Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное   
учреждение высшего образования

Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий, математики и механики

**Отчет по лабораторной работе**

**«Билетная касса»**

**Выполнил**:

студент группы 381803-1

Мишин И.А.

**Проверил**:

доцент кафедры МОСТ, к.т.н.,

Сысоев А.В.

Нижний Новгород

2018

Содержание

[Введение 3](#_Toc532852869)

[1. Постановка задачи 4](#_Toc532852870)

[2. Руководство пользователя 5](#_Toc532852871)

[3. Руководство программиста 7](#_Toc532852872)

[3.1. Описание структуры программы 7](#_Toc532852873)

[3.2. Описание алгоритмов 8](#_Toc532852874)

[4. Результаты экспериментов 9](#_Toc532852875)

[Заключение 11](#_Toc532852876)

[Литература 12](#_Toc532852877)

[Приложение 13](#_Toc532852878)

# Введение

Каждый человек хоть раз бывал в кинотеатре и покупал билет на сеанс. В данной лабораторной работе представлен вариант реализации работы систем покупки билета.

# Постановка задачи

# Требуется написать связанную систему классов реализующую работу билетной кассы.

Обязательные требования:

# Класс Билетная касса должен имитировать работу кассы по продаже билетов на киносеансы в многозальном кинотеатре. Считать, что продажа билетов проводится на сеансы в пределах трех дней от текущей даты. Каждый сеанс описывается датой, временем начала сеанса, названием фильма, номером зала, стоимостью билета в зависимости от зоны (VIP и обычная). Для упрощения считать, что покупатель указывает тип зоны и требуемое число билетов, а места выделяются кассой автоматически (при наличии свободных). Зрительные места в каждом зале описываются номером ряда и номером в ряду. Для упрощения считать, что число мест во всех рядах в одном зале одинаково. Продажа билетов на сеанс прекращается через 10 минут после начала сеанса.

# Информация о всех сеансах на ближайшие 30 дней проката хранится в классе Кинотеатр. Для каждого зала установлена базовая стоимость билетов (на дневные сеансы – от 12.00 до 18.00). Стоимость билетов на утренние сеансы (до 12.00) составляет 75% от базовой, стоимость билетов на вечерние сеансы (после 18.00) – 150% от базовой. Информация о зрительных местах (свободно/занято) в каждом зале на каждом сеансе также хранится в классе Кинотеатр.

# Класс Билетная касса должен предоставлять следующие операции: 1) принять данные покупателя: дату, время сеанса, название фильма, номер зала, тип зоны, число мест, 2) проверить наличие требуемого количества свободных мест в требуемой зоне, 3) зарезервировать требуемое количество мест, 4) рассчитать общую стоимость билетов, 5) отменить заказ билетов, 6) сформировать билеты (каждый билет включает: дату, время сеанса, название фильма, номер зала, номер ряда, номер места в ряду).

# Руководство пользователя

В данной программе пользовательский интерфейс не разрабатывался. В следующем пункте будет описана классовая структура программы.

# Руководство программиста

## Описание структуры программы

Программа содержит в себе 4 класса: *Cinema* (содержит информацию о залах и сеансах), *TicketWindow* (реализует работу билетной кассы), *Seance* (описывает сеанс), *Hall* (описывает зал).

Разберем подробнее структуру каждого класса.

1. Hall

Класс содержит в себе две структуры: *DefaultZone* и *VIPZone*, каждая из которых описывается количеством рядов, мест в ряду и стоимостью места. Причем необходимо заметить, что VIP зоной считается некоторая часть мест в центре зрительного зала, а обычными местами все вокруг этой зоны.

Также класс содержит свое представление в виде двухмерной матрицы, количество свободных мест в премиум и обычной зоне.

При создании объекта зал требуются все параметры, для двух структур. Во время создания объекта вызывается метод *–Init()*, который создает динамический двумерный массив, который описывает зал (на свободных местах в обычной зоне “0”, на свободных в премиальной – “2”, занятое место будет помечаться “1”). Класс имеет два важных метода *-BuyTicket(int type, int &row, int &seat)* (покупка билета) и *-ReturnTicket(vector<int> rows, vector<int> cols, int type)* (возврат билета).

1. Seance

Содержит в себе структуру Ticket (полная информация о приобретенном билете). Каждый сеанс описывается датой, названием и временем. Также каждый сеанс содержит свой собственный зал, с которым может взаимодействовать. Соответственно каждое поле имеет метод, возвращающий его значение.

1. Cinema

Класс содержит два динамических массива: сеансы и залы. При начале работы программы этот класс должен считать данные из файлов. Для этого создано два метода update(), и также два метода, возвращающие эти два массива, для сеансов возвращаются только сеансы на ближайшие три дня.

1. TicketWindow

Основной класс программы. При создании объекта класса запрашиваются ближайшие сеансы. Реализовано два метода: покупка и возврат билета, который обрабатывают данную покупателем информацию.

## Описание алгоритмов

В данной программе реализовано взаимодействие четырех классов с помощью агрегаций и публичных методов.

Hall

Seance

TicketWindow

Cinema

# Заключение

Поставленная задача была выполнена. Реализована программа, позволяющая имитировать работу билетной кассы, минимальные требования к классам также были соблюдены.

# Литература

1. Николай Джосьютис: C++ Стандартная библиотека. 2006

# Приложение

Режимы работы:

void mode1()

{

void(\*funcs[N])(double x , double accuracy, int n\_elements) = { my\_sin, my\_cos, my\_exp, my\_arcsin, my\_tg };

char \*funcNames[N] = { "sin", "cos", "exp", "arcsin", "tg" };

char enteredFunc[10];

double accuracy = 0.0002, x = 0;

int i = N, n\_elements = 1000;

while (i == N)

{

system("cls");

printf("Enter your task(Example: sin 3.14): ");

scanf("%s %lf", enteredFunc, &x);

printf("\nEnter accuracy (>= 0.000001): ");

scanf("%lf", &accuracy);

printf("\nEnter number of elements of a series(between 1 and 1000): ");

scanf("%d", &n\_elements);

printf("\nYou enter: %s %lf\nAccuracy: %lf\nNumber of elements: %d\n", enteredFunc, x, accuracy, n\_elements);

for (i = 0; i < N; i++)

if (!strcmp(funcNames[i], enteredFunc))

{

funcs[i](x, accuracy, n\_elements + 1);

break;

}

if (i == N)

{

printf("Incorrect function.\nPress any button...");

\_getch();

}

}

}

void mode2()

{

void(\*funcs[N])(double x, int n\_experiments) = { my\_sin, my\_cos, my\_exp, my\_arcsin, my\_tg};

char \*funcNames[N] = { "sin", "cos", "exp", "arcsin", "tg" };

char enteredFunc[10];

double x = 0;

int i = N, n\_experiments = 25;

while (i == N)

{

system("cls");

printf("Enter your task(Example: sin 3.14): ");

scanf("%s %lf", enteredFunc, &x);

printf("\nEnter number of experiments(between 1 and 25): ");

scanf("%d", &n\_experiments);

printf("\nYou enter: %s %lf\nNumber of experiments: %d\n", enteredFunc, x, n\_experiments);

for (i = 0; i < N; i++)

if (!strcmp(funcNames[i], enteredFunc))

{

funcs[i](x, n\_experiments + 1);

break;

}

if (i == N)

{

printf("Incorrect function.\nPress any button...");

\_getch();

}

}

}

Вспомогательные функции:

double power(double x, int deg)

{

double result = 1;

for (int i = 1; i <= deg; i++)

result \*= x;

return result;

}

double combinations(int n, int k)

{

double result = 1;

for (unsigned i = 1; i <= n-k; i++)

result \*= double(k + i) / double(i);

return result;

}

double bernulli(int n)

{

if (n > 255)

n = 255;

double bernull[256] = {0};

bernull[0] = 1.;

for (int i = 1; i <= n; i++)

{

for (int j = 1; j <= i; j++)

bernull[i] += combinations(i + 1, j + 1) \* bernull[i - j];

bernull[i] \*= (-1.) / double(i + 1);

}

return bernull[n];

}

Функции подсчета рядов:

void my\_sin(double x, double accuracy, int n\_elements)

{

int i;

double result = 0, middleres;

double machineResult = sin(x \* M\_PI / 180);

printf("\nReference value: %lf", machineResult);

x \*= M\_PI / 180;

middleres = x;

for (i = 1; i < n\_elements; i++)

{

result += middleres;

middleres \*= ((-1) \* power(x,2)) / (double(2 \* i) \* double(2\*i+1));

if (fabs(result - machineResult) < accuracy) break;

}

printStats(result, fabs(result - machineResult), i - 1);

\_getch();

}

void my\_sin(double x, int n\_experiments)

{

x \*= M\_PI / 180;

int i;

double result = 0., middleres = x;

double machineResult = sin(x);

printf("\nReference value: %lf", machineResult);

for (i = 1; i < n\_experiments; i++)

{

result += middleres;

middleres \*= ((-1.) \* power(x, 2)) / (double(2 \* i) \* double(2 \* i + 1));

printStats(i, result, fabs(result - machineResult));

}

\_getch();

}

void my\_cos(double x, double accuracy, int n\_elements)

{

x \*= M\_PI / 180;

int i;

double result = 0., middleres = 1.;

double machineResult = cos(x);

printf("\nReference value: %lf", machineResult);

for (i = 1; i < n\_elements; i++)

{

result += middleres;

middleres \*= ((-1.) \* power(x, 2)) / (double(2 \* i) \* double(2 \* i - 1));

if (fabs(result - machineResult) < accuracy) break;

}

printStats(result, fabs(result - machineResult), i - 1);

\_getch();

}

void my\_cos(double x, int n\_experiments)

{

x \*= M\_PI / 180;

int i;

double result = 0., middleres = 1.;

double machineResult = cos(x);

printf("\nReference value: %lf", machineResult);

for (i = 1; i < n\_experiments; i++)

{

result += middleres;

middleres \*= ((-1) \* power(x, 2)) / (double(2 \* i) \* double(2 \* i - 1));

printStats(i, result, fabs(result - machineResult));

}

\_getch();

}

void my\_exp(double x, double accuracy, int n\_elements)

{

int i;

double result = 1., middleres = x;

double machineResult = exp(x);

printf("\nReference value: %lf", machineResult);

for (i = 1; i < n\_elements; i++)

{

result += middleres;

middleres \*= x / double(i + 1);

if (fabs(result-machineResult) < accuracy) break;

}

printStats(result, fabs(result - machineResult), i - 1);

\_getch();

}

void my\_exp(double x, int n\_experiments)

{

int i;

double result = 1., middleres = x;

double machineResult = exp(x);

printf("\nReference value: %lf", machineResult);

for (i = 1; i < n\_experiments; i++)

{

result += middleres;

middleres \*= x / double(i + 1);

printStats(i, result, fabs(result - machineResult));

}

\_getch();

}

void my\_tg(double x, double accuracy, int n\_elements)

{

x \*= M\_PI / 180;

int i;

double machineResult = tan(x);

printf("\nReference value: %lf", machineResult);

double resultcos = 0., middlerescos = 1., resultsin = 0., middleressin = x;

double machineResultsin = sin(x), machineResultcos = cos(x);

for (i = 1; i < n\_elements; i++)

{

resultcos += middlerescos;

middlerescos \*= ((-1.) \* power(x, 2)) / (double(2 \* i) \* double(2 \* i - 1));

resultsin += middleressin;

middleressin \*= ((-1.) \* power(x, 2)) / (double(2 \* i) \* double(2 \* i + 1));

if (fabs(resultsin - machineResultsin) < accuracy && fabs(resultcos - machineResultcos) < accuracy) break;

}

printf("\nValue of sin/cos : %lf\nAccuracy: %lf\n", resultsin / resultcos, fabs(resultsin / resultcos - machineResult));

double result = 0, middleres1 = 1., middleres2 = -1.;

middleres1 /= x;

middleres2 /= x;

for (i = 1; i < n\_elements; i++)

{

double berni = bernulli(2 \* i) / bernulli (2 \* i - 2);

middleres1 \*= ((-4.) \* power(x, 2) \* berni) / (double(2 \* i) \* double(2 \* i - 1));

middleres2 \*= ((-16.) \* power(x, 2) \* berni) / (double(2 \* i) \* double(2 \* i - 1));

result += middleres1 + middleres2;

if (fabs(result - machineResult) < accuracy)

break;

}

printStats(result, fabs(result - machineResult), i - 1);

\_getch();

}

void my\_tg(double x, int n\_experiments)

{

x \*= M\_PI / 180;

int i;

double result = 0, middleres1 = 1., middleres2 = -1.;

double machineResult = tan(x);

printf("\nReference value: %lf", machineResult);

middleres1 /= x;

middleres2 /= x;

for (i = 1; i < n\_experiments; i++)

{

double berni = bernulli(2 \* i) / bernulli(2 \* i - 2);

middleres1 \*= ((-4.) \* power(x, 2) \* berni) / (double(2 \* i) \* double(2 \* i - 1));

middleres2 \*= ((-16.) \* power(x, 2) \* berni) / (double(2 \* i) \* double(2 \* i - 1));

result += middleres1 + middleres2;

printStats(i, result, fabs(result - machineResult));

}

\_getch();

}

void my\_arcsin(double x, double accuracy, int n\_elements)

{

int i;

double result = x, middleres = x;

double machineResult = asin(x) \* 180 / M\_PI;

printf("\nReference value: %lf", machineResult);

for (i = 1; i < n\_elements; i++)

{

middleres \*= ((double(2 \* i) \* double(2 \* i - 1) \* power(x, 2)) / ((4.) \* power(double(i), 2) \* (2 \* i + 1)));

result += middleres;

middleres /= (2 \* i + 1);

if (fabs(result \* 180 / M\_PI - machineResult) < accuracy) break;

}

result \*= 180 / M\_PI;

printStats(result, fabs(result - machineResult), i - 1);

\_getch();

}

void my\_arcsin(double x, int n\_experiments)

{

int i;

double result = x, middleres = x;

double machineResult = asin(x) \* 180 / M\_PI;

printf("\nReference value: %lf", machineResult);

for (i = 1; i < n\_experiments; i++)

{

middleres \*= ((double(2 \* i) \* double(2 \* i - 1) \* power(x, 2)) / ((4.) \* power(double(i), 2) \* (2 \* i + 1)));

result += middleres;

middleres /= (2 \* i + 1);

printStats(i, result \* 180 / M\_PI, fabs(result \* 180 / M\_PI - machineResult));

}

\_getch();

}