Федеральное государственное бюджетное

образовательное учреждение высшего образования

«Саратовский государственный технический университет

имени Гагарина Ю.А.»

Институт прикладных информационных

технологий и коммуникаций

Кафедра Информационная безопасность

автоматизированных систем

Направление подготовки 10.03.01 Информационная безопасность

Расчётно-графическая работа

по дисциплине «Языки программирования»

Лабиринт, поиск выхода

Выполнил: студент 1 курса

учебной группы б-ИФБС11

очной формы обучения

Немальцев Данила Андреевич

Проверил: ассистент каф. ИБС

Романчук С. П.

Саратов 2020

Аннотация

Дана карта лабиринта (файл, структура которого имеет примерно следующий вид):

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 2 1 1 1 1 1 1 0 1 1 1 0 1 1 1 0

0 1 1 0 0 0 0 0 0 1 1 1 1 1 1 1 0

0 1 0 0 1 0 1 0 0 1 1 1 1 1 1 1 0

0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0

0 1 0 1 0 0 1 1 0 0 1 1 1 1 0 0 0

0 1 0 1 1 1 1 1 0 1 1 1 0 1 1 3 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Здесь обозначены:

0 – стена

1 – проход

2 – начальное положение игрока

3 – выход

Написать программу, которая строит на экране изображение лабиринта и определяет кратчайший маршрут от начального положения пользователя к выходу.

Содержание

Введение………………………………………………………………………………………….4

Теоретическая часть……………………………………………………………………………..5

Практическая часть……………………………………………………………………………...9

Заключение……………………………………………………………………………………...11

Приложения……………………………………………………………………………………..12

Литература………………………………………………………………………………………16

Введение

Целью работы является разработка приложения на языке С++. Тема моей работы: Лабиринт, поиск выхода. Дана карта лабиринта. Необходимо написать программу, которая строит на экране изображение лабиринта и определяет кратчайший маршрут от начального положения пользователя к выходу. Компьютер должен самостоятельно найти этот самый маршрут; если маршрута нет вовсе, то компьютер должен вывести соответствующее сообщение на экран.

Теоретическая часть

При разработке приложения был использован язык С++.

С++ это компилируемый, статически типизированный язык программирования общего назначения.

Поддерживает такие парадигмы программирования, как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование. Язык имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков. В сравнении с его предшественником — языком C, — наибольшее внимание уделено поддержке объектно-ориентированного и обобщённого программирования.

C++ широко используется для разработки программного обеспечения, являясь одним из самых популярных языков программирования. Область его применения включает создание операционных систем, разнообразных прикладных программ, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также игр. Существует множество реализаций языка C++, как бесплатных, так и коммерческих и для различных платформ. Например, на платформе x86 это GCC, Visual C++, Intel C++ Compiler, Embarcadero (Borland) C++ Builder и другие. C++ оказал огромное влияние на другие языки программирования, в первую очередь на Java,C#, Golang, Solidity, PHP.

Синтаксис C++ унаследован от языка C. Одним из принципов разработки было сохранение совместимости с C. Тем не менее, C++ не является в строгом смысле надмножеством C; множество программ, которые могут одинаково успешно транслироваться как компиляторами C, так и компиляторами C++, довольно велико, но не включает все возможные программы на C.

Стандарт C++ состоит из двух основных частей: описание ядра языка и описание стандартной библиотеки.

Первое время язык развивался вне формальных рамок, спонтанно, по мере встававших перед ним задач. Развитию языка сопутствовало развитие кросс-компилятора cfront. Новшества в языке отражались в изменении номера версии кросс-компилятора. Эти номера версий кросс-компилятора распространялись и на сам язык, но применительно к настоящему вр емени речь о версиях языка C++ не ведут. Лишь в 1998 году язык стал стандартизированным.

Общие направления развития C++

По мнению автора языка Бьёрна Страуструпа, говоря о дальнейшем развитии и перспективах языка, можно выделить следующее:

• В основном дальнейшее развитие языка будет идти по пути внесения дополнений в стандартную библиотеку. Одним из основных источников этих дополнений является известная библиотека boost.

• Изменения в ядре языка не должны приводить к снижению уже достигнутой эффективности C++. С точки зрения Страуструпа, предпочтительнее внесение в ядро нескольких серьёзных больших изменений, чем множества мелких правок.

• Базовыми направлениями развития C++ на ближайшее время является расширение возможностей и доработка средств обобщённого программирования, стандартизация механизмов параллельной обработки, а также доработка средств безопасного программирования, таких как различные проверки и безопасные преобразования типов, проверка условий и так далее.

• В целом C++ спроектирован и развивается как мультипарадигменный язык, впитывающий в себя различные методы и технологии программирования, но реализующий их на платформе, обеспечивающей высокую техническую эффективность. Поэтому в будущем не исключено добавление в язык средств функционального программирования, автоматической сборки мусора и других отсутствующих в нём сейчас механизмов. Но в любом случае это будет делаться на имеющейся платформе высокоэффективного компилируемого языка.

• Хотя формально одним из принципов C++ остаётся сохранение совместимости с языком C, фактически группы по стандартизации этих языков не взаимодействуют, а вносимые ими изменения не только не коррелируют, но и нередко принципиально противоречат друг другу идеологически. Так, элементы, которые новые стандарты C добавляют в ядро, в стандарте C++ являются элементами стандартной библиотеки и в ядре вообще отсутствуют, например, динамические массивы, массивы с фиксированными границами, средства параллельной обработки. Как считает Страуструп, объединение разработки этих двух языков принесло бы большую пользу, но оно вряд ли возможно по политическим соображениям. Так что практическая совместимость между C и C++ постепенно будет утрачиваться.

Достоинства

C++ содержит средства разработки программ контролируемой эффективности для широкого спектра задач, от низкоуровневых утилит и драйверов до весьма сложных программных комплексов. В частности:

• Высокая совместимость с языком Си : код на Си может быть с минимальными переделками скомпилирован компилятором C++. Внешнеязыковой интерфейс является прозрачным, так что библиотеки на Си могут вызываться из C++ без дополнительных затрат, и более того — при определённых ограничениях код на C++ может экспортироваться внешне не отличимо от кода на Си (конструкция extern "C").

• Как следствие предыдущего пункта — вычислительная производительность. Язык спроектирован так, чтобы дать программисту максимальный контроль над всеми аспектами структуры и порядка исполнения программы. Один из базовых принципов C++ — «не платишь за то, что не используешь» (см. Философия C++) — то есть ни одна из языковых возможностей, приводящая к дополнительным накладным расходам, не является обязательной для использования. Имеется возможность работы с памятью на низком уровне.

• Поддержка различных стилей программирования: традиционное императивное программирование (структурное, объектно-ориентированное), обобщённое программирование, функциональное программирование, порождающее метапрограммирование.

• Автоматический вызов деструкторов объектов в адекватном порядке (обратном вызову конструкторов) упрощает и повышает надёжность управления памятью и другими ресурсами (открытыми файлами, сетевыми соединениями, соединениями с базами данных и т. п.).

• Перегрузка операторов позволяет кратко и ёмко записывать выражения над пользовательскими типами в естественной алгебраической форме.

• Имеется возможность управления константностью объектов (модификаторы const, mutable, volatile). Использование константных объектов повышает надёжность и служит подсказкой для оптимизации. Перегрузка функций-членов по признаку константности позволяет определять выбор метода в зависимости цели вызова (константный для чтения, неконстантный для изменения). Объявление mutable позволяет сохранять логическую константность при виде извне кода, использующего кэши и ленивые вычисления.

• Шаблоны C++ дают возможность построения обобщённых контейнеров и алгоритмов для разных типов данных. Попутно шаблоны дают возможность производить вычисления на этапе компиляции.

• Возможность встраивания предметно-ориентированных языков программирования в основной код. Такой подход использует, например библиотека Boost.Spirit, позволяющая задавать EBNF-грамматику парсеров прямо в коде C++.

• Доступность. Для C++ существует огромное количество учебной литературы, переведённой на всевозможные языки. Язык имеет высокий порог вхождения, но среди всех языков такого рода обладает наиболее широкими возможностями.

Недостатки

К числу обычно упоминаемых недостатков языка можно отнести:

• Отсутствие системы модулей. C++ унаследовал от Си подключение заголовочных файлов с помощью препроцессора. Это вынуждает дублировать описания объектов, порождает неочевидные требования к коду и увеличивает объём компилируемого текста, а значит и время компиляции.

• Наличие более чем одного механизма для выполнения одних и тех же задач, что усложняет язык и приводит к неоптимальному и небезопасному кодированию.

• Унаследованные от Си опасные и провоцирующие ошибки возможности (макроопределения, адресная арифметика, неявное приведение типов, возможность прямого управления распределением памяти).

• Отсутствие встроенных механизмов статической валидации времени жизни объектов, приводящее к внезапному краху программ из-за обращения к уничтоженной переменной, или из-за неправильной многопоточной работы с объектами.

• Шаблоны порождают объёмный и не всегда оптимальный код. Частичное определение шаблонов усложняет как сам язык, так и программы, где оно используется.

• Множественное (в том числе виртуальное) наследование приводит к созданию громоздких иерархий классов, которые при любом изменении требований к программе могут потребовать серьёзного пересмотра.

• Сложный синтаксис и объёмная спецификация языка затрудняют его изучение.

• Язык не поощряет создание надёжного, легко читаемого и удобного в сопровождении кода, вместо этого зачастую предлагая выбор между короткими и простыми, но опасными средствами, унаследованными от Си, и новыми, объёмными и сложными, но более безопасными механизмами.

• Сложная и постоянно разрастающаяся стандартная библиотека, затрудняющая изучение языка.

Практическая часть

Для выполнения поставленной задачи в приложении используются библиотеки: <iostresm>, <fstream>, <windows.h>.

<iostream> — заголовочный файл с классами, функциями и переменными для организации ввода-вывода в языке программирования C++. Он включён в стандартную библиотеку C++. Название образовано от Input/Output Stream («поток ввода-вывода»). В языке C++ и его предшественнике, языке программирования Си, нет встроенной поддержки ввода-вывода, вместо этого используется библиотека функций. iostream управляет вводом-выводом, как и stdio.h в Си. iostream использует объекты cin, cout, cerr и clog для передачи информации и из стандартных потоков ввода, вывода, ошибок без буферизации и ошибок с буферизацией соответственно. Являясь частью стандартной библиотеки C++, эти объекты также являются частью стандартного пространства имён — std.

<fstream> (сокращение от «FileStream») — заголовочный файл из стандартной библиотеки C++, включающий набор классов, методов и функций, которые предоставляют интерфейс для чтения/записи данных из/в файл. Для манипуляции с данными файлов используются объекты, называемые потоками («stream»).

<windows.h> открывает доступ к WinAPI для изменения примитивного консольного интерфейса

В основе приложения лежит алгоритм волновой трассировки (он же алгоритм Ли)

Алгоритм Ли – это алгоритм поиска пути, алгоритм поиска кратчайшего пути на планарном графе. Принадлежит к алгоритмам, основанным на методах поиска в ширину.

В основном используется при компьютерной трассировке (разводке) печатных плат, соединительных проводников на поверхности микросхем. Другое применение волнового алгоритма — поиск кратчайшего расстояния на карте в компьютерных стратегических играх.

Описание алгоритма

Алгоритм работает на дискретном рабочем поле (ДРП), представляющем собой ограниченную замкнутой линией фигуру, не обязательно прямоугольную, разбитую на прямоугольные ячейки, в частном случае — квадратные. Множество всех ячеек ДРП разбивается на подмножества: «проходимые» (свободные), т. е при поиске пути их можно проходить, «непроходимые» (препятствия), путь через эту ячейку запрещён, стартовая ячейка (источник) и финишная (приемник). Назначение стартовой и финишной ячеек условно, достаточно — указание пары ячеек, между которыми нужно найти кратчайший путь.

Алгоритм предназначен для поиска кратчайшего пути от стартовой ячейки к конечной ячейке, если это возможно, либо, при отсутствии пути, выдать сообщение о непроходимости.

Работа алгоритма включает в себя три этапа: инициализацию, распространение волны и восстановление пути.

Во время инициализации строится образ множества ячеек обрабатываемого поля, каждой ячейке приписываются атрибуты проходимости/непроходимости, запоминаются стартовая и финишная ячейки.

Далее, от стартовой ячейки порождается шаг в соседнюю ячейку, при этом проверяется, проходима ли она, и не принадлежит ли ранее меченной в пути ячейке.

Соседние ячейки принято классифицировать двояко: в смысле окрестности Мура и окрестности фон Неймана, отличающийся тем, что в окрестности фон Неймана соседними ячейками считаются только 4 ячейки по вертикали и горизонтали, в окрестности Мура — все 8 ячеек, включая диагональные.

При выполнении условий проходимости и непринадлежности её к ранее помеченным в пути ячейкам, в атрибут ячейки записывается число, равное количеству шагов от стартовой ячейки, от стартовой ячейки на первом шаге это будет 1. Каждая ячейка, меченная числом шагов от стартовой ячейки, становится стартовой и из неё порождаются очередные шаги в соседние ячейки. Очевидно, что при таком переборе будет найден путь от начальной ячейки к конечной, либо очередной шаг из любой порождённой в пути ячейки будет невозможен.

Восстановление кратчайшего пути происходит в обратном направлении: при выборе ячейки от финишной ячейки к стартовой на каждом шаге выбирается ячейка, имеющая атрибут расстояния от стартовой на единицу меньше текущей ячейки. Очевидно, что таким образом находится кратчайший путь между парой заданных ячеек. Трасс с минимальной числовой длиной пути, как при поиске пути в окрестностях Мура, так и фон Неймана может существовать несколько. Выбор окончательного пути в приложениях диктуется другими соображениями, находящимися вне этого алгоритма. Например, при трассировке печатных плат — минимумом линейной длины проложенного проводника.

Заключение

В результате работы были выполнены все поставленные задачи; была разработана программа, позволяющая определить кратчайший путь от начального положения к выходу в любом заданном лабиринте.

Приложения

#include <iostream>

#include <fstream>

#include <windows.h>

using namespace std;

struct LabrItem

{

char Symb;

int NumStep;

};

LabrItem Labr[256][256];

void SetAttr(WORD attr) //чтобы можно было менять атрибуты

{

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, attr);

}

void GotoXY(short X, short Y)

{

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

COORD cor = {0, 0};

cor.X = X;

cor.Y = Y;

SetConsoleCursorPosition(hConsole, cor);

}

void PrintXY(char T, short X, short Y, WORD attr)

{

GotoXY(X, Y);

SetAttr(attr);

cout << T;

}

int main()

{

setlocale(LC\_ALL, "RUS");

ifstream file; // создаем объект класса ifstream

file.open("C:\\Users\\danil\\Desktop\\labs\\sem2\\РГР (АЛЬФА)\\Лабиринт.txt"); // открываем файл

int row = 0, col;

while (!file.eof())

{

char str[256];

file.getline(&str[0], 256, '\n');

col = 0;

while (str[col] != 0)

{

Labr[row][col].Symb = str[col];

Labr[row][col].NumStep = INT\_MAX;

++col;

}

++row;

}

file.close();

// Вывод лабиринта на экран, поиск начального положения игрока и выхода

int RowBegin, ColBegin;

int RowEnd, ColEnd;

for (row = 0; row < 256; row++)

{

if (Labr[row][0].Symb == 0) break;

for (col = 0; col < 256; col++)

{

if (Labr[row][col].Symb != 0)

{

std::cout << Labr[row][col].Symb;

if (Labr[row][col].Symb == '2')

{

RowBegin = row;

ColBegin = col;

}

else if (Labr[row][col].Symb == '3')

{

RowEnd = row;

ColEnd = col;

Labr[row][col].NumStep = 0;

}

}

}

std::cout << '\n';

}

//АЛГОРИТМ ЛИ

int d = 0;

bool FindBegin = false;

bool WasWave = true;

while (!FindBegin && WasWave)

{

WasWave = false;

for (row = 0; row < 256; row++)

{

for (col = 0; col < 256; col++)

{

if (Labr[row][col].Symb != 0 && Labr[row][col].NumStep == d)

{

if (Labr[row][col - 1].NumStep == INT\_MAX && col > 0 && (Labr[row][col - 1].Symb == '1' || Labr[row][col - 1].Symb == '2'))

{

Labr[row][col - 1].NumStep = d + 1;

WasWave = true;

if (Labr[row][col - 1].Symb == '2') FindBegin = true;

}

if (Labr[row - 1][col].NumStep == INT\_MAX && row > 0 && (Labr[row - 1][col].Symb == '1' || Labr[row - 1][col].Symb == '2'))

{

Labr[row - 1][col].NumStep = d + 1;

WasWave = true;

if (Labr[row - 1][col].Symb == '2') FindBegin = true;

}

if (Labr[row][col + 1].NumStep == INT\_MAX && col < 255 && (Labr[row][col + 1].Symb == '1' || Labr[row][col + 1].Symb == '2'))

{

Labr[row][col + 1].NumStep = d + 1;

WasWave = true;

if (Labr[row][col + 1].Symb == '2') FindBegin = true;

}

if (Labr[row + 1][col].NumStep == INT\_MAX && row < 255 && (Labr[row + 1][col].Symb == '1' || Labr[row + 1][col].Symb == '2'))

{

Labr[row + 1][col].NumStep = d + 1;

WasWave = true;

if (Labr[row + 1][col].Symb == '2') FindBegin = true;

}

}

}

}

++d;

}

if (FindBegin)

{

col = ColBegin;

row = RowBegin;

GotoXY(ColBegin, RowBegin);

PrintXY('2', col, row, (WORD)((0 << 4) | 14));

do

{

Sleep(50);

if (col > 0 && Labr[row][col - 1].NumStep == Labr[row][col].NumStep - 1) --col;

else if (row > 0 && Labr[row - 1][col].NumStep == Labr[row][col].NumStep - 1) --row;

else if (col < 255 && Labr[row][col + 1].NumStep == Labr[row][col].NumStep - 1) ++col;

else if (row < 255 && Labr[row + 1][col].NumStep == Labr[row][col].NumStep - 1) ++row;

if(Labr[row][col].Symb != '3') PrintXY('¤'/\*'1'\*/, col, row, (WORD)((0 << 4) | 14));

else PrintXY('3', col, row, (WORD)((0 << 4) | 14));

}

while (col != ColEnd || row != RowEnd);

}

else std::cout << "\n\nПрохода к выходу нет!\n";

GotoXY(0, 20);

SetAttr((WORD)((0 << 4) | 7));

}

Литература

1. <https://ru.wikipedia.org/wiki/Алгоритм_Ли>
2. https://ru.wikipedia.org/wiki/C%2B%2B