МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

ИНСТИТУТ НЕПРЕРЫВНОГО И ДИСТАНЦИОННОГО ОБРАЗОВАНИЯ

|  |
| --- |
| КАФЕДРА 43 |

ОЦЕНКА

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Старший преподаватель |  |  |  | Шумова Е.О. |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ |
| Программирование с использованием шаблонов объектно-ориентированного программирования |
| по дисциплине: Объектно-ориентированное программирование |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | Z9431 |  | 25.01.2022 |  | Пайвин А.А. |
|  | номер группы |  | подпись, дата |  | инициалы, фамилия |
| Студенческий билет № | 2018/4265 | |  |  |  |

Санкт-Петербург 2022

Задание

Согласно варианту № 6, составить программу, отыскивающую проход по лабиринту. Лабиринт должен представляться в виде карты-матрицы. Если элемент матрицы равен 1 – считать его стеной, если 0 – проходом. Для карты задаются вход и выход в виде элементов матрицы (они должны быть свободными, т.е. нулевыми). Проход по лабиринту возможен по горизонтали или вертикали, но не по диагонали. Программа должна найти проход через лабиринт от входа к выходу. После отыскания прохода программа выводит найденный путь в виде координат точек пути. Для хранения пути использовать стек. Матрицу задать в виде динамического массива.

.

Цель работы

Приобрести практический навык внедрения шаблонов объектно-ориентированного программирования в код программных систем

Разработка классов

Создадим абстрактный класс Task

class Task // Задача

{

public:

void virtual solve\_task() = 0; // решить задачу

};

Унаследуем от него класс labyrinth и реализуем метод solve\_task() , который и будем вызывать.

class labyrinth : public Task

{

point in;

point out;

point\_search p\_search;

stack <point\_search> path;

int count;

bool\*\* matrix;

public:

labyrinth(bool\*\* matrix\_lab, int count);

void solve\_task()override;

void set\_in(int x, int y);

void set\_out(int x, int y);

~labyrinth();

};

Создадим класс user, в котором будем вызывать метод для решения задачи

class user

{

public:

void To\_Do\_Working(Task& job)

{

job.solve\_task();

}

};

Также создадим две структуры, первую, point, для хранения параметров точки, x и y, вторую, point\_search, для хранения сторон вокруг точки, в какие стороны есть проход.

struct point {

int x;

int y;

};

struct point\_search : point {

bool up;

bool right;

bool down;

bool left;

};

Создадим статический двумерный массив в функции main и передадим его по указателю в конструктор при создании объекта класса labyrinth

bool matrix[20][20] = // вход 0 3, выход 17 19

{

{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1},

{1,0,0,0,1,1,0,1,1,1,0,1,1,1,0,0,0,0,1,1},

{1,1,1,0,0,0,0,1,1,1,0,1,1,1,0,1,1,0,0,1},

{0,0,0,0,1,0,1,1,0,0,0,1,0,1,0,1,1,0,1,1},

{1,0,1,1,1,0,1,1,1,1,0,0,0,0,0,1,0,0,1,1},

{1,0,1,1,1,0,1,0,0,0,1,0,1,1,1,1,1,0,1,1},

{1,0,0,0,1,1,0,0,1,0,0,0,0,1,1,1,1,0,1,1},

{1,1,1,0,1,1,0,1,1,1,1,1,0,0,0,1,1,1,1,1},

{1,0,0,0,1,1,0,0,0,1,1,1,1,1,1,1,1,1,1,1},

{1,0,1,1,1,1,0,1,0,0,0,1,1,1,1,0,1,0,0,1},

{1,0,0,0,0,1,0,1,1,1,0,0,0,1,0,0,0,0,1,1},

{1,1,1,0,1,1,0,1,1,1,1,1,0,1,1,0,1,1,1,1},

{1,1,1,0,0,0,0,1,1,1,0,0,0,0,0,0,1,1,1,1},

{0,1,0,0,0,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1},

{0,1,0,1,1,1,1,0,0,0,0,0,0,0,1,1,1,0,1,1},

{0,1,0,0,1,1,1,0,1,1,1,1,0,1,1,1,0,0,1,1},

{1,1,1,0,1,1,1,1,1,1,1,1,0,0,0,0,0,1,1,1},

{1,0,0,0,0,1,1,1,0,0,0,1,1,1,0,1,1,1,1,1},

{1,0,1,1,0,0,0,0,0,1,0,0,1,1,0,0,0,0,1,1},

{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1}

};

В конструкторе лабиринт будет создаваться с выделением динамической памяти, как требуется в задании. В деструкторе класса освободим выделенную память.

Диаграмма классов

Знаком «-» отмечены приватные методы

Знаком «+» отмечены публичные методы

Защищенные методы в данной работе не используются

Используется шаблонный метод

Диаграмма классов указана на рисунке 1

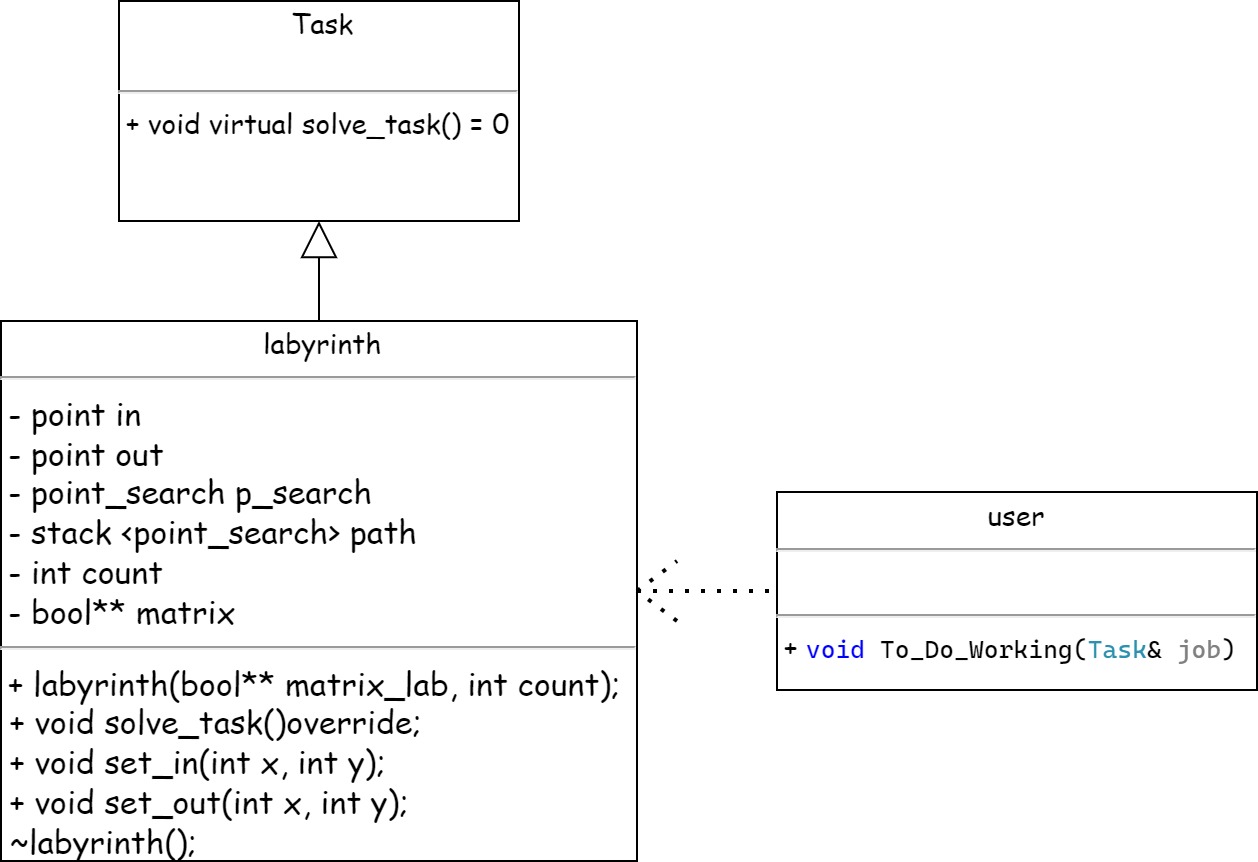


Рисунок 1

Листинг программы

#include <iostream>

#include <ctime>

#include <stack>

#include <deque>

using namespace std;

struct point {

int x;

int y;

};

struct point\_search : point {

bool up;

bool right;

bool down;

bool left;

};

class Task // задача

{

public:

void virtual solve\_task() = 0; // решить задачу

};

class labyrinth : public Task

{

point in;

point out;

point\_search p\_search;

stack <point\_search> path;

int count;

bool\*\* matrix;

public:

labyrinth(bool\*\* matrix\_lab, int count);

void solve\_task()override;

void set\_in(int x, int y);

void set\_out(int x, int y);

~labyrinth();

};

labyrinth::labyrinth(bool\*\* matrix\_lab, int count) {

bool\* p\_matrix\_lab = (bool\*)matrix\_lab;

this->count = count;

matrix = new bool\* [count];

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

matrix[i] = new bool[count];

}

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

for (int j = 0; j < count; j++) {

matrix[i][j] = p\_matrix\_lab[i \* count + j];

}

}

cout << "\tЛабиринт :" << endl << endl;

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

cout << "\t";

for (int j = 0; j < count; j++) {

cout << matrix[i][j] << " ";

}

cout << endl;

}

}

void labyrinth::solve\_task(){

if (in.x < 0 || in.x >= count || in.y < 0 || in.y >= count) cout << "Ошибка, точка входа выходит за границы лабиринта" << endl;

else if (out.x < 0 || out.x >= count || out.y < 0 || out.y >= count) cout << "Ошибка, точка выхода выходит за границы лабиринта" << endl;

else if (matrix[in.y][in.x]) cout << "Ошибка, в точке входа в лабиринт стена" << endl;

else if (matrix[out.y][out.x]) cout << "Ошибка, в точке выхода из лабиринта стена" << endl;

else {

cout << "Начало программы, поиск выхода из лабиринта" << endl;

cout << "Точка входа в лабиринт x, y: " << in.x << " " << in.y << endl;

cout << "Точка выхода из лабиринта x, y: " << out.x << " " << out.y << endl;

p\_search.x = in.x;

p\_search.y = in.y;

bool\*\* path\_matrix = new bool\* [count];

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

path\_matrix[i] = new bool[count];

}

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

for (int j = 0; j < count; j++) {

path\_matrix[i][j] = false;

}

}

bool first\_point = false;

bool no\_exit = false;

while (p\_search.x != out.x || p\_search.y != out.y) {

if (!path\_matrix[p\_search.y][p\_search.x]) path\_matrix[p\_search.y][p\_search.x] = true;

p\_search.up = false;

p\_search.right = false;

p\_search.down = false;

p\_search.left = false;

if (p\_search.y > 0) {

if (!matrix[p\_search.y - 1][p\_search.x] && !path\_matrix[p\_search.y - 1][p\_search.x]) {

p\_search.up = true;

}

}

if (p\_search.x <= count - 2) {

if (!matrix[p\_search.y][p\_search.x + 1] && !path\_matrix[p\_search.y][p\_search.x + 1]) {

p\_search.right = true;

}

}

if (p\_search.y <= count - 2) {

if (!matrix[p\_search.y + 1][p\_search.x] && !path\_matrix[p\_search.y + 1][p\_search.x]) {

p\_search.down = true;

}

}

if (p\_search.x > 0) {

if (!matrix[p\_search.y][p\_search.x - 1] && !path\_matrix[p\_search.y][p\_search.x - 1]) {

p\_search.left = true;

}

}

if (!first\_point) {

path.push(p\_search);

first\_point = true;

}

if (p\_search.up) {// если можно пойти наверх

p\_search.y--;

path.push(p\_search);

continue;

}

if (p\_search.right) {// если можно пойти направо

p\_search.x++;

path.push(p\_search);

continue;

}

if (p\_search.down) {// если можно пойти вниз

p\_search.y++;

path.push(p\_search);

continue;

}

if (p\_search.left) {// если можно пойти налево

p\_search.x--;

path.push(p\_search);

continue;

}

if (p\_search.x == in.x && p\_search.y == in.y && !p\_search.up && !p\_search.right && !p\_search.down && !p\_search.left) {

cout << "Из лабиринта нет выхода!";

no\_exit = true;

break;

}

path.pop();

p\_search = path.top();

}

if (!no\_exit) { // если выход найден

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

for (int j = 0; j < count; j++) {

path\_matrix[i][j] = false;

}

}

auto deq = path.\_Get\_container();

int i = 0;

do {

path\_matrix[deq[i].y][deq[i].x] = true;

i++;

} while (deq[i].x != out.x || deq[i].y != out.y);

path\_matrix[deq[i].y][deq[i].x] = true;

cout << "\tПуть лабиринта:" << endl;

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

cout << "\t";

for (int j = 0; j < count; j++) {

if (path\_matrix[i][j]) cout << "\* ";

else cout << " ";

}

cout << endl;

}

i = 0;

cout << endl << "\tКоординаты точек пути:" << endl;

cout << "\t{x, y}" << endl;

do {

cout << "\t{" << deq[i].x << ", " << deq[i].y << "}" << endl;

i++;

} while (deq[i].x != out.x || deq[i].y != out.y);

cout << "\t{" << deq[i].x << ", " << deq[i].y << "}" << endl;

}

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

delete[] path\_matrix[i];

}

delete[] path\_matrix;

}

}

void labyrinth::set\_in(int x, int y) {

in.x = x;

in.y = y;

}

void labyrinth::set\_out(int x, int y) {

out.x = x;

out.y = y;

}

labyrinth::~labyrinth(){

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

delete[] matrix[i];

}

delete[] matrix;

}

class user

{

public:

void To\_Do\_Working(Task& job)

{

job.solve\_task();

}

};

int main()

{

setlocale(LC\_ALL, "ru");

srand(time(NULL));

const int size\_matrix = 20;

bool matrix[20][20] = // вход 0 3, выход 17 19

{

{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1},

{1,0,0,0,1,1,0,1,1,1,0,1,1,1,0,0,0,0,1,1},

{1,1,1,0,0,0,0,1,1,1,0,1,1,1,0,1,1,0,0,1},

{0,0,0,0,1,0,1,1,0,0,0,1,0,1,0,1,1,0,1,1},

{1,0,1,1,1,0,1,1,1,1,0,0,0,0,0,1,0,0,1,1},

{1,0,1,1,1,0,1,0,0,0,1,0,1,1,1,1,1,0,1,1},

{1,0,0,0,1,1,0,0,1,0,0,0,0,1,1,1,1,0,1,1},

{1,1,1,0,1,1,0,1,1,1,1,1,0,0,0,1,1,1,1,1},

{1,0,0,0,1,1,0,0,0,1,1,1,1,1,1,1,1,1,1,1},

{1,0,1,1,1,1,0,1,0,0,0,1,1,1,1,0,1,0,0,1},

{1,0,0,0,0,1,0,1,1,1,0,0,0,1,0,0,0,0,1,1},

{1,1,1,0,1,1,0,1,1,1,1,1,0,1,1,0,1,1,1,1},

{1,1,1,0,0,0,0,1,1,1,0,0,0,0,0,0,1,1,1,1},

{0,1,0,0,0,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1},

{0,1,0,1,1,1,1,0,0,0,0,0,0,0,1,1,1,0,1,1},

{0,1,0,0,1,1,1,0,1,1,1,1,0,1,1,1,0,0,1,1},

{1,1,1,0,1,1,1,1,1,1,1,1,0,0,0,0,0,1,1,1},

{1,0,0,0,0,1,1,1,0,0,0,1,1,1,0,1,1,1,1,1},

{1,0,1,1,0,0,0,0,0,1,0,0,1,1,0,0,0,0,1,1},

{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1}

};

labyrinth lab((bool\*\*)matrix, size\_matrix);

user user;

cout << endl << "Введите координаты входа в лабиринт x, y:" << endl;

int x, y;

cin >> x >> y;

lab.set\_in(x, y);

cout << endl << "Введите координаты выхода из лабиринта x, y:" << endl;

cin >> x >> y;

lab.set\_out(x, y);

user.To\_Do\_Working(lab);

system("pause");

}

Скриншоты с результатами

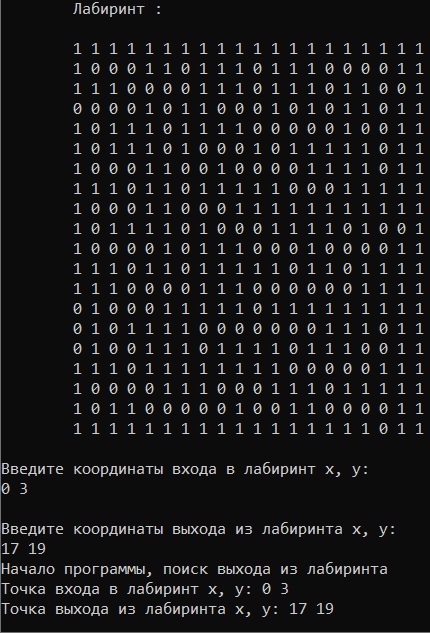


Рисунок 2

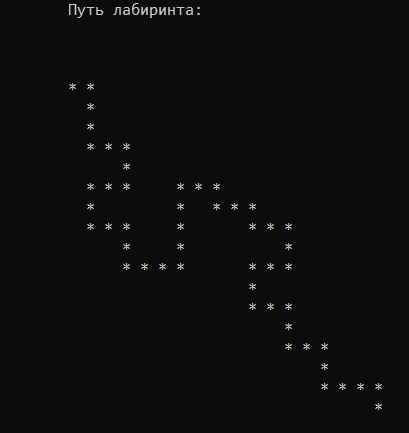


Рисунок 3

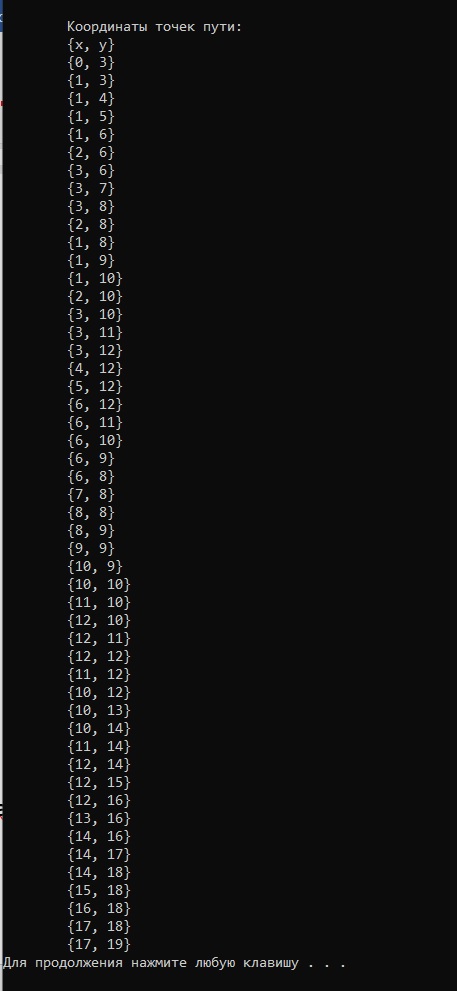


Рисунок 4

Вывод

Во время выполнения лабораторной работы был приобретен практический навык внедрения шаблонов объектно-ориентированного программирования в код программных систем. Были использованы контейнеры STL. При желании можно расширить функционал программы, а благодаря использованию шаблонного класса нет необходимости переписывать уже существующий код, что очень удобно.

К достоинствам программы можно отнести:

• Программа выполняет поставленную задачу и работает без ошибок

• Задание реализовано с использованием шаблонного класса, что позволяет использовать его в других проектах.

Недостатков обнаружено не было.