

Основы программирования

ФИСТ 1 курс

Власенко

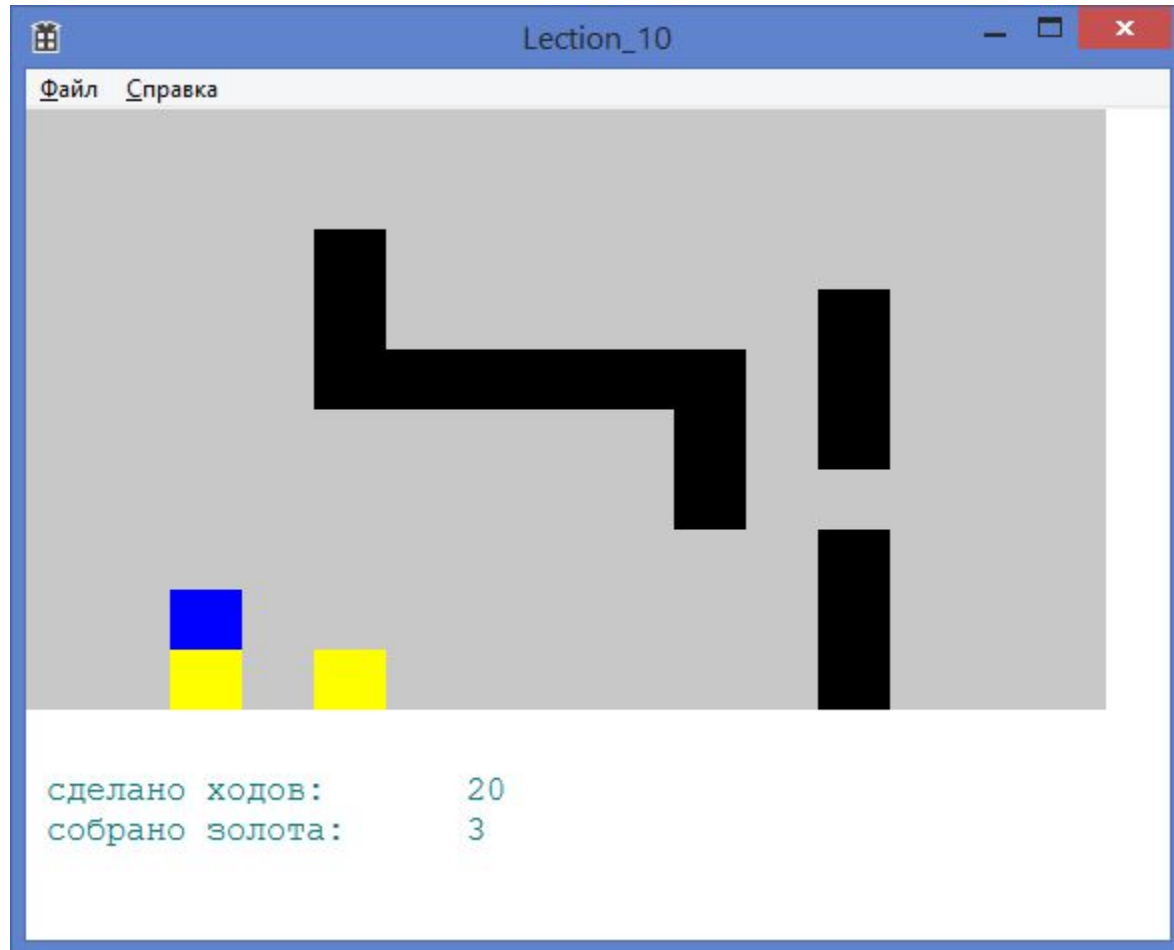
Олег

Федосович

Лекция 9.2

Простая игра на двумерном массиве

Делаем игру на основе 2D массива



Кодируем состояние игры в 2D массиве

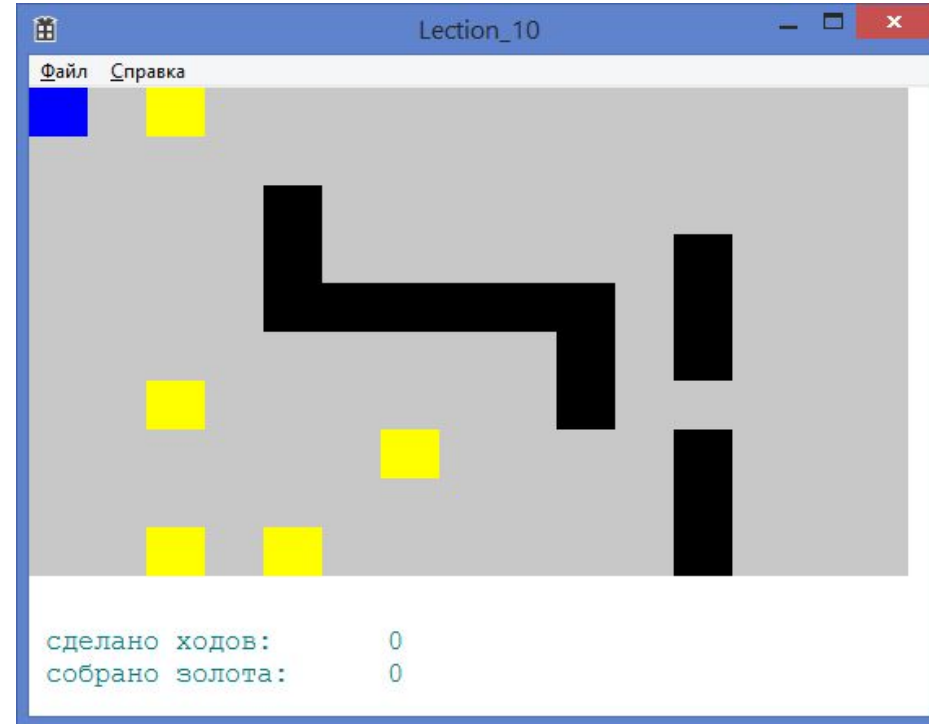
```
#define N 10
```

```
#define M 15
```

```
int a[N][M] = {  
{ 3, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },  
{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },  
{ 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },  
{ 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0 },  
{ 0, 0, 0, 0, 2, 2, 2, 2, 2, 2, 0, 2, 0, 0, 0 },  
  
{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 0, 0, 0 },  
{ 0, 0, 1, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0 },  
{ 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 2, 0, 0, 0 },  
{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0 },  
{ 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0 }  
};
```

```
int steps = 0;
```

```
int gold = 0;
```



Коды ячеек

// 0 - ???

// 1 - ???

// 2 - ???

// 3 - ???

Кодируем состояние игры в 2D массиве

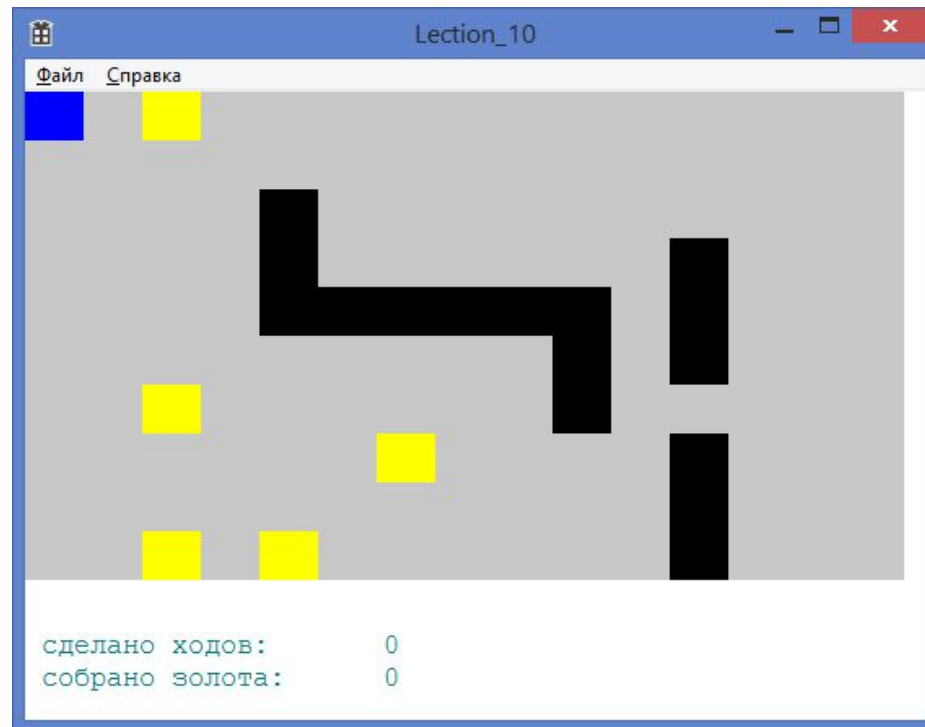
```
#define N 10
```

```
#define M 15
```

```
int a[N][M] = {  
{ 3, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },  
{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },  
{ 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },  
{ 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0 },  
{ 0, 0, 0, 0, 2, 2, 2, 2, 2, 0, 2, 0, 0, 0, 0 },  
  
{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 0, 0, 0 },  
{ 0, 0, 1, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0 },  
{ 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 2, 0, 0, 0 },  
{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0 },  
{ 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0 }  
};
```

```
int steps = 0;
```

```
int gold = 0;
```



Коды ячеек

// 0 - свободно

// 1 - золото

// 2 - стена

// 3 - игрок

Код функции WndProc

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam,
LPARAM lParam)
{
    switch (message)
    {

    case WM_PAINT:
    {
        PAINTSTRUCT ps;
        HDC hdc = BeginPaint(hWnd, &ps);

        DrawField(hdc);

        EndPaint(hWnd, &ps);
    }
    break;
```

Код функции WndProc

```
case WM_KEYDOWN:  
    switch (wParam)  
    {  
    case VK_DOWN:  
        moveDown();  
        InvalidateRect(hWnd, NULL, TRUE);  
        break;  
    case VK_LEFT:  
        moveToLeft();  
        InvalidateRect(hWnd, NULL, TRUE);  
        break;  
    case VK_UP:  
        moveUp();  
        InvalidateRect(hWnd, NULL, TRUE);  
        break;  
    case VK_RIGHT:  
        moveToRight();  
        InvalidateRect(hWnd, NULL, TRUE);  
        break;  
    }  
    break;
```

Изменение состояния игры: двигаем игрока

ВЛЕВО

```
void moveToLeft() {
    int i, j;
    i = 0;
    while (i < N) {
        j = 1;
        while (j < M) {
            if (a[i][j] == 3) {
                if (a[i][j - 1] == 0) {
                    a[i][j - 1] = 3;
                    a[i][j] = 0;
                    steps++;
                } else if (a[i][j - 1] == 1) {
                    a[i][j - 1] = 3;
                    a[i][j] = 0;
                    steps++;
                    gold++;
                }
            }
            j++;
        }
        i++;
    }
}
```

Изменение состояния игры: двигаем игрока вправо

```
void moveToRight() {
    int i = 0;
    while (i < N) {
        int j = M - 2;
        while (j >= 0) {
            if (a[i][j] == 3) {
                if (a[i][j + 1] == 0) {
                    a[i][j + 1] = 3;
                    a[i][j] = 0;
                    steps++;
                } else if (a[i][j + 1] == 1) {
                    a[i][j + 1] = 3;
                    a[i][j] = 0;
                    steps++;
                    gold++;
                }
            }
            j--;
        }
        i++;
    }
}
```


Изменение состояния игры: двигаем игрока вверх

```
void moveUp() {
    int i = 1;
    while (i < N) {
        int j = 0;
        while (j < M) {
            if (a[i][j] == 3) {
                if (a[i - 1][j] == 0) {
                    a[i - 1][j] = 3;
                    a[i][j] = 0;
                    steps++;
                } else if (a[i - 1][j] == 1) {
                    a[i - 1][j] = 3;
                    a[i][j] = 0;
                    steps++;
                    gold++;
                }
            }
            j++;
        }
        i++;
    }
}
```

Изменение состояния игры: двигаем игрока вниз

```
void moveDown() {
    int i = N;
    while (i >= 0) {
        int j = 0;
        while (j < M) {
            if (a[i][j] == 3) {
                if (a[i + 1][j] == 0) {
                    a[i + 1][j] = 3;
                    a[i][j] = 0;
                    steps++;
                } else if (a[i + 1][j] == 1) {
                    a[i + 1][j] = 3;
                    a[i][j] = 0;
                    steps++;
                    gold++;
                }
            }
            j++;
        }
        i--;
    }
}
```

Отрисовка состояния игры

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam,
LPARAM lParam)
{
    switch (message)
    {
        ...
        case WM_PAINT:
        {
            PAINTSTRUCT ps;
            HDC hdc = BeginPaint(hWnd, &ps);

            DrawField(hdc);

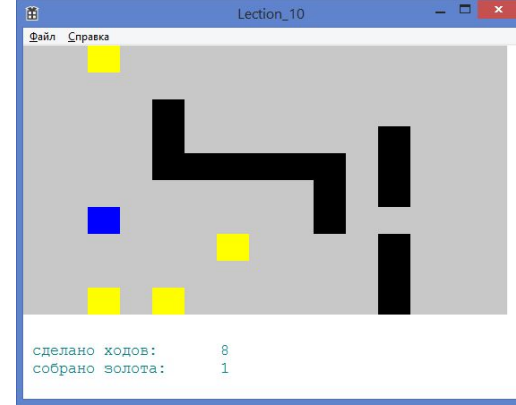
            EndPaint(hWnd, &ps);
        }
        break;
        ...
        return 0;
    }
}
```

Отрисовка состояния игры (2)

```
int sizeX = 36;
```

```
int sizeY = 30;
```

```
void DrawField(HDC hdc) {
```



```
HBRUSH hBrushEmptyCell; //создаём кисть для пустого поля  
hBrushEmptyCell = CreateSolidBrush(RGB(200, 200, 200)); // серый
```

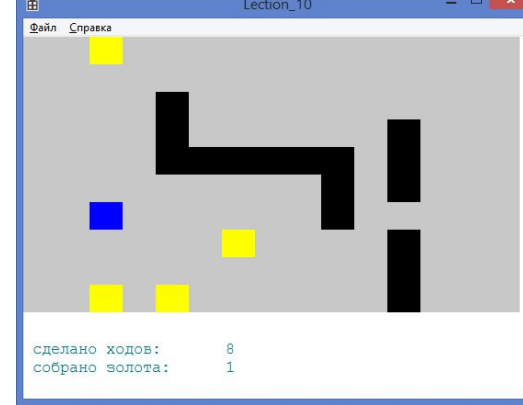
```
HBRUSH hBrushGold; //создаём кисть для поля с золотом  
hBrushGold = CreateSolidBrush(RGB(255, 255, 0)); // желтый
```

```
HBRUSH hBrushWall; //создаём кисть для стены  
hBrushWall = CreateSolidBrush(RGB(0, 0, 0)); // черный
```

```
HBRUSH hBrushMan; //создаём кисть для игрока  
hBrushMan = CreateSolidBrush(RGB(0, 0, 255)); // синий
```

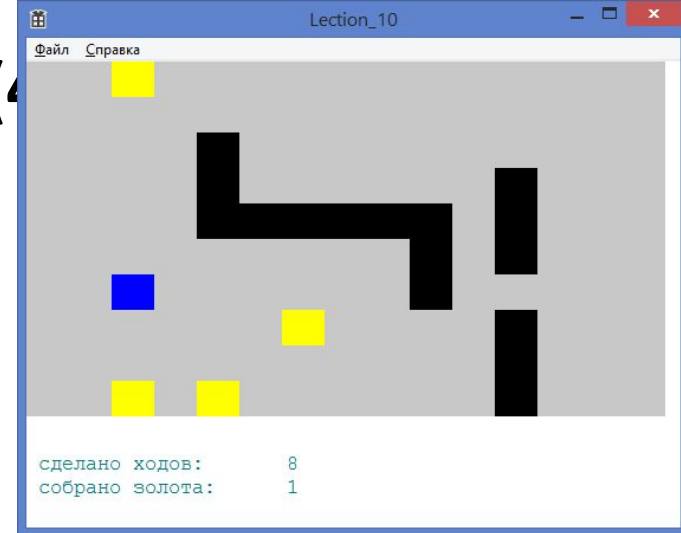
Отрисовка состояния игры (3)

```
int i, j;
i = 0;
while (i < N) {
    j = 0;
    while (j < M) {
        RECT rect = { j * sizeX, i * sizeY, (j + 1) * sizeX, (i + 1) * sizeY };
        if (a[i][j] == 0) {
            FillRect(hdc, &rect, hBrushEmptyCell);
        } else if (a[i][j] == 1) {
            FillRect(hdc, &rect, hBrushGold);
        } else if (a[i][j] == 2) {
            FillRect(hdc, &rect, hBrushWall);
        } else if (a[i][j] == 3) {
            FillRect(hdc, &rect, hBrushMan);
        } else {
            // тут никогда не должны оказаться
        }
        j++;
    }
    i++;
}
```



Отрисовка состояния игры (4

```
HFONT hFont;  
hFont = CreateFont(20,  
    0, 0, 0, 0, 0, 0, 0,  
    DEFAULT_CHARSET,  
    0, 0, 0, 0,  
    L"Courier New"  
);  
SelectObject(hdc, hFont);  
SetTextColor(hdc, RGB(0, 128, 128));  
  
TCHAR string1[] = _T("сделано ходов:");  
TCHAR string2[] = _T("собрано золота:");  
TextOut(hdc, 10, sizeY * (N + 1), (LPCWSTR)string1, _tcslen(string1));  
TextOut(hdc, 10, sizeY * (N + 1) + 20, (LPCWSTR)string2, _tcslen(string2));
```



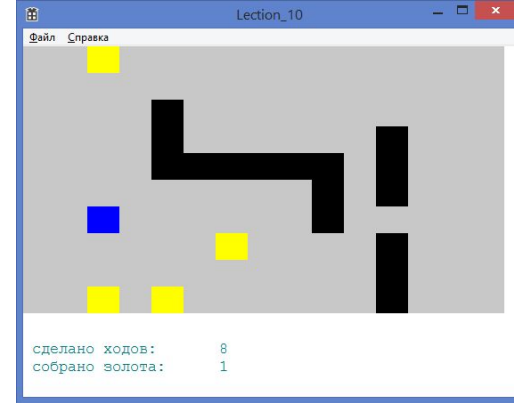
Отрисовка состояния игры (5)

```
char sSteps[5];
TCHAR tsSteps[5];
sprintf(sSteps, "%d", steps);
OemToChar(sSteps, tsSteps);
TextOut(hdc, 220, sizeY * (N + 1), (LPCWSTR)tsSteps, _tcslen(tsSteps));

char sGold[5];
TCHAR tsGold[5];
sprintf(sGold, "%d", gold);
OemToChar(sGold, tsGold);
TextOut(hdc, 220, sizeY * (N + 1) + 20, (LPCWSTR)tsGold, _tcslen(tsGold));

DeleteObject(hFont);
DeleteObject(hBrushEmptyCell);
DeleteObject(hBrushGold);
DeleteObject(hBrushWall);
DeleteObject(hBrushMan);

} // конец функции void DrawField(HDC hdc)
```



Домашнее задание

** Соберите игру из предложенного кода

*** Сделайте свою игру на основе
двухмерного массива

Источники информации

- msdn
- google