Міністерство освіти та науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет прикладної математики
Кафедра системного програмування і спеціалізованих комп’ютерних систем

Розрахунково-графічна робота
з дисципліни
“ Програмування 2. Програмування мовою С”

Виконав: Фесенко Д.О.
Студент групи КВ-34
Варіант №20

Київ 2024

Постановка задачі

Створити ігрову програму мовою програмування С.

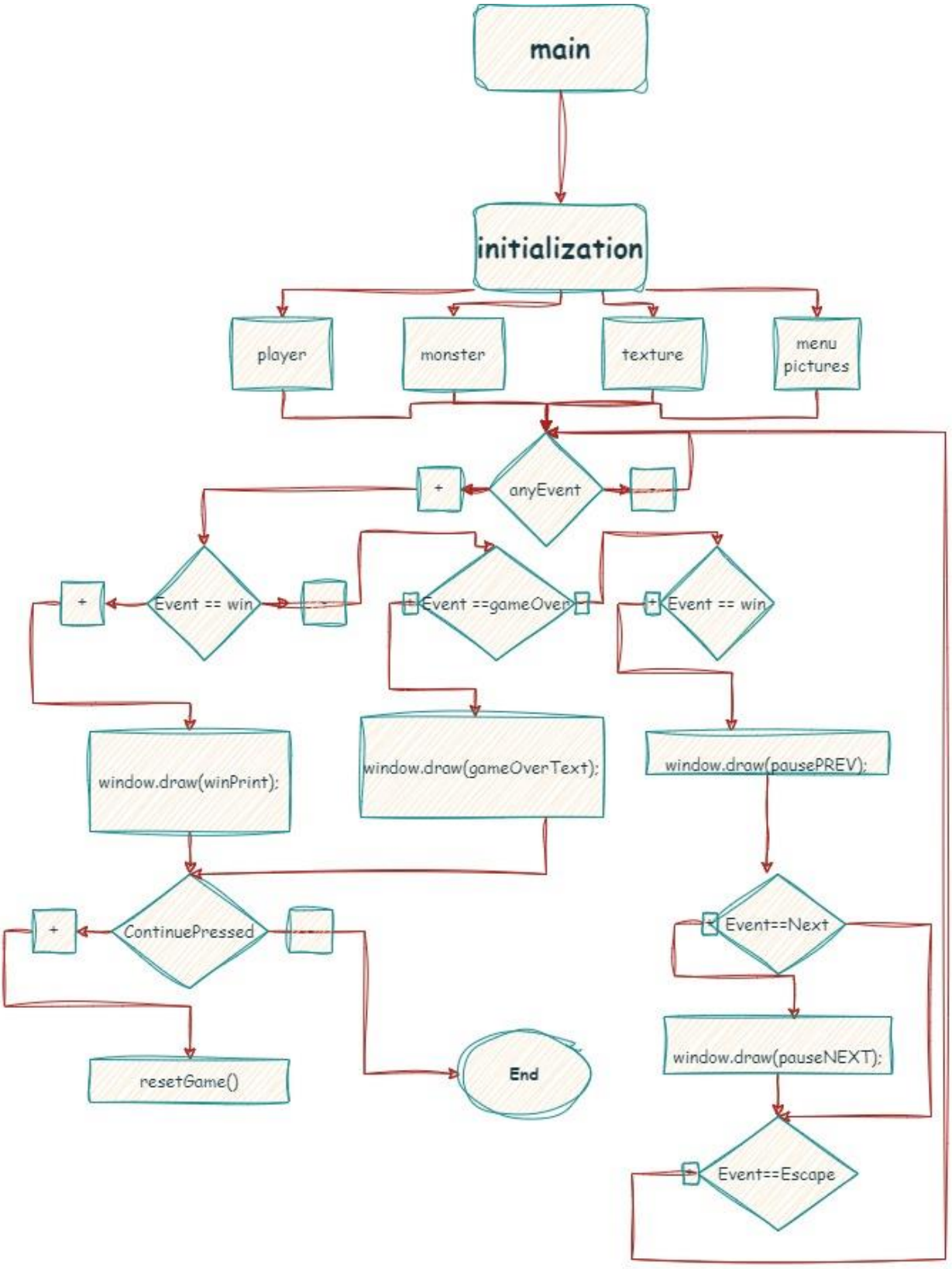
Розробка і реалізація ігрових програм має вестися з врахуванням графічних та звукових можливостей, що надаються конкретним комп’ютером.

Програма мусить коректно розв’язувати поставлену задачу. Логічно відокремлені частини алгоритма реалізувати за допомогою окремих функцій.

Також потрібно передбачити та забезпечити виконання всіх можливих розгалужень алгоритма, тобто програма повинна коректно реагувати на будь-які можливі ситуації (наприклад, виникнення помилкових ситуацій, перевірка файлів на порожність, правильність введених з клавіатури значень і т. д.). Передбачити взаємодію з користувачем (наприклад, можливість виводу правил гри, допомоги), таймер, лічильник числа ходів відповідно до поставленої в конкретному варіанті задачі.

Завдання за варіантом №20

«Лабіринт». Керування відбувається за допомогою стрілок на клавіатурі. Потрібно провести заданого персонажа через намальований лабіринт.


main
initialization
player
monster
texture
menu pictures
anyEvent
Event == win
Event ==gameOver
Event == win
window.draw(winPrint);
window.draw(gameOverText);
window.draw(pausePREV);
ContinuePressed
Event==Next
resetGame()
End
window.draw(pauseNEXT);
Event==Escape

Код програми

```
 //Your welcome...
#include <SFML/Graphics.hpp>
#include <vector>
#include <cstdlib>
#include <ctime>

using namespace sf;

const int H = 29;
const int W = 33;
const int ts = 111;

String TileMap[H] = {//E - real exit
   "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA",
   "A    A    A  A A  AA   A    A",
   "AA  A AA AA   A AAA AA AA A  AA A",
   "A A A  A  AA   A A         A A",
   "A   A AA   A A     A A A A A  A",
   "AA AA  A A  A A  AAA A  A AA A A ",
   "A   A  A A   A  A  A  A   A A  A",
   "A AA A A AA  A AA A A AA  A  A AA",
   "A A  A     A   A  AAAA  AA A   A",
   "AA  AA AA A A AAA A    A   AA AA",
   "A A    A A A   A   A  AAAA A   A",
   "A  A AA A A A AA AAAAAAA AA  AA A",
   "AA       A     A  A  AA A A   A",
   "A  AAAA AAAAAAA   A A     A A A",
   "A     AA    AA A    AA A   A A",
   "A AAAAAA A A    AA A AA   A A  A",
   "A     AA   AA  A  AAA  A   AAAA",
   "AA A A  A A A   A   A AA A   A ",
   "A AA AA   AA AA  AA A A  A AA  A",
   "A     A  A A    AA A  AA  AA AA",
   "A AAA AAA AA AA AA AA  A   AA  A ",
   "AA   A    A A  A  A A  A A   AA",
   "A  A AAA AA AA       A AAA  A  A",
   "A  A      A  A   AA   AA  AA  A",
   "AA A  A A A  A AAA  A AA    A  AA",
   "A   A   A  A     A A  AAA AAA  A",
   "A A A  A    AAAA   AA  A     AA",
   "A  A  A A  A  A AA A  AA  A A   A",
   "AAA AA   AAAAAAAAAAAAAAAEAAAAAAAA",
};

float camX = 3;
float camY = ts * -1;

bool win = false;
bool gameOver = false; // flag to indicate game over state
int escape = 2;
bool pnext = false;

// Structure to represent a monster
struct Monster {
   int x, y;
```

```
    float speed; // Speed of the monster
    float moveTimer; // Timer for movement

    Monster(int startX, int startY, float s)
      : x(startX), y(startY), speed(s), moveTimer(0) {}

    void move(float deltaTime) {
      moveTimer += deltaTime;
      if (moveTimer >= speed) {
        int direction = rand() % 4;
        int newX = x;
        int newY = y;
        if (direction == 0) newY -= 1; // Up
        if (direction == 1) newY += 1; // Down
        if (direction == 2) newX -= 1; // Left
        if (direction == 3) newX += 1; // Right

        // Check boundaries and walls
        if (newX >= 0 && newX < W && newY >= 0 && newY < H && TileMap[newY][newX] == ' ') {
          x = newX;
          y = newY;
        }
        moveTimer = 0;
      }
    }
  };

  void resetGame(int& x, int& y, float& camX, float& camY, bool& win, bool& gameOver, std::vector<Monster>&
monsters) {

    x = 4; //
    y = 4; // Reset player position
    TileMap[y][x] = 'P';//pers-player
    camX = 3;
    camY = ts * -1;
    win = false;
    gameOver = false; // Reset gameOver flag

    // Clear monsters
    monsters.clear();

    // Reset TileMap
    for (int i = 0; i < H; ++i) {
      for (int j = 0; j < W; ++j) {
        if (TileMap[i][j] == 'P' || TileMap[i][j] == 'M') {
          TileMap[i][j] = ' ';
        }
      }
    }

    TileMap[y][x] = 'P';

    // Reset monsters positions
    monsters = {
      Monster(2, 6, 0.2f),
      Monster(10, 10, 0.2f),
```

```
        Monster(15, 5, 0.2f),
        Monster(20, 15, 0.2f),
        Monster(25, 20, 0.2f),
        Monster(18, 28, 0.2f),
        Monster(31, 26, 0.2f),
        Monster(1, 4, 0.2f),
        Monster(5, 25, 0.2f)
    };
}

int main() {
    srand(static_cast<unsigned>(time(0)));
    RenderWindow window(VideoMode(9 * ts, 9 * ts), "One more?");

    Texture t;
    t.loadFromFile("C:/Users/Denis/Downloads/maze.png");//full maze pic
    Sprite plat(t);

    Texture w;
    w.loadFromFile("C:/Users/Denis/Downloads/win.png");//win screen
    Sprite winPrint(w);
    winPrint.setPosition(250, 250);

    Texture gover;
    gover.loadFromFile("C:/Users/Denis/Downloads/gameover.png");//over(monster-meeting)
    Sprite gameOverText(gover);
    gameOverText.setPosition(100, 350);

    Texture pp;
    pp.loadFromFile("C:/Users/Denis/Downloads/menuPREV.png");//pause screen(first)
    Sprite pausePREV(pp);
    pausePREV.setPosition(0, 0);

    Texture pn;
    pn.loadFromFile("C:/Users/Denis/Downloads/menuNEXT.png");////pause screen(second)
    Sprite pauseNEXT(pn);
    pauseNEXT.setPosition(0, 0);

    int x = 4, y = 4;
    int newx = 4, newy = 4;
    TileMap[y][x] = 'P';
    int nx = 4, ny = 4; // position of pers

    // Initialize monsters with reduced speeds (increased values)
    std::vector<Monster> monsters = {// x, y, speed
        Monster(2, 6, 0.2f),
        Monster(10, 10, 0.2f),
        Monster(15, 5, 0.2f),
        Monster(20, 15, 0.2f),
        Monster(25, 20, 0.2f),
        Monster(18, 28, 0.2f),
        Monster(31, 26, 0.2f),
        Monster(1, 4, 0.2f),
        Monster(5, 25, 0.2f)
    };
```

```
Clock clock;
bool continuePressed = false;
bool exitPressed = false;
while (window.isOpen()) {
   float deltaTime = clock.restart().asSeconds();

   Event event;
   while (window.pollEvent(event)) {
      if (event.type == Event::Closed)
         window.close();

      if (event.type == Event::KeyPressed) {
         if (!win && !gameOver) {//while not win and not break by monster
            if(escape == 1){
            newx = x;
            newy = y;

            if (event.key.code == Keyboard::Right) {
               newx += 1;

               if (TileMap[newy][newx] == ' ')
                  nx++;

               if (nx > 5) {
                  camX += ts;

                  nx = 5;
               }
            }
            if (event.key.code == Keyboard::Left) {
               newx -= 1;

               if (TileMap[newy][newx] == ' ')
                  nx--;

               if (nx < 4) {
                  camX -= ts;

                  nx = 4;
               }
            }
            if (event.key.code == Keyboard::Up) {
               newy -= 1;

               if (TileMap[newy][newx] == ' ')
                  ny--;

               if (ny < 3) {
                  camY -= ts;

                  ny = 3;
               }
            }
            if (event.key.code == Keyboard::Down) {
               newy += 1;
```

```
                if (TileMap[newy][newx] == ' ')
                    ny++;

                if (ny > 4) {
                    camY += ts;

                    ny = 4;
                }
            }
        }
            if (event.key.code == Keyboard::Escape) {//for pause
                escape++;

                if (escape == 3) {
                    escape = 1;
                }
            }
            if (escape == 2 && Keyboard::isKeyPressed(Keyboard::N)) {
                pnext = true;
            }

        }
    }
}

if (TileMap[newy][newx] == ' ') {
    TileMap[y][x] = ' ';
    TileMap[newy][newx] = 'P';
    x = newx;
    y = newy;
}

if (TileMap[newy][newx] == 'E') {
    TileMap[y][x] = ' ';
    TileMap[newy][newx] = 'P';

    x = newx;
    y = newy;

    win = true;
}

// Move monsters only if the game is not won and not in game over state
if (!win && !gameOver) {
    for (auto& monster : monsters) {
        int oldX = monster.x;
        int oldY = monster.y;
        if (escape == 1) {
            monster.move(deltaTime);

            // Check collision with player
            if ((((monster.x == x + 1) || (monster.x == x - 1)) && monster.y == y) || (((monster.y == y + 1) || (monster.y
== y - 1)) && monster.x == x)) {
                gameOver = true; // Set gameOver as true
            }
```

```
            else {
                TileMap[oldY][oldX] = ' '; // Clear old position
                TileMap[monster.y][monster.x] = 'M'; // New position
            }
        }
    }
}

window.clear(Color::White);
for (int i = 0; i < H; i++)
    for (int j = 0; j < W; j++) {
        if (TileMap[i][j] == 'A')
            plat.setTextureRect(IntRect(0, 0, ts, ts)); // walls
        if (TileMap[i][j] == 'P')
            plat.setTextureRect(IntRect(ts, 0, ts, ts)); // player
        if (TileMap[i][j] == 'M')
            plat.setTextureRect(IntRect(ts * 3, 0, ts, ts)); // monster
        if (TileMap[i][j] == ' ')
            plat.setTextureRect(IntRect(ts * 2, 0, ts, ts));// space
        if (TileMap[i][j] == 'E')
            plat.setTextureRect(IntRect(ts * 4, 0, ts, ts)); // exit

        //FALSE EXIT++
        if (i == 0 && j == 1)
            plat.setTextureRect(IntRect(ts * 4, 0, ts, ts));
        if (i == 0 && j == 11)
            plat.setTextureRect(IntRect(ts * 4, 0, ts, ts));
        if (i == 13 && j == 12)
            plat.setTextureRect(IntRect(ts * 4, 0, ts, ts));
        if (i == 17 && j == 12)
            plat.setTextureRect(IntRect(ts * 4, 0, ts, ts));
        if (i == 2 && j == 27)
            plat.setTextureRect(IntRect(ts * 4, 0, ts, ts));
        if (i == 3 && j == 20)
            plat.setTextureRect(IntRect(ts * 4, 0, ts, ts));
        if (i == 31 && j == 20)
            plat.setTextureRect(IntRect(ts * 4, 0, ts, ts));

        plat.setPosition(j * ts - camX, i * ts - camY);//cooperate with camera
        window.draw(plat);
    }

if (win) {
    // Win is real
    window.draw(winPrint);

    for (auto& monster : monsters) {//Update the monster's position for not stacking after win
        int oldX = monster.x;
        int oldY = monster.y;
        monster.move(deltaTime);

        TileMap[oldY][oldX] = ' '; // Clear old position
        TileMap[monster.y][monster.x] = ' '; // New position
    }
    // Key event
    if (Keyboard::isKeyPressed(Keyboard::Y)) {
```

```
            continuePressed = true;
        }
        else if (Keyboard::isKeyPressed(Keyboard::N)) {
            exitPressed = true;
        }
    }



    if (gameOver) {// game over
        window.draw(gameOverText);

        if (Keyboard::isKeyPressed(Keyboard::Y)) {
            continuePressed = true;
        }
        else if (Keyboard::isKeyPressed(Keyboard::N)) {
            exitPressed = true;
        }
    }

    if (escape == 2) {//main menu
        window.draw(pausePREV);
            if (pnext)//next menu's topic
              window.draw(pauseNEXT);

    }

    window.display();

    if (continuePressed) {//resault of pressed
        continuePressed = false;
        resetGame(x, y, camX, camY, win, gameOver, monsters);
    }
    else if (exitPressed) {
        window.close();
    }
  }
  return 0;
}
```

Роздруковані графічні результати