Автор: Андрей
02.04.2009 22:23 - Обновлено 20.04.2009 20:57

Описание программы

В этой программе по полю, поделенному на клетки, перемещается некий "персонаж", изображаемый кружочком. Управление осуществляется стрелками, выход происходит по нажатию клавиши "Escape". "Персонаж" может перемещаться только по свободным клеткам поля. Он представлен классом, включающим в себя его координаты, размер, конструктор для определения начальных данных, функцию рисования, стирания и функцию движения, которой в качестве аргументов передается двумерный массив клеток поля и направление движения.

Код программы

```
    #include "graphics.h"
#include "stdlib.h"
#include "stdio.h"
#include "conio.h"

 enum Direction {LEFT, UP, RIGHT, DOWN};

 const int Width = 15;
const int Height = 15;
 const int CellSize = 30;

 int Cell[Width][Height]={
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},
{1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1},
{1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1},
{1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1},
{1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1},
{1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1},
{0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0},
{1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1},
{1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1},
{1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1},
{1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1},
```

```
 {1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1},
 {1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1},
 {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
 };

   void DrawField()
   {
   setcolor(WHITE);
   rectangle(0, 0, 30 * Width, 30 * Height);
   for (int i = 1; i                 line(i * CellSize, 1, i * CellSize, Height * CellSize);
   for (i = 1; i  0) && Cell[X - 1][Y] != 1) X--; break;
    case UP: if (Y > 0 && Cell[X][Y - 1] != 1) Y--; break;
    case RIGHT: if (X          case DOWN: if (Y          }
   Draw();
   }

   void main()
 {
GraphInit();

   int exitOk = 0;
  char c;

   DrawField();

   CharacterClass Character;
  Character.Draw();

   while (!exitOk)
   {
   c = getch();
   switch (c)
    {
    case 27: exitOk = !0;
    case 75: Character.Move(Cell, LEFT); break;
    case 72: Character.Move(Cell, UP); break;
    case 77: Character.Move(Cell, RIGHT); break;
    case 80: Character.Move(Cell, DOWN); break;
    }
   DrawField();
   }
  closegraph();
  }
```

Код программы с комментариями

```
    #include <graphics.h> //Подключаем графическую библиотеку
#include <stdlib.h> //Библиотека содержит функцию exit
//(см. GraphInit)
#include <stdio.h> //Библиотека содержит функцию printf
//(см. GraphInit)
#include <conio.h> //Библиотека содержит функцию getch

 enum Direction {LEFT, UP, RIGHT, DOWN};
//Перечисляемый тип - направления движения

 const int Width = 15; //Ширина поля в клетках
const int Height = 15; //Высота поля в клетках
 const int CellSize = 30; //Размер клетки в пикселах

 int Cell[Width][Height]={ //Задаем массив клеток поля
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},
{1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1},
{1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1},
{1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1},
{1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1},
{1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1},
{0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0},
{1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1},
{1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1},
{1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1},
{1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1},
{1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1},
{1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1},
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
}; //Ноль - клетка свободна, единица - клетка занята

 void DrawField() //Функция, рисующая поле
 {
 setcolor(WHITE); //Устанавливаем цвет линий
 rectangle(0, 0, 30 * Width, 30 * Height); //Внешняя граница поля
 for (int i = 1; i               //Рисуем вертикальные линии сетки
  line(i * CellSize, 1, i * CellSize, Height * CellSize);
 for (i = 1; i  0) && Cell[X - 1][Y] != 1) X--; break;
  case UP: if (Y > 0 && Cell[X][Y - 1] != 1) Y--; break;
  case RIGHT: if (X          case DOWN: if (Y          }
 Draw(); //Рисуем в новом положении
 }

 void main()
```

```
 {
GraphInit();   //Текст функции расположен по ссылке.
 //Функция инициализирует графику

  int exitOk = 0; //Флаг, означающий выход из программы
 char c; //Считываемый с клавиатуры символ

  DrawField(); //Сначала рисуем поле

  CharacterClass Character;
 Character.Draw(); //Рисуем "персонаж"

  while (!exitOk)
  {
  c = getch(); //Считываем с клавиатуры символ
  switch (c) //Если была нажата клавиша-стрелка,..
   { //...перемещаем "персонаж"
   case 27: exitOk = !0; //Выход по "Escape"
   case 75: Character.Move(Cell, LEFT); break;
   case 72: Character.Move(Cell, UP); break;
   case 77: Character.Move(Cell, RIGHT); break;
   case 80: Character.Move(Cell, DOWN); break;
   }
  DrawField(); //После перемещения вновь рисуем поле
  }
 closegraph();
 }
```

Советы по улучшению и расширению программы

Данную программу можно рассматривать как простейшую игру по прохождению лабиринта - при соответствующем задании поля оно и будет этим самым лабиринтом. Однако предварительное задание поля в коде программы в определенном смысле неудобно. В качестве альтернативы можно включить в программу редактор уровня. Для этого я рекомендую:

- Создать для поля отдельный класс (можно также собрать все данные и функции в один модуль с помощью пространства имен, которого не было в Borland C++ 3.1, если вы адаптировали программу для более современной среды разработки).
- Помимо уже используемых функций добавить функцию Edit, которая действовала бы аналогично функции Move, только без проверки, пуста ли клетка-цель, а при нажатии, например, клавиши "Enter" меняла бы состояние текущей клетки "занята"-"свободна" на противоположное.
- В качестве аргумента функции Move из класса CharacterClass во избежание излишнего копирования данных передавать ссылку на объект типа FieldClass:

```
    FieldClass& refField = Field;
  Move(refField, LEFT);
```

Кроме того, можно заставить нашего "персонажа" самостоятельно обходить лабиринт, используя простейший алгоритм - всегда сворачивать налево (направо). Т.е. если можно - идти налево (направо), иначе - прямо, иначе - направо (налево), иначе - назад. При этом необходимо будет добавить переменную, хранящую направление, в котором он двигался только что, а текущее направление будет определяться сдвигом этого значения на нужное число единиц.

Можно также добавить периодически появляющиеся на свободных участках поля объекты, сбор которых приносит очки. Для этого клетки поля должны будут принимать не только значения "свободна" и "занята", но и "содержит бонус", который также надо будет рисовать. И, конечно, нужно будет добавить в класс "персонажа" счетчик очков.

Собирать бонусные очки можно не только в одиночку, но и наперегонки с кем-то еще. Для этого достаточно будет добавить второй объект класса CharacterClass, и движение его осуществлять, например, по нажатию клавиш "W", "A", "S", "D". Конечно, он должен визуально отличаться от первого - например, цветом, который можно передавать в качестве аргумента конструктору.

На такой плодотворной почве можно придумать и множество других расширений программы, как графических, например, анимацию движения, так и функциональных.