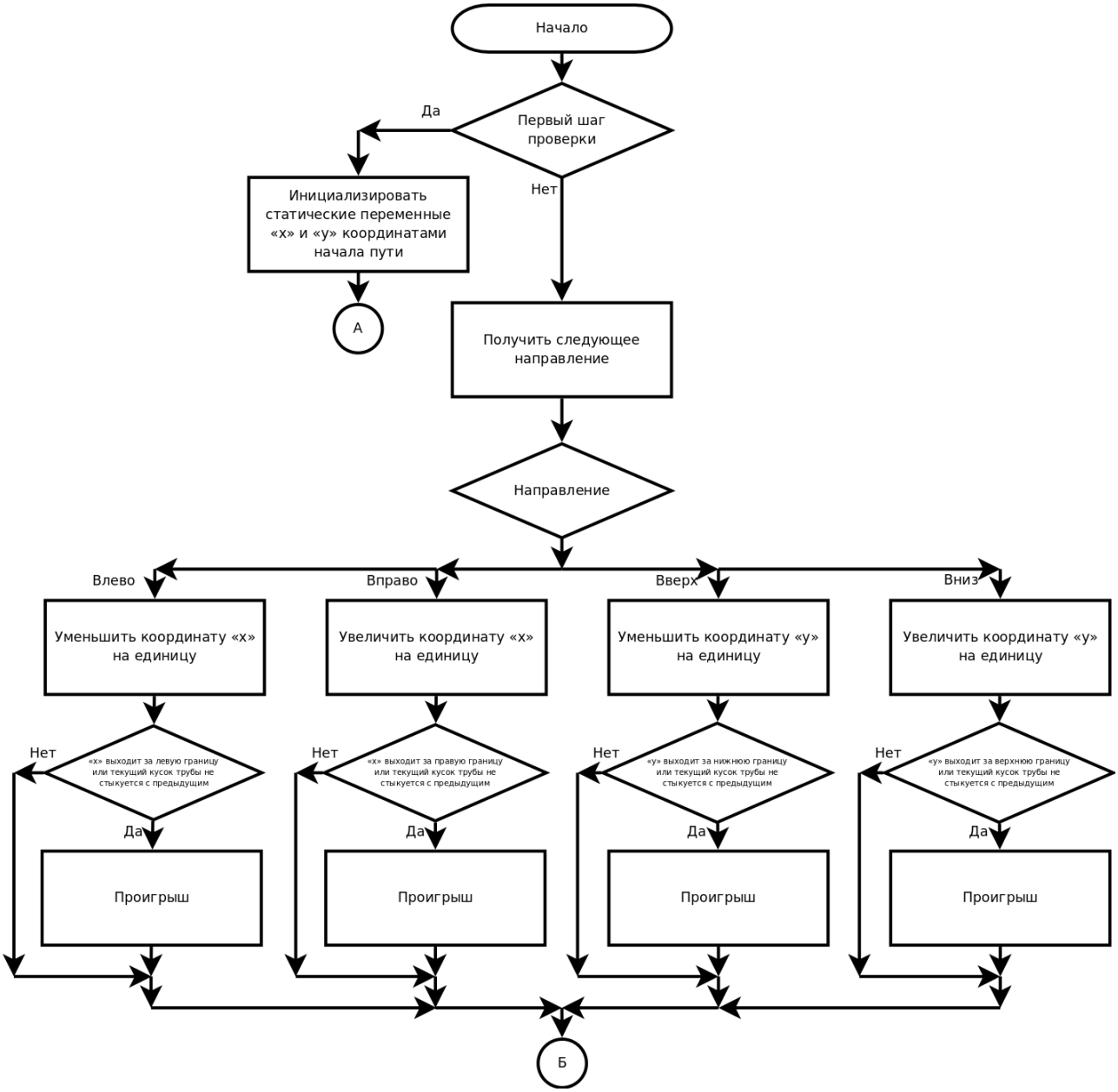

Рисунок Б.4 — Схема функции checkStep()

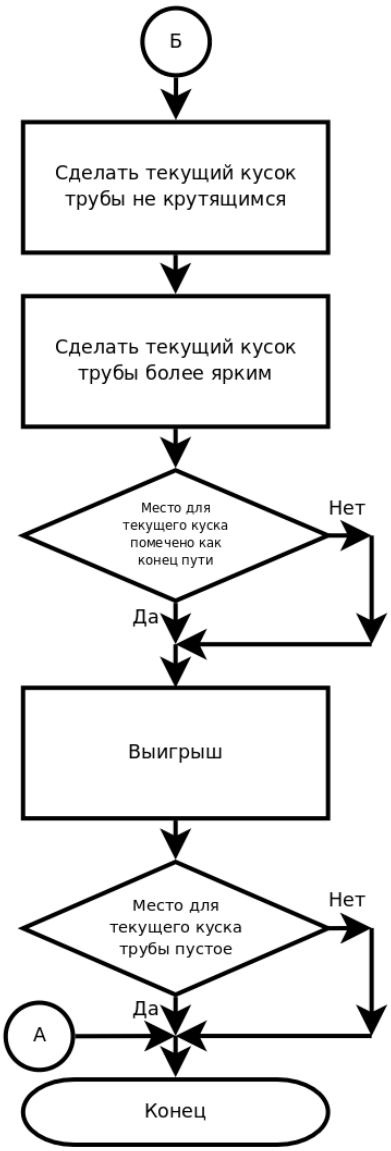

Рисунок Б.5 — Окончание схемы функции checkStep()