МИНИСТЕРСТВО науки и высшего ОБРАЗОВАНИЯ РОссИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(национальный исследовательский университет)»

Институт №3 «Системы управления, информатика и электроэнергетика»

Кафедра № 304 «Вычислительные машины, системы и сети»

Отчет по курсовому проекту по дисциплине

"Программирование"

Название

«Распознавание и воспроизведение анимации с определенной вероятностью»

Выполнил:

Студент группы М3О-219Бк-20 **Бокарев С.М.** \_\_\_\_\_\_\_\_\_\_\_

Принял:

Старший преподаватель к. 304 **Ивашенцев И.В.**\_\_\_\_\_\_\_\_\_\_\_

Москва 2021 г.

# **Оглавление**

[**Оглавление** 2](#_Toc90734250)

[Задание 3](#_Toc90734251)

[Экспертное обоснование выбранной для использования языковой среды 4](#_Toc90734252)

[Паспорт программы 7](#_Toc90734253)

[Руководство пользователя 7](#_Toc90734254)

[Руководство программиста 7](#_Toc90734255)

[Функциональная и событийная реализация 9](#_Toc90734256)

[Алгоритмизация 10](#_Toc90734257)

[Код программы 23](#_Toc90734258)

# Задание

Во входном двоичном потоке с вероятностью (85-N), где N - номер варианта, находятся две мультимедийные (со звуком) панорамы, с вероятностью (95 -N), находится одна мультимедийная панорама.

Каждая панорама представляет из себя законченный, по смыслу, мультик, сформированный на базе библиотеки<graphics.h>

Мелодия создается из набора нот библиотеки <dos.h>, причем количество звуков должно быть (50-N).

Необходимо найти вероятность не нахождения панорам в потоке, обнаружить, распознать и воспроизвести хотя бы одну мультимедийную панораму на экране.

В разделе Экспертное обоснование выбранной для использования языковой среды необходимо проанализировать и обосновать Ваш выбор среды.

В разделе Функциональная и событийная реализация необходимо перечислить весь используемый Вами функционал, а также вероятность его использования в Вашем алгоритме.

В разделе Алгоритмизация необходимо привести развернутый алгоритм программы.

В разделе Код программы привести код программы с необходимыми комментариями.

В разделе скрин результата привести скрин мультика.

В разделе Паспорт программы указать языковую среду, операционную среду и аппаратуру реализации.

# Экспертное обоснование выбранной для использования языковой среды

Техническое задание требует от разработчика некоторых условий, которые сужают круг используемых языков и решений.

1. Программа должна выполняться быстро.
2. Программа позволит решить задачи смены звука и изображения.
3. Показ и работа должны быть осуществлены без лишних денежных затрат.

Для данной работы рассматривались следующие критерии, которые важны при выборе языка программирования, используемого для отрисовки 2D графики и игры с управлением клавишами на клавиатуре.

* + Графика 2D – простота использования графики и скорости ее отрисовки.
  + Скорость выполнения – время, затрачиваемое на выполнение программы – показывает производительность языка.
  + Кроссплатформенность - как легко будет запускать программу на разных платформах (Windows, Mac, Lunix).
  + Размер исходного кода – количество символов в программе без учета разделителей – показывает громоздкость полученных программ.
  + Популярность среди программистов и работодателей – распространенность языка – показывает перспективность языка.

Рассмотрим несколько различных решений:

* 1. Язык С, библиотека graphics.h

Язык Си позволяет довольно быстро развернуть исполнение программы на различных устройствах. Язык существует довольно давно, поэтому большинство вычислительных машин способны запускать приложения, написанные на нем. Средства разработки предоставляют широкий спектр выбора высококлассных библиотек и функций для работы с графикой, звуком, математическим аппаратом, памятью и вычислительной системой. Данный язык предоставляет также большой набор низкоуровневых библиотек и функций, позволяющих реализовывать близкие к аппаратной части ситуации. Cи распространяется с эмулятором операционной системы, необходимой для корректной работы, поэтому запуск кода возможен не только на компьютере разработчика, но и на устройствах других пользователей.

Для написания программы на Си целесообразно использовать среду разработки Visual Studio Code потому, что это современная, интеллектуальная и быстрая по времени компиляции среда, подходящая для обучения студентов профессиональным методам решения задач. Так же данная IDE максимально проста для понимания и удобна в использовании.

Данный метод был выбран, как основной.

* 1. Python.

Python является одним из высокоуровневых языков программирования, что прекрасно подходит для выполнения задания, но данный язык является интерпретируемым, что затрудняет быстрое выполнение кода. Данный язык работает с собственным интерпретатором, который так же необходимо загрузить. К счастью, он довольно часто попадает в дистрибутивы linux, как основное приложение для первичной загрузки.

Python является языком с огромным набором высокоуровневых операций, но со слабым набором низкоуровневых обработчиков. Задачи чтения и записи в бинарный файл реализованы поверхностно, что мешает решать задачи, описанные в курсовом проекте.

* 1. Java.

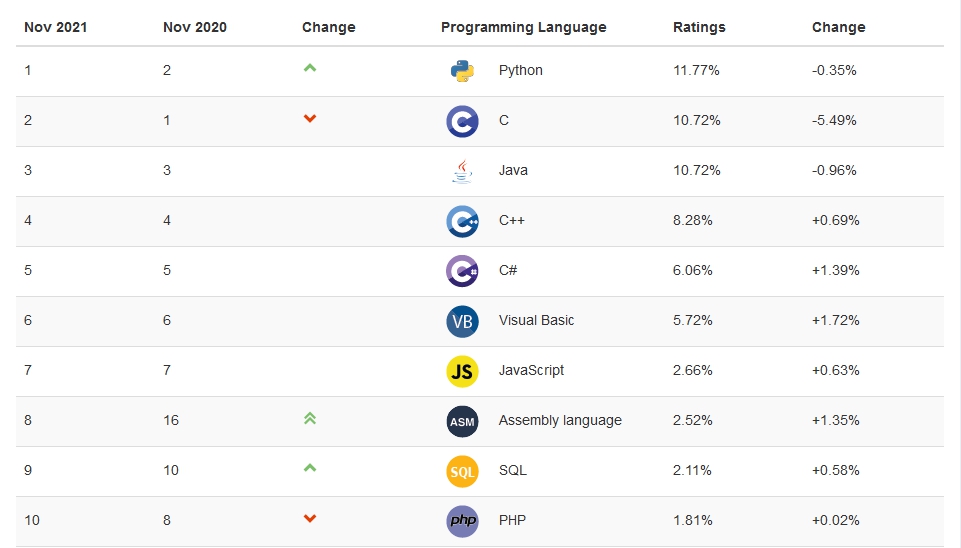
Ситуация с Java аналогична ситуации с Python, этот язык программирования прекрасно подошел бы для выполнения задачи. Но данный язык довольно долго запускается в своей личной виртуальной машине, что мешает быстродействию. Также Java не является моим основным языком программирования. По этой причине мои решения поставленных проблем скорее всего оказались бы сложными и ресурсозатратными.

* 1. PHP.

Являлся одним из кандидатов для выполнения задания, но самостоятельно, без помощи сторонних средств по типу JavaSсript PHP запустить невозможно. В связи с чем было решено отказаться от его использования.

Так же нам следует учитывать популярность данных языков.

На ноябрь 2021 года популярность выбранных языков программирования, изменение популярности по сравнению с прошлым годом, рейтинг и изменение в процентах по сравнению с прошлым годом приводятся в следующей таблице.



Источник: индекс TIOBE.

Вывод

Для выполнения курсового проекта был выбрал язык Си по той причине, что он является оптимальным по скорости работы, по актуальности и востребованности, а также по простоте работы с графикой и звуком.

В качестве среды написания кода была выбрана Visual Studio Code потому, что это бесплатный, популярный, широко распространённый редактор кода, с интеллектуальным анализом кода, позволяющий ускорить процесс написания программы на языке Си.

# Паспорт программы

Программа была написана в среде Visual Studio Code 2021 c использованием библиотеки graphics.h на 64 битном компьютере с платформой Intel.

Processor AMD Ryzen 7 4800H with Radeon Graphics 2.90 GHz

Installed RAM 16,0 ГБ (usable: 15,4 ГБ)

System type 64-bit operating system, x64-based processor

Pen and touch Not available

Edition Windows 10

Version 21H2

Installed on 01/01/2020

OS build 19044.1415

Experience Windows Feature Experience Pack 120.2212.3920.0

Результат работы программы можно посмотреть по ссылке до 28.12.2028:

<https://clck.ru/Zvaey>

# Руководство пользователя

1. Скачать файлы программы: исходный код, командный файл DOS <https://clck.ru/ZvckK> можно до 28.12.2028.
2. Двойным щелчком нажать на командный файл DOS с расширением .bat
3. Открыть двойным щелчком мыши файл программы с мультипликацией DemoGraphics.exe.
4. Появиться окно с мультипликацией.

# Руководство программиста

1. Скачать Visual Studio Code <https://code.visualstudio.com/>
2. Скачать MinGW c++ − набор инструментов разработки программного обеспечения для создания приложений <https://www.mingw-w64.org/downloads/>
3. Загрузить файлы заголовков библиотек graphics.h, dos.h, winbgim.h, libbgi.a по адресу <https://clck.ru/ZvgVJ>
4. Скопируйте файлы "graphics.h" и "winbgim.h" и "dos.h" в папку "include" среды MinGw (C:\MinGW\include)
5. Скопируйте файл "libbgi.a" в папку "lib" среды MinGw (C:\MinGW\lib)
6. Скопируйте файлы "graphics.h", "winbgim.h", "libbgi.a" в рабочую папку, где пишется код программы.
7. Откройте среду разработки Visual Studio Code. Назовите файл с программой, например так: Demo05.c
8. Нажмите вкладку «Терминал» и вставьте команду сборки “g++ Demo05.c -o DemoGraphics -lbgi -lgdi32 -lcomdlg32 -luuid -loleaut32 -lole32 -lWinmm”

.\DemoGraphics

1. Появиться окно с мультипликацией.

# Функциональная и событийная реализация

В работе используется:

Из основной функции main() происходит вызов подпрограмм, воспроизведение музыки, изменение координат и времени, а также подсчет вероятности.

**int get\_rand\_range\_int(const int min, const int max)** – получение рандомного выходного значения при заданных значениях.

**void Man(int x, int corner, int GroundY, bool proxod)** – основная функция, позволяющая отрисовать на экране человека, держащего в руках транспортную карту.

**void Fence()** – функция отрисовки ограждения платформы от непрошедших через турникет с распашными створками.

**void TurnStile(bool proxod)** – отрисовка турникета с распашными створками.

**void Train(int x, int ldisp, int GroundY)**  – отрисовка поезда.

**void RailWay()** – позволяет нарисовать на экране железнодорожные рельсы.

**void Platform()** – вывод на экран прямоугольной платформы.

**void playSoundFirstPart()** – проигрыш первой части музыки.

**void playSoundSecondPart()** – проигрыш второй части музыки.

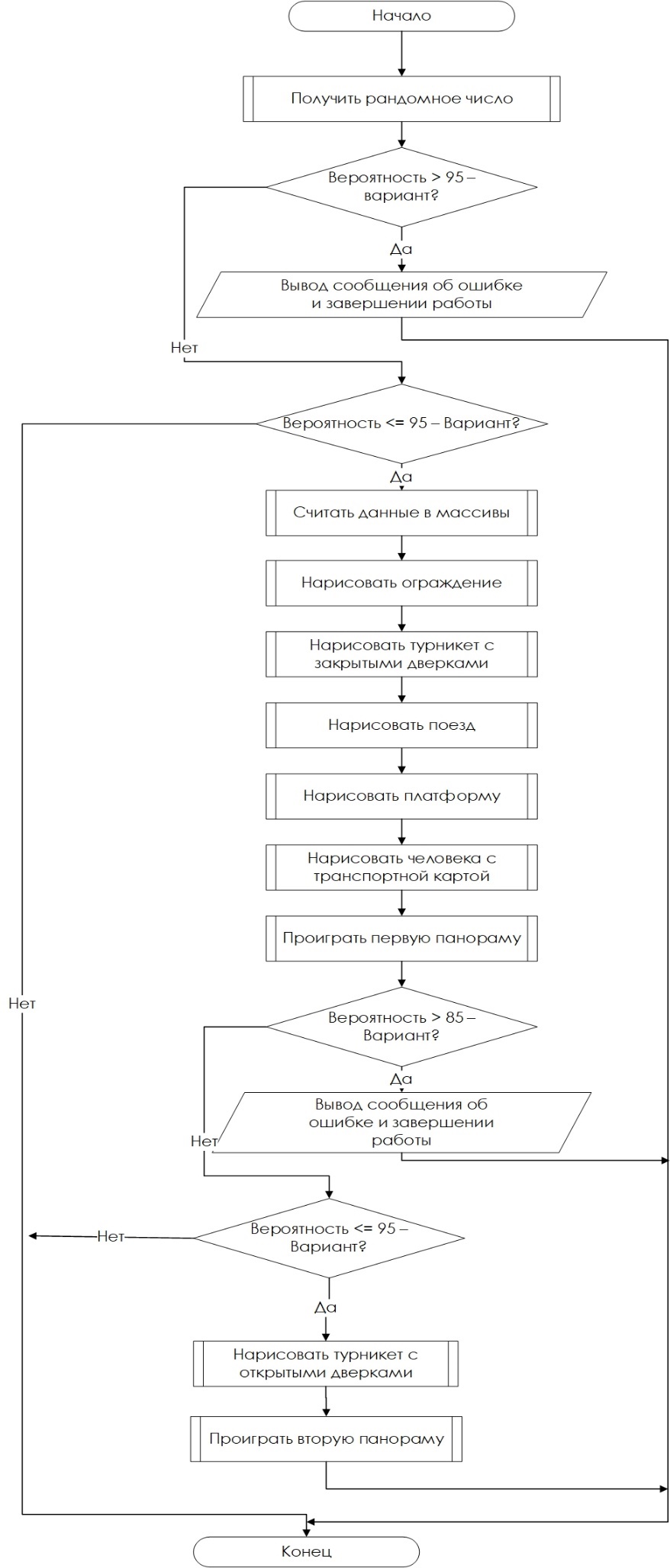
**int main()** – подготовка к показу и воспроизведение мультика.

# Алгоритмизация

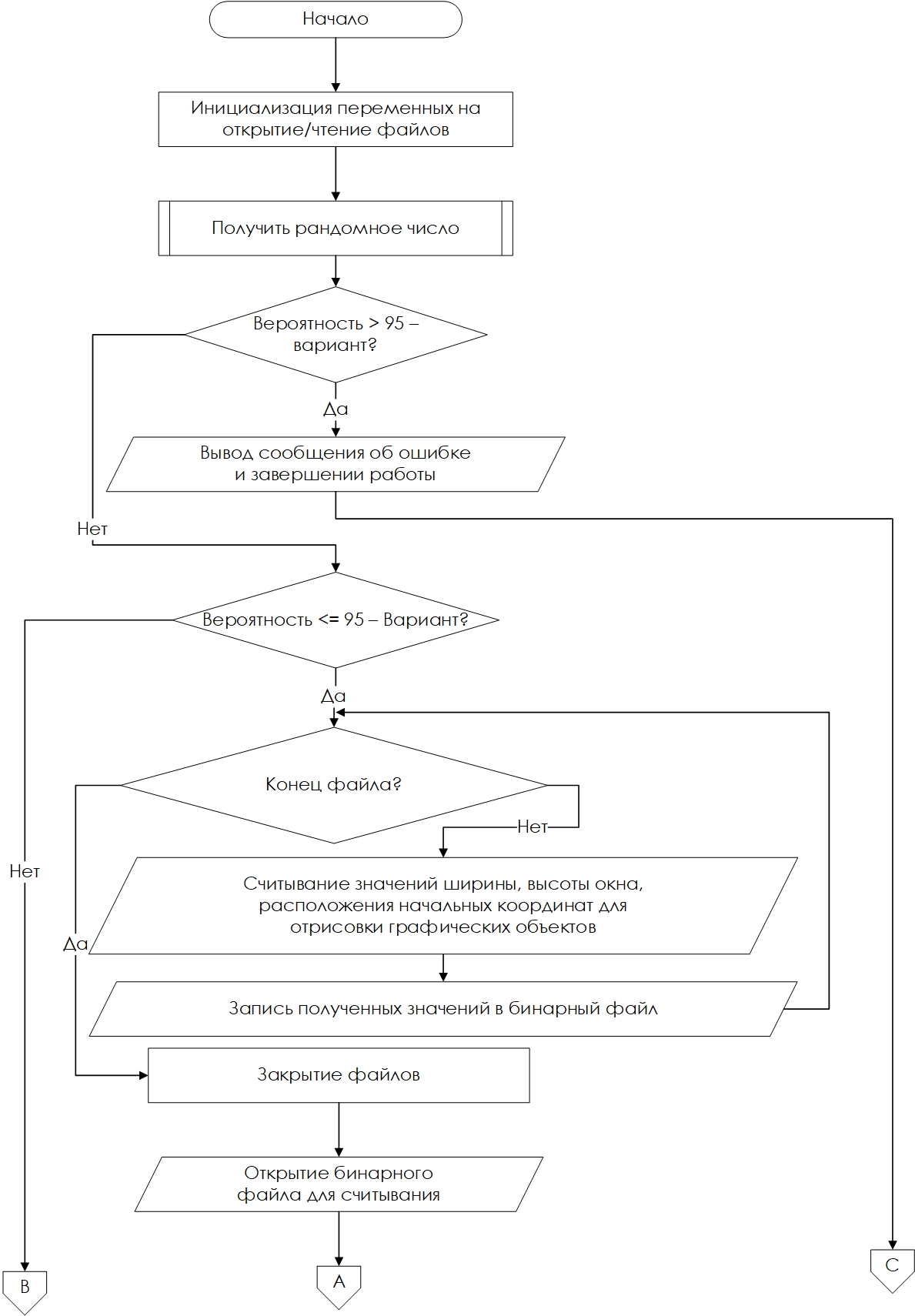
int main()

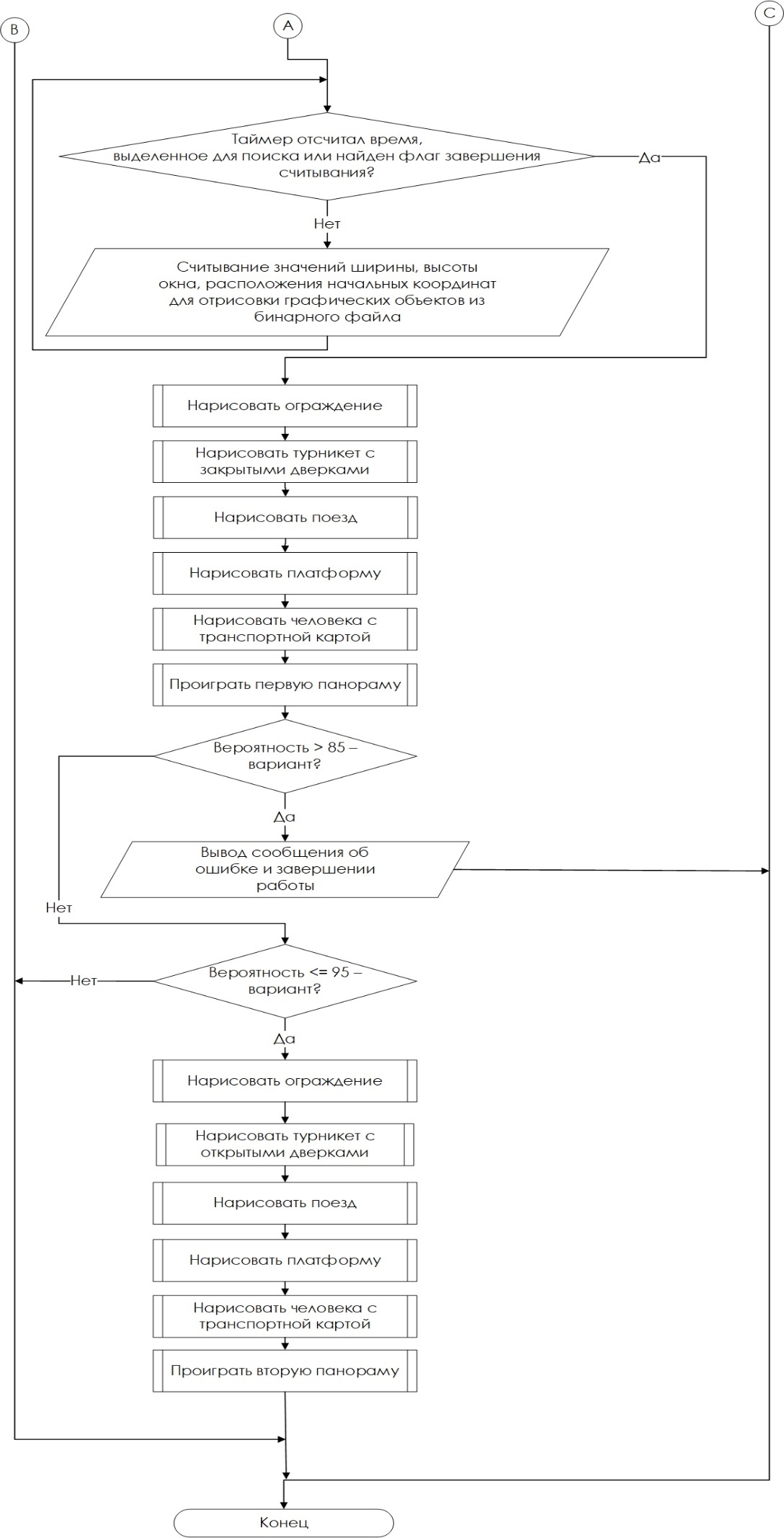
Алгоритм

1. Инициализация переменных.
2. Получение значения вероятности.
3. Запуск первой панорамы, если вероятность 95%.
4. Запуск второй панорамы, если вероятность 85%.
5. Завершение работы, если вероятность 5%.



Структурная схема алгоритма main.



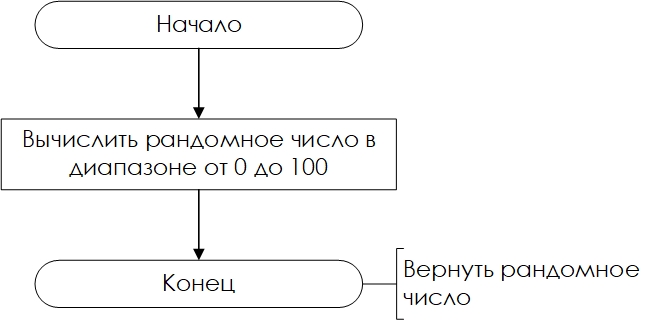


Функциональная схема main.

int get\_rand\_range\_int()

Алгоритм

1. Получить границы измерений.
2. Получить значение рандома.
3. Перемножить границы и рандом.
4. Получить искомое число.

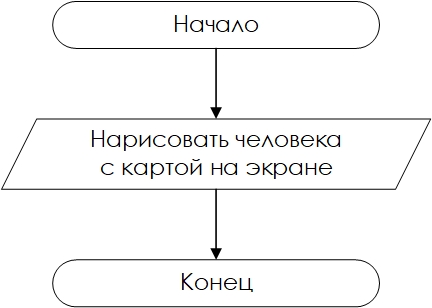


Функциональная схема алгоритма get\_rand\_range\_int.

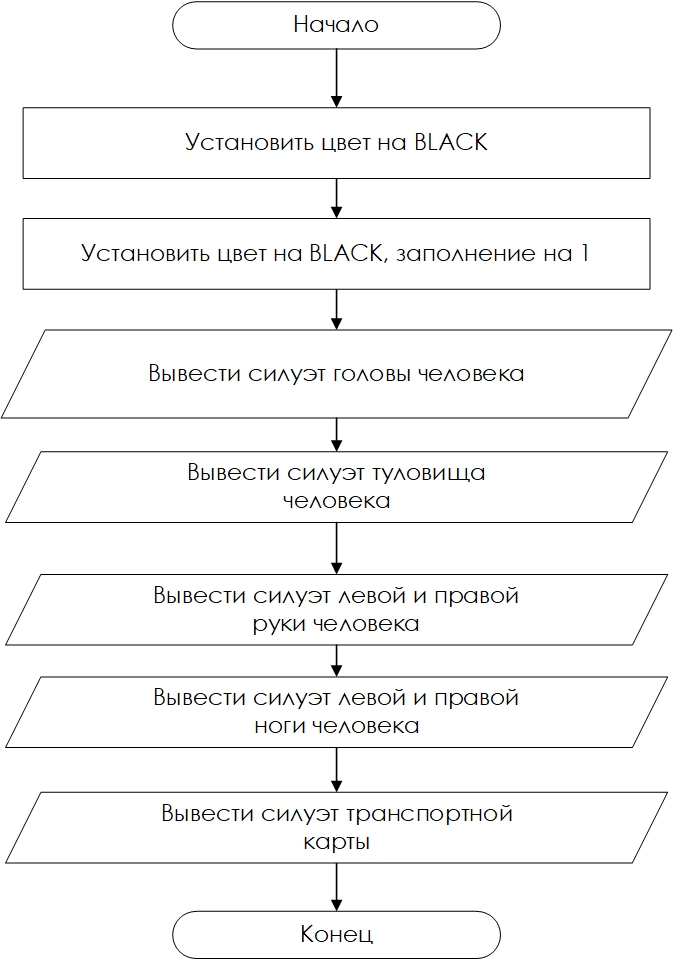
void Man(int x, int corner, int GroundY, bool proxod)

Алгоритм

1. Поменять цвет и заливку.
2. Вывести голову человека.
3. Вывести туловище.
4. Вывести левую и правую руку.
5. Вывести левую и правую ногу.
6. Вывести транспортную карту в правой руке.



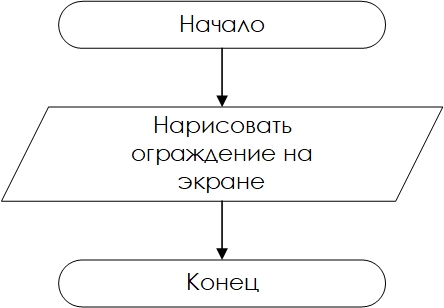
Структурная схема алгоритма Man.



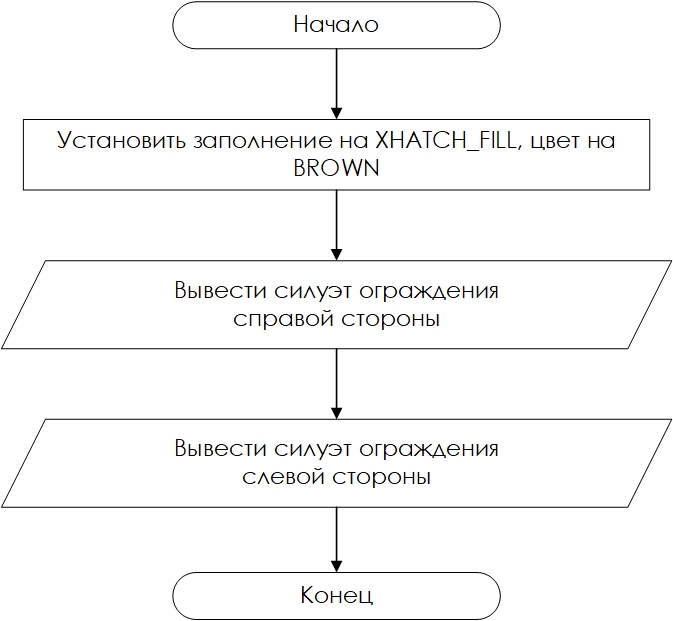
Функциональная схема алгоритма Man.

void Fence()

Алгоритм

1. Поменять цвет и заливку.
2. Нарисовать левую границу ограждения.
3. Нарисовать правую границу ограждения.

Структурная схема алгоритма Fence.

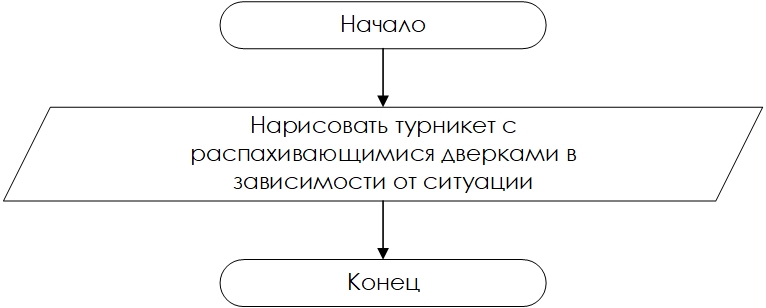


Функциональная схема алгоритма Fence.

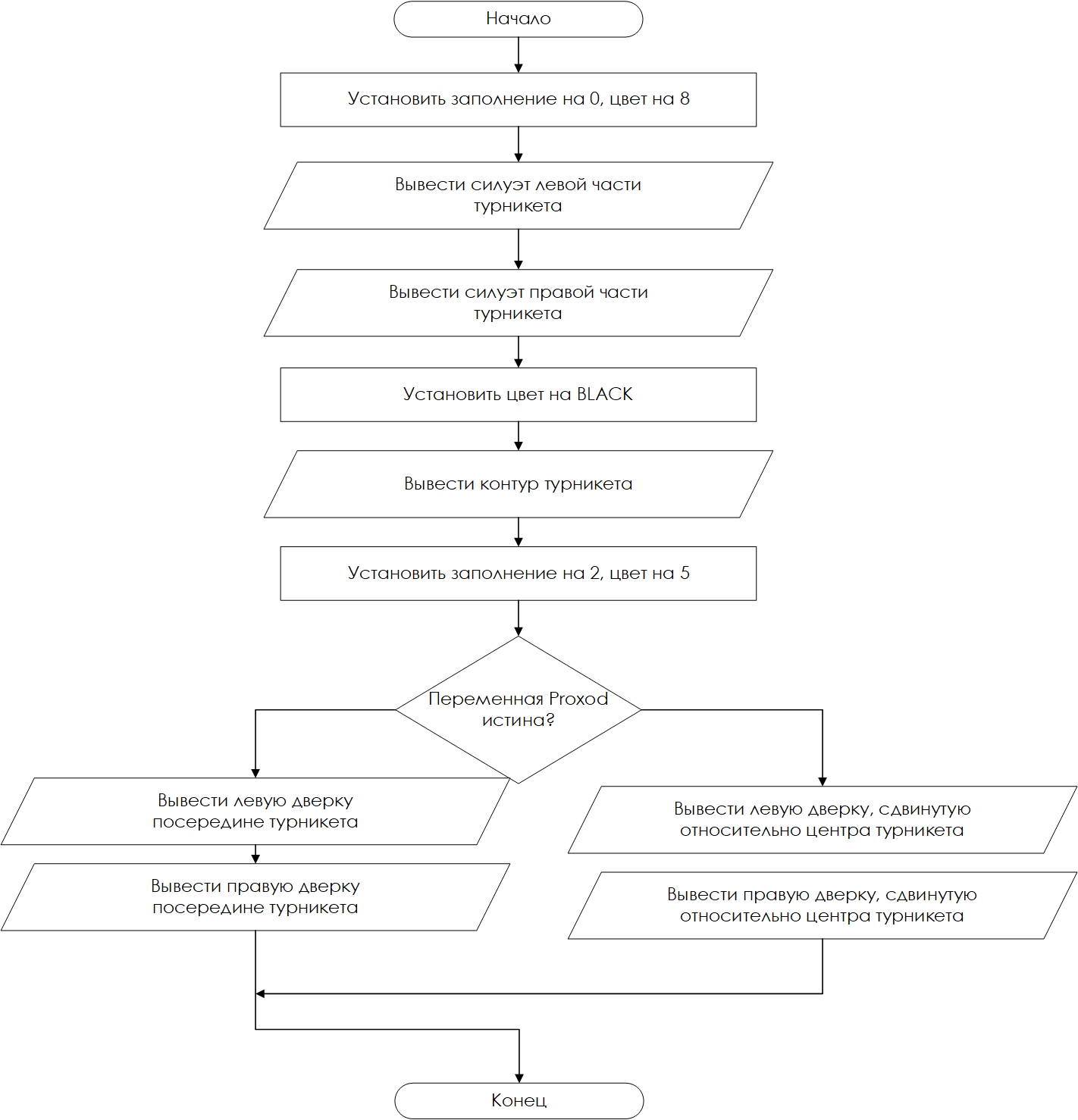
void TurnStile(bool proxod)

Алгоритм

1. Поменять цвет и заливку.
2. Отрисовать левый блок турникета.
3. Отрисовать правый блок турникета.
4. Поменять заливку и цвет.
5. Отрисовать левую дверку турникета.
6. Отрисовать правую дверку турникета.
7. Поменять заливку и цвет.
8. Нарисовать круг, обозначающий считыватель данных с транспортной карты.



Структурная схема алгоритма TurnStile.

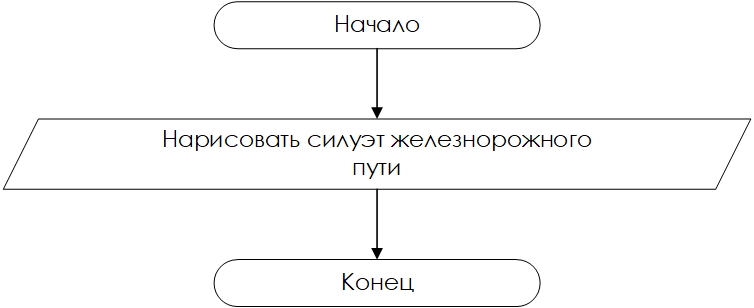


Функциональная схема алгоритма TurnStile.

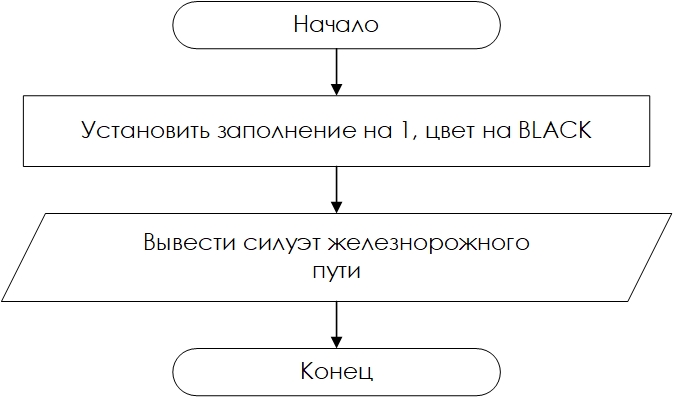
void RailWay()

Алгоритм

1. Поменять цвет и заливку.
2. Отобразить силуэт железнодорожного пути.



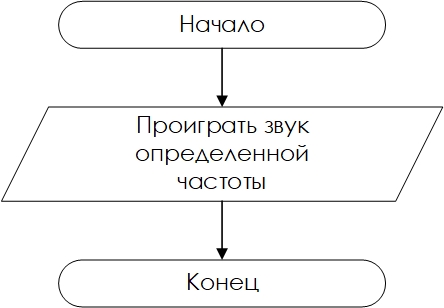
Структурная схема алгоритма.



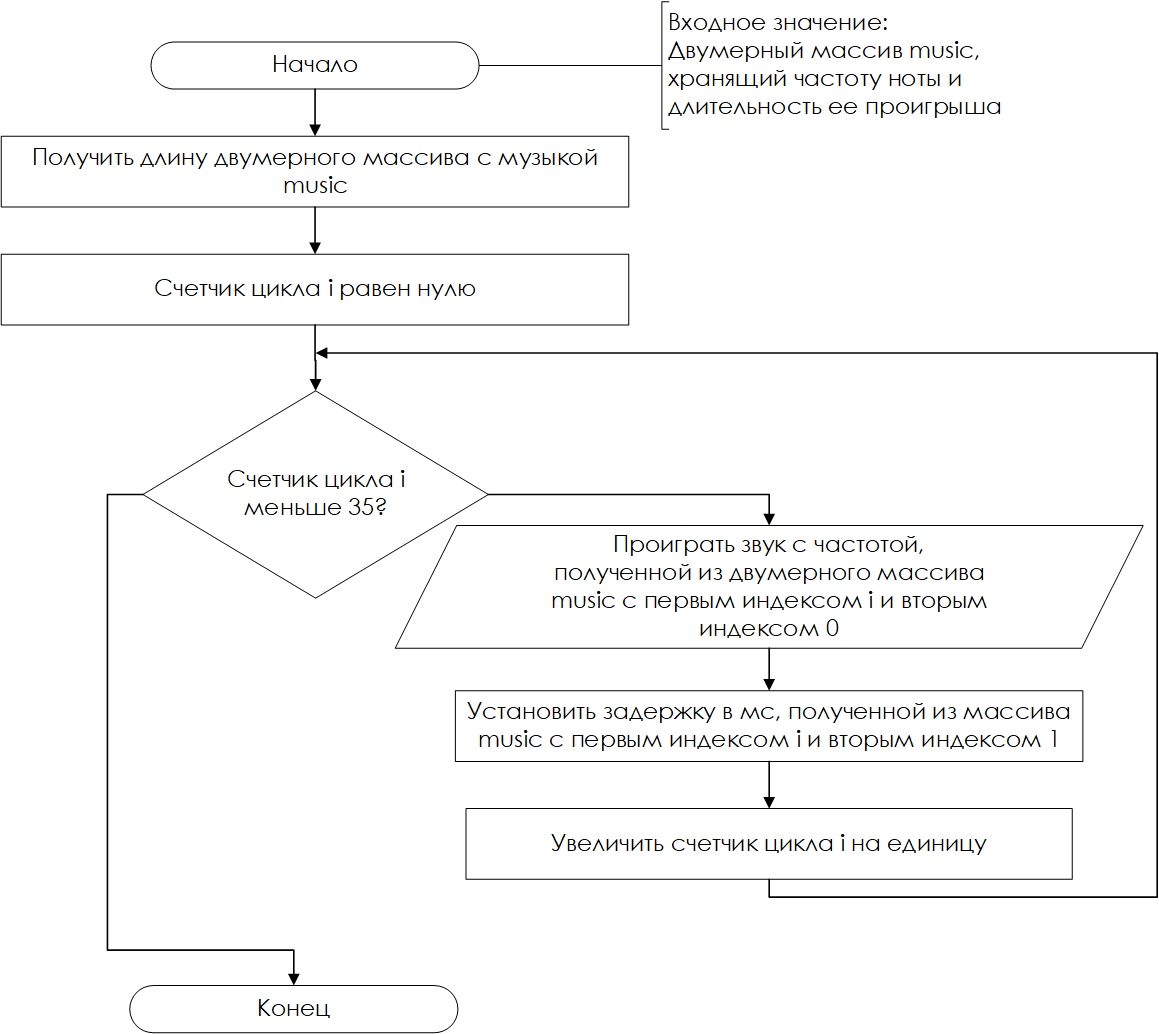
Функциональная схема алгоритма RailWay.

void playSoundFirstPart()

1. Получить базовую частоту.
2. Получить из двумерного массива частоту звука и задержку, с которой необходимо проиграть звук.
3. В цикле от 0 до 35 проиграть звук и сделать задержку.



Структурная схема алгоритма playSoundFirstPart.



Функциональная схема алгоритма playSoundFirstPart.

# Код программы

#include "graphics.h" // graphics library

#include <dos.h> // library of delays and work with sound

#include <stdio.h> // input and output library

#include <time.h> // library for more accurate randomness

#include <conio.h> // library for console input

#include <fstream> // read and write tto files

short int get\_rand\_range\_int(const short int min, const short int max) // returns a random value from min to max

{

    return rand() % (max - min + 1) + min;

}

void Man(short int xy\_start\_man[], bool proxod) // отрисовка человека с транспортной картой

{

    setcolor(BLACK);

    if(proxod){ // проход через турникет выполнен

        short int x = xy\_start\_man[3];

        short int corner = xy\_start\_man[4];

        short int GroundY = xy\_start\_man[5];

        line(x, GroundY - 70, x + 20, GroundY - 40); // рука правая опущена

        line(x + 13, GroundY - 40, x + 30, GroundY - 50); // карта 1 сторона

        line(x + 13, GroundY - 40, x + 20, GroundY - 35); // карта 2 сторона

        line(x + 20, GroundY - 35, x + 38, GroundY - 45); // карта 3 сторона

        line(x + 38, GroundY - 45, x + 30, GroundY - 50); // карта 4 сторона

        setfillstyle(1, BLACK);

        circle(x, GroundY - 90, 10); // Голова

        line(x, GroundY - 80, x, GroundY - 30); // спина

        line(x, GroundY - 70, x - 20, GroundY - 40); // рука левая

        line(x, GroundY - 30, x + corner - 30, GroundY); // нога левая

        line(x, GroundY - 30, x - corner + 30, GroundY); // нога правая

    }else{ // проход через турникет не выполнен

        short int x = xy\_start\_man[0];

        short int corner = xy\_start\_man[1];

        short int GroundY = xy\_start\_man[2];

        line(x, GroundY - 70, x + 20, GroundY - 90 ); // рука правая поднята

        line(x + 13, GroundY - 90, x + 30, GroundY - 100); // карта 1 сторона

        line(x + 13, GroundY - 90, x + 20, GroundY - 85); // карта 2 сторона

        line(x + 20, GroundY - 85, x + 38, GroundY - 95); // карта 3 сторона

        line(x + 38, GroundY - 95, x + 30, GroundY - 100); // карта 4 сторона

        setfillstyle(1, BLACK);

        circle(x, GroundY - 90, 10); // Голова

        line(x, GroundY - 80, x, GroundY - 30); // спина

        line(x, GroundY - 70, x - 20, GroundY - 40); // рука левая

        line(x, GroundY - 30, x + corner - 30, GroundY); // нога левая

        line(x, GroundY - 30, x - corner + 30, GroundY); // нога правая

    }

}

void Fence(short int xy\_start\_fence[]){ // функция отрисовки ограждения

    setfillstyle(XHATCH\_FILL, BROWN); //Установка стиля заливки

    bar(xy\_start\_fence[0], xy\_start\_fence[1], xy\_start\_fence[2], xy\_start\_fence[3]); // левая стена ограды

    bar(xy\_start\_fence[4], xy\_start\_fence[5], xy\_start\_fence[6], xy\_start\_fence[7]); // правая стена ограды

}

void TurnStile(bool proxod){ // функция отрисовки турникета с открывающимися дверками

    setfillstyle(0, 8); //Установка стиля заливки

    bar(500, 350, 550, 450); // левый блок

    bar(650, 350, 700, 450); // правый блок

    setcolor(BLACK);

    rectangle(500, 350, 550, 450); // контур

    rectangle(650, 350, 700, 450); // контур

    setfillstyle(2, 5); //Установка стиля заливки

    if(proxod){ // открыть дверки

        bar(525, 375, 555, 435); // левый блок

        bar(630, 375, 670, 435); // правый блок

        setcolor(BLACK);

        rectangle(525, 375, 555, 435); // контур

        rectangle(630, 375, 670, 435); // контур

    }

    else{ // закрыть дверки

        bar(550, 375, 590, 435); // левый блок

        bar(600, 375, 650, 435); // правый блок

        setcolor(BLACK);

        rectangle(550, 375, 590, 435); // контур

        rectangle(600, 375, 650, 435); // контур

        outtextxy(500, 330, "Read...");

    }

    setcolor(GREEN);

    circle(525, 365, 10);

    setfillstyle(CLOSE\_DOT\_FILL, RED); // Установка стиля заливки

    fillellipse(525, 365, 8, 8); // круг кардридера

}

void Train(short int xy\_start\_train[]) // функция отрисовки поезда

{

    short int x = xy\_start\_train[0]; // х координата старта отрисовки

    short int GroundY = xy\_start\_train[1]; // у координата старта отрисовки

    setcolor(WHITE);

    line(x - 35, GroundY - 180, x + 45, GroundY - 180);

    line(x + 45, GroundY - 180, x + 45, GroundY - 100);

    line(x + 45, GroundY - 100, x + 200, GroundY - 100);

    line(x + 200, GroundY - 100, x + 250, GroundY - 30);

    line(x - 35, GroundY - 30, x + 250, GroundY - 30);

    line(x - 35, GroundY - 30, x -35, GroundY - 180);

    line(x + 150, GroundY - 100, x + 150, GroundY - 130);

    line(x + 120, GroundY - 100, x + 120, GroundY - 130);

    line(x + 150, GroundY - 130, x + 170, GroundY - 160);

    line(x + 120, GroundY - 130, x + 100, GroundY - 160);

    line(x + 100, GroundY - 160, x + 170, GroundY - 160);

    setfillstyle(1, RED);

    fillellipse(x, GroundY - 30, 30, 30);

    fillellipse(x + 90, GroundY - 30, 30, 30);

    fillellipse(x + 180, GroundY - 30, 30, 30);

    line(x - 30  , GroundY - 35  , x + 190  , GroundY - 35  );

    line(x + 120, GroundY - 160, x + 145  , GroundY - 195);

    line(x + 160, GroundY - 160, x + 150  , GroundY - 200);

    line(x + 150  , GroundY - 200, x + 100  , GroundY - 240);

    line(x + 145  , GroundY - 195, x + 100  , GroundY - 210);

    line(x + 100  , GroundY - 240, x + 50  , GroundY - 240);

    line(x + 100  , GroundY - 210, x + 50  , GroundY - 210);

    rectangle(95, 80, 120, 110);

}

void RailWay(short int xy\_start\_railway[]){ // отрисовка желехнодорожного пути

    setfillstyle(1, BLACK); // установка стиля и цвета заливки

    bar(xy\_start\_railway[0], xy\_start\_railway[1], xy\_start\_railway[2], xy\_start\_railway[3]); // отрисовка жд рельсы

}

void Platform(short int xy\_platform[]){ // отрисовка жд платформы

    setcolor(BLACK);

    line(720, 50, 970, 50); // верхняя часть

    line(670, 200, 920, 200); // нижняя часть

    line(670, 240, 920, 240); // нижняя часть платформы 3д

    line(720, 50, 670, 200); // левая сторона

    line(970, 50, 920, 200);// праввя сторона

    line(970, 90, 920, 240); // правая 3д

    line(670, 200, 670, 240); // нижняя 3д

    line(920, 200, 920, 240); // левая 3д

    line(970, 50, 970, 90); // боковая 3д

}

void playSoundFirstPart(short int music\_prop[]){ // музыка первая часть

    short int tempo = 310;

    short int step = ((60 \* 1000) / tempo)  / 4;

    // музыка

    short int music[music\_prop[0]][2] = {// массив с нотами и длительностью проигрывания

        {185, step \* 1},{207, step \* 1},{233, step \* 1},{277,step \* 1},

        {220,step \* 1},{311,step \* 5},{0,step \* 4},{277,step \* 2},

        {329, step \* 1},{233, step \* 1},{277, step \* 1},{220,step \* 1},

        {185,step \* 6},{329,step \* 1},{440,step \* 2},{440,step \* 2},

        {554, step \* 2},{440, step \* 3},{329, step \* 1},{329,step \* 1},

        {329,step \* 2},{311,step \* 5},{0,step \* 4},{277,step \* 2},

        {329, step \* 1},{493, step \* 1},{440, step \* 1},{415,step \* 1},

        {369,step \* 6},{440,step \* 2},{440,step \* 2},{311,step \* 1},

        {311, step \* 3},{311, step \* 2},{415, step \* 2}

    };

    for(short int i = 0; i < 35; i++){// последовательное проигрывание музыки

        sound(music[i][0], music[i][1]);

    }

}

void playSoundSecondPart(short int music\_prop[]){ // музыка вторая часть

    short int tempo = 110;

    short int step = ((60 \* 1000) / tempo)  / 4;

    short int music[music\_prop[1]][2] = { // массив с нотами и длительностью проигрывания

        {349, step \* 4},{440, step \* 4},{392, step \* 2},{261, step \* 4},

        {349, step \* 4},{392, step \* 2},{440, step \* 2},{349, step \* 4},

        {349, step \* 4},{392, step \* 2},{261, step \* 4},{349, step \* 4},

        {392, step \* 2},{440, step \* 2},{349, step \* 4}

    };

    for(short int i = 0; i < 15; i++){ // последовательное проигрывание музыки

        sound(music[i][0], music[i][1]);

    }

}

void writeDataInFile(short int lenght, short int array[], FILE\* outputbin){

    for(short int i = 0; i < lenght; i++)

        fwrite(&array[i], sizeof(short int), 1, outputbin);

}

int main() // главная функция

{

    const short int N = 2; // variant

    FILE \*outputbin = NULL;

    outputbin = fopen("output.bin", "wb");// opening a file for binary writing

    if (outputbin == NULL) {

        printf("Error opening file");

        getch();

        exit(0);

    }

    short int wight = 1000, height = 600, randProb = 0; // decision variables

    short int aflag = 999; // flag to start read/write data

    srand(time(NULL)); // set a seed for pseudo random generation

    randProb = get\_rand\_range\_int(1, 100); // get a random value for the probability

    fwrite(&randProb, sizeof(short int), 1, outputbin); // write probability in file

    fwrite(&aflag, sizeof(short int), 1, outputbin); // flag start read/write data

    fwrite(&wight, sizeof(short int), 1, outputbin); // write wight in file

    fwrite(&height, sizeof(short int), 1, outputbin); // write height in file

    // инициализация массива координат

    short int xy\_start\_train[2] = {100, 250};

    short int xy\_start\_man[6] = {510, 50, 480, 823, 50, 190};

    short int xy\_start\_railway[4] = {0, 250, 1000, 270};

    short int xy\_start\_fence[8] = {0, 450, 500, 300, 700, 450, 1000, 300};

    short int xy\_platform[36] = {720, 50, 970, 50, 670, 200, 920, 200, 670, 240, 920, 240, 720, 50, 670,

    200, 970, 50, 920, 200, 970, 90, 920, 240, 670, 200, 670, 240, 920, 200,

    920, 240, 970, 50, 970, 90};

    // инициализация размера массива нот

    short int music\_prop[2] = {35, 15};

    // запись массива координат

    writeDataInFile(2, xy\_start\_train, outputbin);

    writeDataInFile(2, music\_prop, outputbin);

    writeDataInFile(4, xy\_start\_railway, outputbin);

    writeDataInFile(6, xy\_start\_man, outputbin);

    writeDataInFile(8, xy\_start\_fence, outputbin);

    writeDataInFile(36, xy\_platform, outputbin);

    fclose(outputbin); // close file for writing

    FILE \*inputbin = NULL; // input text file

    inputbin = fopen("output.bin", "rb"); // input binary file

    if (inputbin == NULL) { // open file for reading

        printf("Error opening file");

        getch();

        exit(0);

    }

    short int valueOfProbability; // значение вероятности

    fread(&valueOfProbability, sizeof(short int), 1, inputbin);

    initwindow(wight, height); // инициализация окна с мультимедиа панорамой

    if (valueOfProbability > (95 - N)) // less than 5 percent chance

    { // none of the panoramas are played

        printf("Your cartoon in another program");

    }

    if (valueOfProbability <= (95 - N)) // probability up to 95 percent

    { // it is possible to play one or two panoramas

        while (true) { // find the start of data by value flag

            fread(&aflag, sizeof(short int), 1, inputbin);

            if(aflag == 999) break; // найден флаг начала данных мультимедиа панорамы

        }

        fread(&wight, sizeof(short int), 1, inputbin);

        fread(&height, sizeof(short int), 1, inputbin);

        printf("Read from binary file probability = %d\n", valueOfProbability);

        printf("Read from binary file wight = %d\n", wight);

        printf("Read from binary file height = %d\n", height);

        // чтение данных из двоичного потока

        for(short int i = 0; i < 2; i++) fread(&xy\_start\_train[i], sizeof(short int), 1, inputbin);

        for(short int i = 0; i < 2; i++) fread(&music\_prop[i], sizeof(short int), 1, inputbin);

        for(short int i = 0; i < 4; i++) fread(&xy\_start\_railway[i], sizeof(short int), 1, inputbin);

        for(short int i = 0; i < 6; i++) fread(&xy\_start\_man[i], sizeof(short int), 1, inputbin);

        for(short int i = 0; i < 8; i++) fread(&xy\_start\_fence[i], sizeof(short int), 1, inputbin);

        for(short int i = 0; i < 36; i++) fread(&xy\_platform[i], sizeof(short int), 1, inputbin);

        setfillstyle(SOLID\_FILL, DARKGRAY);

        bar(0, 0, wight, height); // background

        Train(xy\_start\_train);

        RailWay(xy\_start\_railway);

        setbkcolor(LIGHTGRAY);

        Fence(xy\_start\_fence);

        Platform(xy\_platform);

        TurnStile(false);

        Man(xy\_start\_man, false);

        playSoundFirstPart(music\_prop);

    }

    if (valueOfProbability <= (85 - N)) //less than 85 percent chance

    { // play both panoramas

        cleardevice();

        setfillstyle(SOLID\_FILL, DARKGRAY);

        bar(0,0,1000,600); // background

        Fence(xy\_start\_fence);

        TurnStile(true);

        Train(xy\_start\_train);

        RailWay(xy\_start\_railway);

        Platform(xy\_platform);

        Man(xy\_start\_man, true);

        playSoundSecondPart(music\_prop);

        cleardevice();

        setfillstyle(SOLID\_FILL, DARKGRAY);

        bar(0, 0 ,wight, height); // background

        Train(xy\_start\_train);

        RailWay(xy\_start\_railway);

        setbkcolor(LIGHTGRAY);

        Fence(xy\_start\_fence);

        Platform(xy\_platform);

        TurnStile(false);

        Man(xy\_start\_man, true);

    }

    getch();

}

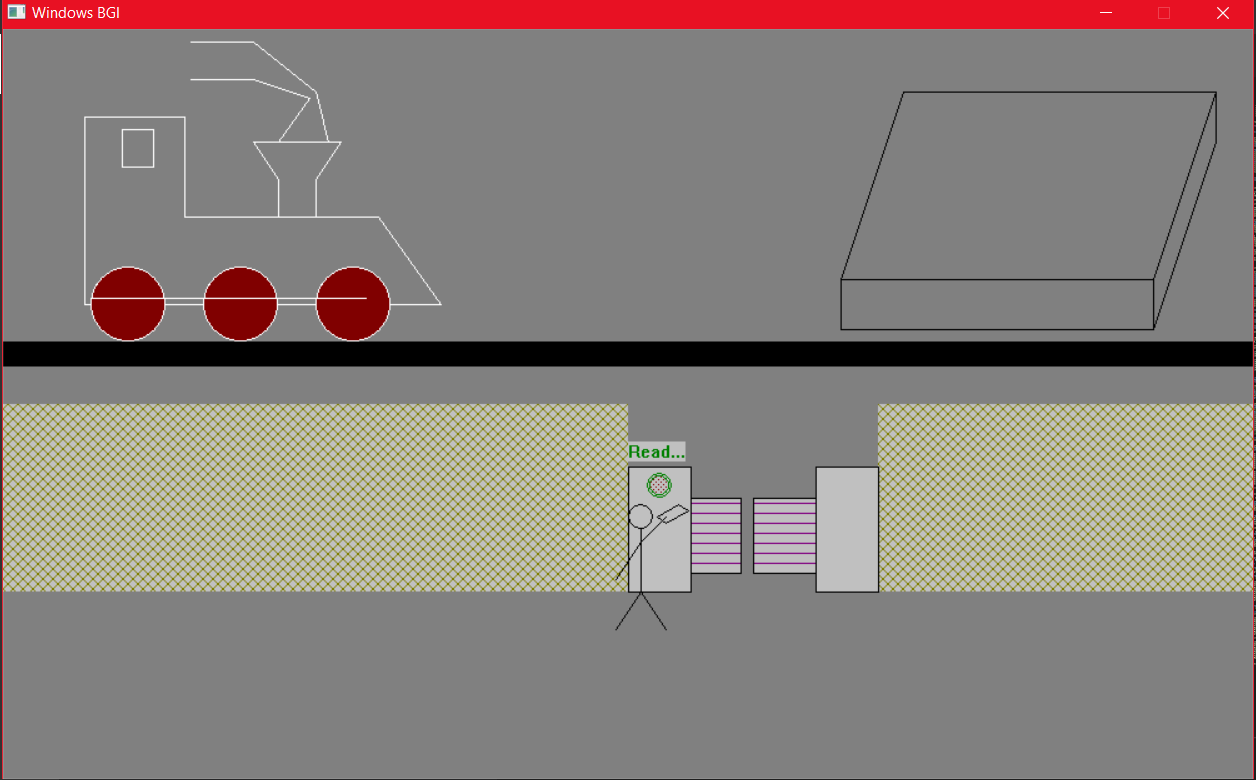
# Словесное описание изображения

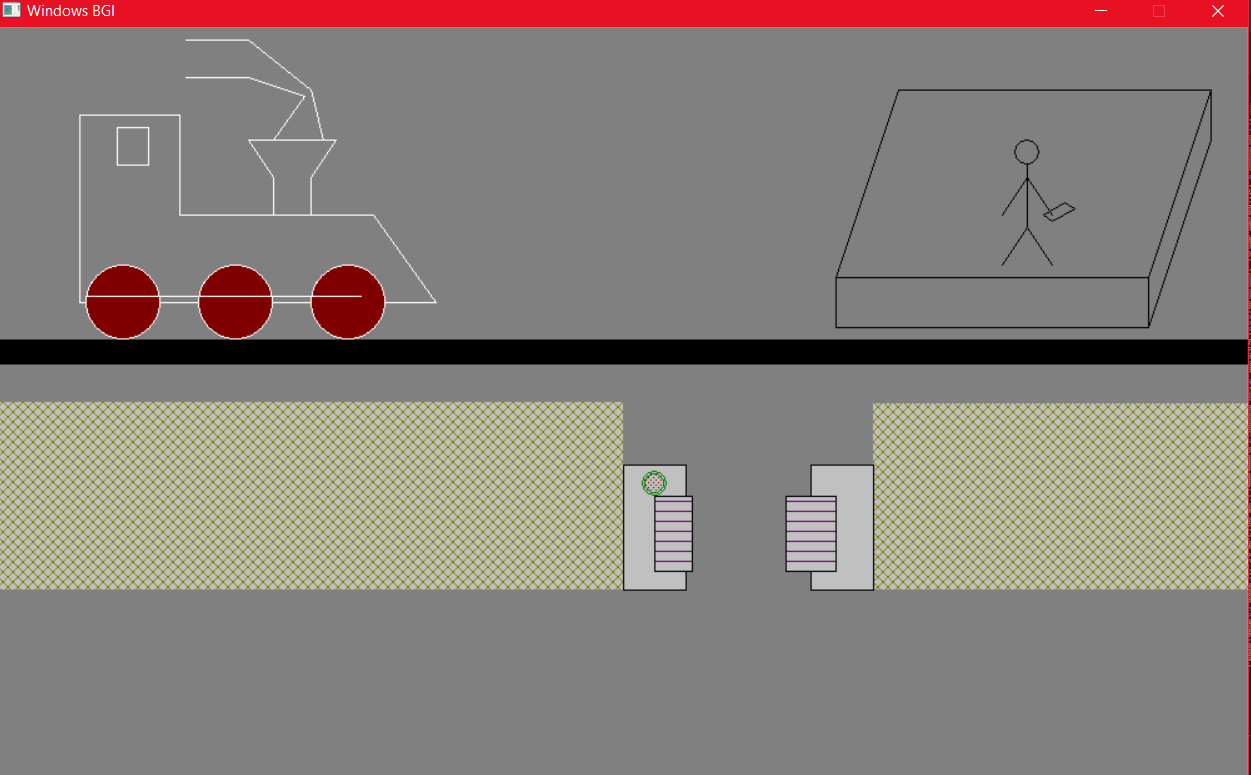
Пассажир стоит перед входом железнодорожной станции.

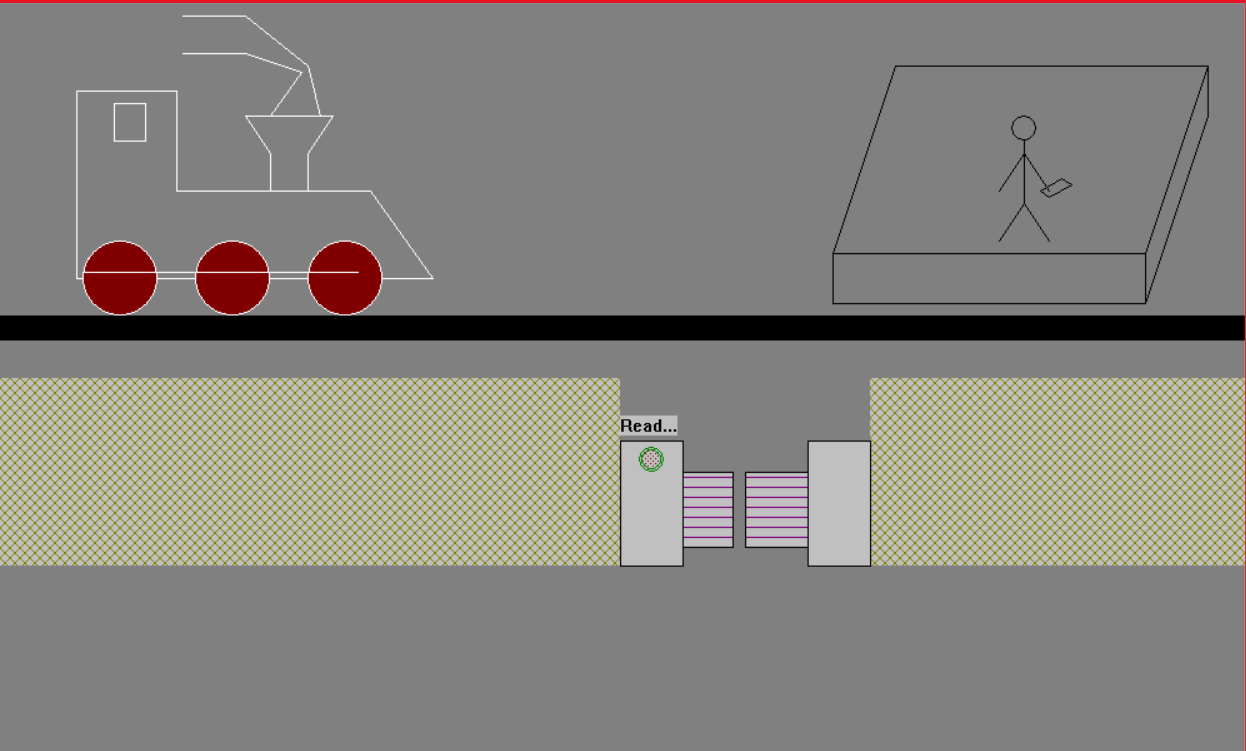
В первой сцене человек поднимает руку с транспортной картой и прикладывает ее к валидатору. Идет процесс считывания данных с карты и списание денежной валюты для прохода на станцию. Баланс карты позволяет пройти через турникет, тогда пассажир приготовился пройти через турникет, с открывающимися дверками.

Во второй сцене показано изображение пассажира, находящегося на железнодорожной платформе и ожидающего поезда. В качестве успеха списания некоторой суммы денег с транспортной карты, через динамики турникета играет спокойная мелодия.

# Результаты работы программы







# Используемая литература

* Документация к библиотекам graphics.h и dos.h <http://mycpp.ru/cpp/scpp/cppf_settextstyle.htm>
* Онлайн справочник программиста <http://mycpp.ru/>

# Вывод

В ходе курсовой работы были разработаны программы и подпрограммы, воспроизводящие звуковую мультимедиапанораму. Для воспроизведения графики использована библиотека graphics.h, для воспроизведения звуков - библиотека dos.h.

Была использована среда разработки Visual Studio Code потому, что это легкая по весу и быстрая по времени компиляции среда, подходящая для обучения студентов профессиональным методам решения задач. Так же данная IDE максимально проста для понимания и удобна в использовании.

Анализируя код программы, можно прийти к выводу, что было бы рационально оптимизировать запись и чтение из бинарного файла, продумав единый прототип для функций, записывающих и считывающих значения координат для отрисовки. Для этого в листинге программы необходимо определить однотипные входные аргументы и задать выполнение логики на запись или чтение значений. Выполнив, эту оптимизацию, можно увеличить скорость выполнения программы и сократить количество строк кода. В остальном программа соответствует отсутствию избыточности и дублирования кода. Размерность типа данных с небольшим запасом соответствует принимаемым значениям переменных при выполнении программы и гарантирует целостность переменных.

При написании программы использовался функциональная декомпозиция задачи − каждая процедура является самостоятельной подзадачей, совокупность данных процедур реализуют поставленную задачу. Код легко поддается редактированию, добавлению или удалению отдельных функций благодаря тому, что функции связаны между собой только по входным и входным значениям. В результате при смене графического пакета graphics.h на любой другой, потребуется минимум усилий и времени для данного интеграционного перехода. Так необходимо будет проверить соответствие между требуемыми входными параметрами интегрируемых функций и существующими входными параметрами, а затем применить автозамену названий текущих функций графической библиотеки, на новые, соблюдая соответствие входных аргументов. Таким образом код программы является легко адаптируемым для смены графической библиотеки на другую, потому что в этом случае в программе потребуется изменить только часть, отвечающую за отрисовку.

Чтобы дополнить мультипликацию новой мультимедиа панорамой, потребуется в функции main добавить таймер или флаг поиска, который при чтении бинарного файла считает N-ую диаграмму. Пока таймер работает или по нахождению флага новой мультимедиа панорамы, осуществляется поиск в входном двоичном файле, по прошествии выделенного времени, поиск считается завершенным. При этом на си последовательно ищем и воспроизводим найденную диаграмму мультипликации, а при использовании с++ появляется возможность параллельного поиска и воспроизведения панорамы. В случае параллельного поиска и проигрыша следует сделать две подпрограммы: одна ищет графические диаграммы, а вторая воспроизводит их.