Содержание

Введение

Объектно-ориентированное программирование (ООП): C++ является полностью объектно-ориентированным языком, что позволяет создавать модульные и масштабируемые программы. C++ поддерживает статическую типизацию, что позволяет обнаруживать большинство ошибок типов во время компиляции, что делает программы более надежными и простыми в отладке.

Синтаксис C++ обладает богатым и выразительным синтаксисом, что делает его привлекательным для работы и позволяет создавать эффективный и гибкий код. Механизм виртуализации: C++ поддерживает механизм виртуализации, что позволяет реализовать полиморфизм, обеспечивая возможность использовать базовые классы в качестве производных классов. Встраиваемые функции в C++ поддерживает встраиваемые функции, что позволяет улучшить производительность программы, встраивая короткие функции непосредственно в место их вызова. Выделение памяти в C++ предоставляет различные способы выделения памяти, включая статическое, динамическое и автоматическое выделение памяти.

STL: C++ включает в себя Standard Template Library, которая предоставляет широкий набор готовых к использованию контейнеров, итераторов и алгоритмов, что значительно упрощает разработку программ.

В целом, C++ обладает большим набором возможностей и предоставляет мощные инструменты для разработки качественных и производительных программ и позволяет провести моделирования деятельности предприятия.

1 Постановка задачи

Разработать модель работы предприятия в заданной предметной области (согласно варианту). Спроектировать иерархию классов, согласно предметной области. При разработке модели предусмотреть выполнение следующих задач:

1. создать текстовый файл, содержащий объекты предметной области (не менее 30 объектов);
2. обеспечить вывод информации по имеющимся объектам в контейнер (согласно варианту – vector) объектно-ориентированного приложения;
3. провести моделирование деятельности предприятия в течении указанного в варианте периода, объекты деятельности должны выбираться из контейнера случайным образом;
4. вывести в текстовый файл результат работы предприятия в указанный период по дням (по часам);
5. определить указанные в варианте характеристики предприятия или его объектов в соответствии с вариантом задания.

Предметная область: антикафе.

Предприятие: антикафе "Уютный Уголок", предоставляющее услуги аренды столов для посетителей.

Объекты предметной области: столики, чеки, менеджеры.

Период деятельности предприятия: 1 день.

Задания, согласно варианту 17:

1) Определить, какой из менеджеров обслужил больше всего столов.

2) Определить, какой из столиков наиболее востребован.

3) Определить количество компаний, посетивших антикафе за день.

4) Определить выручку антикафе за один рабочий день.

Основные принципы работы антикафе:

Столик имеет статус "свободен" или "занят". В случае, когда он занят, регистрируется информация о посетителе или группе.

Менеджер управляет столиками, обрабатывает запросы на бронирование, предоставляет столики клиентам, производит расчет и освобождает столы после ухода посетителей.

Посетитель может занимать свободные столики, заказывать дополнительные услуги (чай, кофе, печенье), бронировать столики на определенное время и освобождать их по окончании посещения.

2 Описание классов

CafeConstants: Статический класс, предоставляющий константы для параметров работы антикафе.

```
class CafeConstants {
    public:
        static const int NUM_MANAGERS = 3;\\ кол-во менеджеров
        static const int NUM_ZONES = 3; // количество залов
        static const int TABLES_PER_ZONE = 3; // кол-во столов в
    каждом зале
        static const int CAFE_OPEN_TIME = 9 * 60; // время
    открытия кафе
        static const int CAFE_CLOSE_TIME = 23 * 60; // время
    закрытия кафе
        static const int HOUR_RATE = 200; // стоимость времени в
    кафе
};
```

Table: Класс, представляющий отдельный стол в антикафе. Содержит номер стола, зону расположения, флаги занятости и использования в данный момент времени.

```
class Table {
    public:
       int number; // номер стола
       int zone; // номер зала
       bool occupied; // индикатор занятия стола
};
```

Check: Класс, представляющий чек за использование стола. Включает в себя время начала и окончания использования, номер зоны и стола, имя менеджера, подсчёт количества посетителей, список заказов и сумму к оплате.

```
class Check {
    public:
        int startTime; // время начала работы в чеке
        int endTime; // время завершения работы в чеке
        int zone; // зал
        string managerName; // имя менеджера
        int tableNumber; // номер стола
        int numPeople; // кол-во людей в группе
        vector<string> orders; // вектор заказов
        double payment; // оплата
};
```

Manager: Класс, представляющий менеджера. Содержит имя, список чеков (Check) и общую выручку, полученную от обслуживания столов.

```
class Manager {
    public:
        string name; // имя
        vector<Check> checks; // вектор чеков
        double totalRevenue; // общая выручка за день
};
```

3 Программное конструирование

Код программы включает в себя определения вышеупомянутых классов и основную логику работы антикафе, приведенную в функции main(). В результате выполнения курсовой работы реализованы следующие функции:

- Инициализация менеджеров и столов.
- Проведение процесса обслуживания посетителей менеджерами.
- Расчёт и вывод статистики по каждому чеку.
- Подсчёт выручки от каждого стола.
- Определение менеджера, обслужившего наибольшее количество столов.
- Определение наиболее используемого столика.
- Запись итоговых данных как в консоли, так и в файл для отчетности.

Текст программы:

```
#include <iostream>
#include <vector>
#include <ctime>
#include <cstdlib>
#include <iomanip>
#include <unordered_set>
#include <limits>
#include <string>
#include <fstream>

using namespace std;

class CafeConstants {
public:
    static const int NUM_MANAGERS = 3;
    static const int NUM_ZONES = 3;
```

```
    static const int TABLES_PER_ZONE = 3;
    static const int CAFE_OPEN_TIME = 9 * 60;
    static const int CAFE_CLOSE_TIME = 23 * 60;
    static const int HOUR_RATE = 200;
};
class Table {
public:
    int number;
    int zone;
    bool occupied;
    bool inUse;
};
class Check {
public:
    int startTime;
    int endTime;
    int zone;
    string managerName;
    int tableNumber;
    int numPeople;
    vector<string> orders;
    double payment;
};
class Manager {
public:
    string name;
    vector<Check> checks;
    double totalRevenue;
};

void printCheck(const Check& check) {
    cout << setfill('-') << setw(40) << "" << setfill(' ') <<
endl;
    cout << setw(20) << left << "| Guests arrived: " <<
setw(19) << right << check.numPeople << endl;
```

```
    cout << setw(20) << left << "| Hall: " << setw(19) <<
right << check.zone << endl;
    cout << setw(20) << left << "| Table: " << setw(19) <<
right << "Table " << check.tableNumber << endl;
    cout << "| Orders: ";
    for (const string& order : check.orders) {
        cout << order << " ";
    }
    cout << setw(18) << right << endl;
    cout << setw(20) << left << "| Start time: " << setw(18)
<< right << check.startTime / 60 << ":" << check.startTime %
60 << endl;
    int exitHours = check.endTime / 60 % 24;
    int exitMinutes = check.endTime % 60;
    cout << setw(20) << left << "| Exit time: " << setw(18)
<< right << exitHours << ":" << exitMinutes << endl;
    cout << setw(20) << left << "| Payment: " << setw(18) <<
right << check.payment << " rubles " << endl;
    cout << setfill('-') << setw(40) << "" << setfill(' ') <<
endl;
}

int main() {
    //----------------------Вывод в файл-------------------

    time_t t = time(0);
    struct tm now;
    localtime_s(&now, &t);
    char filename[80];
    strftime(filename,   sizeof(filename),   "%d.%m.%Y___%H-
%M_check.txt", &now);
    // Create ofstream for writing to the file
    ofstream outputFile(filename);
    // Redirect cout to the file
```

```
    streambuf* coutbuf = cout.rdbuf(); // Save the original
cout buffer
    cout.rdbuf(outputFile.rdbuf());     // Redirect cout to
the file
//----------------------Вывод в файл-----------------------
    srand(static_cast<unsigned int>(time(0)));
    vector<Manager> managers(CafeConstants::NUM_MANAGERS);
    vector<Table>     tables(CafeConstants::NUM_ZONES     *
CafeConstants::TABLES_PER_ZONE);
    for (int i = 0; i < CafeConstants::NUM_MANAGERS; ++i) {
        managers[i].name = "Manager " + to_string(i + 1);
        managers[i].totalRevenue = 0;
    }
    for  (int  i  =  0;  i  <  CafeConstants::NUM_ZONES  *
CafeConstants::TABLES_PER_ZONE; ++i) {
        tables[i].zone = i / CafeConstants::TABLES_PER_ZONE
+ 1;
        tables[i].number          =          i          %
CafeConstants::TABLES_PER_ZONE + 1;
        tables[i].occupied = false;
        tables[i].inUse = false;
    }
    int currentTime = CafeConstants::CAFE_OPEN_TIME;
    int managerCalls[CafeConstants::NUM_MANAGERS] = { 0 };
    int        tableUsage[CafeConstants::NUM_ZONES        *
CafeConstants::TABLES_PER_ZONE] = { 0 };
    double totalRevenue = 0;
    //-------------------------------
    // Итоги:
    // задание 1
    int maxManagerGenerations = 0;
    string mostFrequentManager;
    // задание 2
    int maxTableGenerations = 0;
    int mostFrequentTableNumber = 0;
```

```
    // задание 3
    int numGroupsGenerated = rand() % 11 + 90;
    //------------------------------
    unordered_set<int> usedTables;
    while (currentTime <= CafeConstants::CAFE_CLOSE_TIME) {
        int numGroups = rand() % 5 + 1;
        for (int group = 0; group < numGroups; ++group) {
            int numPeople = rand() % 8 + 1;
            int zone = rand() % CafeConstants::NUM_ZONES + 1;
            // вектор для хранения свободных столиков
            vector<int> availableTables;
            for (int i = 0; i < CafeConstants::TABLES_PER_ZONE; ++i) {
                int tableNumber = (zone - 1) * CafeConstants::TABLES_PER_ZONE + i + 1;
                // проверка не занят ли стол
                if (!tables[tableNumber - 1].inUse && usedTables.find(tableNumber) == usedTables.end()) {
                    availableTables.push_back(tableNumber);
                }
            }
            if (!availableTables.empty()) {
                // Случайно генерируется доступный стол
                int randomTableIndex = rand() % availableTables.size();
                int selectedTableNumber = availableTables[randomTableIndex];
                int tableIndex = selectedTableNumber - 1;
                tables[tableIndex].inUse = true;
                usedTables.insert(selectedTableNumber);
                Check check;
                check.startTime = currentTime;
                check.numPeople = numPeople;
                check.zone = zone;
                check.tableNumber = selectedTableNumber;
```

```
                int numCoffees = rand() % 3 + 1;
                int numTeas = rand() % 3 + 1;
                int numCookies = rand() % 2 + 1;
                int numMarmalades = rand() % 2 + 1;
                for (int i = 0; i < numCoffees; ++i) {
                    check.orders.push_back("Coffee");
                }
                for (int i = 0; i < numTeas; ++i) {
                    check.orders.push_back("Tea");
                }
                for (int i = 0; i < numCookies; ++i) {
                    check.orders.push_back("Cookie");
                }
                for (int i = 0; i < numMarmalades; ++i) {
                    check.orders.push_back("Marmalade");
                }
                check.endTime = currentTime + (numCoffees +
numTeas + numCookies + numMarmalades) * 15;
                check.payment = (check.endTime -
check.startTime) / 60.0 * CafeConstants::HOUR_RATE *
numPeople;
                int randomManagerIndex = rand() %
CafeConstants::NUM_MANAGERS;
                check.managerName =
managers[randomManagerIndex].name;

managers[randomManagerIndex].checks.push_back(check);
                managers[randomManagerIndex].totalRevenue +=
check.payment; // добавление к общему доходу менеджера
                totalRevenue += check.payment; // Добавление
к общему доходу
                // обновляет столы и количество вызовов
менеджера.
                managerCalls[randomManagerIndex]++;
                tableUsage[selectedTableNumber - 1]++;
```

```
                // случайно выбирается менеджер
                cout << "Time: " << currentTime / 60 << ":"
<< setw(2) << setfill('0') << currentTime % 60 << setfill('
') << endl;
                cout      <<      "Manager:      "      <<
managers[randomManagerIndex].name << endl;
                printCheck(check);
                tables[tableIndex].inUse = false;
                usedTables.erase(selectedTableNumber);
            }
        }
        int randomOffset = (rand() % 12) * 5 + 5;
        currentTime += randomOffset;
    }
    // задание 1
    for (int i = 0; i < CafeConstants::NUM_MANAGERS; ++i) {
        if (managerCalls[i] > maxManagerGenerations) {
            maxManagerGenerations = managerCalls[i];
            mostFrequentManager = "Manager " + to_string(i +
1);
        }
    }
    // задание 2
    for  (int  i  =  0;  i  <  CafeConstants::NUM_ZONES  *
CafeConstants::TABLES_PER_ZONE; ++i) {
        if (tableUsage[i] > maxTableGenerations) {
            maxTableGenerations = tableUsage[i];
            mostFrequentTableNumber = i + 1;
        }
    }
    //--------------------task1 & 2 end----------------
    //--------------------доп задания----------------
    cout << " " << endl;
    cout << " " << endl;
```

```
    cout << "+-----------------------------------------------
-------------------------+" << endl;
    cout << "|                                        Dop. zadaniya
|" << endl;
    cout << "+-----------------------------------------------
-------------------------+" << endl;
    cout << "| 1) The most frequent manager: " <<
mostFrequentManager << " generated " << maxManagerGenerations
<< " times." << endl;
    cout << "| 2) The most frequent table number: Table " <<
mostFrequentTableNumber << " was generated " <<
maxTableGenerations << " times." << endl;
    cout << "| 3) Total revenue: " << totalRevenue << "
rubles." << endl;
    cout << "| 4) Number of groups generated: " <<
numGroupsGenerated << "." << endl;
    cout << "+-----------------------------------------------
-------------------------+" << endl;

    //----------------------Вывод в файл--------------------
------------

            // Close the file and restore cout
    outputFile.close();
    cout.rdbuf(coutbuf); // Reset cout to the original buffer

//----------------------Вывод в файл-------------------------
--------

    return 0;
}
```

4 Тестовые примеры

Тестовый пример 1:

```
Time: 9:00
Manager: Manager 1
----------------------------------------
| Guests arrived:                    3
| Hall:                              3
| Table:                         Table 9
| Orders: Coffee Coffee Coffee Tea Tea Cookie Marmalade
| Start time:                        9:0
| Exit time:                       10:45
| Payment:                      1050 rubles
----------------------------------------
Time: 9:00
Manager: Manager 1
----------------------------------------
| Guests arrived:                    4
| Hall:                              3
| Table:                         Table 7
| Orders: Coffee Tea Tea Cookie Cookie Marmalade Marmalade
| Start time:                        9:0
| Exit time:                       10:45
| Payment:                      1400 rubles
----------------------------------------
Time: 9:00
Manager: Manager 3
----------------------------------------
| Guests arrived:                    7
| Hall:                              2
| Table:                         Table 4
| Orders: Coffee Tea Tea Tea Cookie Cookie Marmalade
| Start time:                        9:0
| Exit time:                       10:45
| Payment:                      2450 rubles
```

• • • (и т.д.) • • •

```
Time: 22:20
Manager: Manager 1
----------------------------------------
| Guests arrived:                      5
| Hall:                                3
| Table:                         Table 8
| Orders: Coffee Coffee Coffee Tea Tea Cookie Marmalade
| Start time:                      22:20
| Exit time:                        0:5
| Payment:                      1750 rubles
----------------------------------------
Time: 22:20
Manager: Manager 1
----------------------------------------
| Guests arrived:                      1
| Hall:                                1
| Table:                         Table 3
| Orders: Coffee Coffee Coffee Tea Tea Tea Cookie Marmalade
| Start time:                      22:20
| Exit time:                        0:20
| Payment:                       400 rubles
----------------------------------------
```

```
+-------------------------------------------------------------+
|                         Dop. zadaniya                       |
+-------------------------------------------------------------+
| 1) The most frequent manager: Manager 1 generated 33 times.
| 2) The most frequent table number: Table 5 was generated 11
times.
| 3) Total revenue: 121800 rubles.
| 4) Number of groups generated: 99.
+-------------------------------------------------------------+
```

Тестовый пример 2:

```
Time: 9:00
Manager: Manager 2
----------------------------------------
| Guests arrived:                     3
| Hall:                               1
| Table:                          Table 3
| Orders: Coffee Tea Cookie Cookie Marmalade Marmalade
| Start time:                        9:0
| Exit time:                       10:30
| Payment:                        900 rubles
----------------------------------------
Time: 9:00
Manager: Manager 3
----------------------------------------
| Guests arrived:                     5
| Hall:                               1
| Table:                          Table 2
| Orders: Coffee Tea Tea Cookie Cookie Marmalade
| Start time:                        9:0
| Exit time:                       10:30
| Payment:                        1500 rubles
----------------------------------------
Time: 9:00
Manager: Manager 2
----------------------------------------
| Guests arrived:                     4
| Hall:                               2
| Table:                          Table 6
| Orders: Coffee Coffee Tea Cookie Cookie Marmalade Marmalade
| Start time:                        9:0
| Exit time:                       10:45
| Payment:                        1400 rubles
```

• • • (и т.д.) • • •

```
Time: 22:50
Manager: Manager 2
----------------------------------------
| Guests arrived:                    1
| Hall:                              2
| Table:                         Table 4
| Orders: Coffee Tea Cookie Marmalade
| Start time:                        22:50
| Exit time:                         23:50
| Payment:                         200 rubles
----------------------------------------
Time: 22:50
Manager: Manager 1
----------------------------------------
| Guests arrived:                    6
| Hall:                              3
| Table:                         Table 9
| Orders: Coffee Coffee Tea Tea Cookie Marmalade
| Start time:                        22:50
| Exit time:                          0:20
| Payment:                        1800 rubles
----------------------------------------


+------------------------------------------------------------------+
|                          Dop. zadaniya                           |
+------------------------------------------------------------------+
| 1) The most frequent manager: Manager 2 generated 30 times.
| 2) The most frequent table number: Table 6 was generated 14
times.
| 3) Total revenue: 137200 rubles.
| 4) Number of groups generated: 96.
+------------------------------------------------------------------+
```

Заключение

Разработана модель работы предприятия в заданной предметной области (по варианту). Спроектированы классы, согласно предметной области. При разработке модели предусмотрено выполнение следующих задач:

1. создание текстового файла, содержащего объекты предметной области (не менее 30 объектов);
2. обеспечен вывод информации по имеющимся объектам в контейнер (согласно варианту – vector) объектно-ориентированного приложения;
3. проведено моделирование деятельности предприятия в течении указанного в варианте периода, объекты деятельности выбираются из контейнера случайным образом;
4. результат работы предприятия в указанный период по дням (по часам) выводится в текстовый файл;
5. определены указанные в варианте характеристики предприятия или его объектов в соответствии с вариантом задания.

В процессе моделирования деятельности антикафе определено:

Какой из менеджеров обслужил больше всего столов.

Какой из столиков наиболее востребован.

Количество компаний, посетивших антикафе за день.

Выручка антикафе за один рабочий день.

Все поставленные в работе задачи выполнены в полном объеме.

Перечень использованных информационных ресурсов

1. Гайдышев, И. Решение научных и инженерных задач средствами Excel, VBA и C/C++ / И. Гайдышев.–М.:БХВ–Петербург, 2020.– 206 с.

2. Герберт, Шилдт C++. Базовый курс / Шилдт Герберт. – М.: Диалектика / Вильямс, 2022. – 564 с.

3. Дейтел, Пол Как программировать на C / Пол Дейтел , Харви Дейтел. – М.: Бином, 2022. – 858 с.

4. Джамса, К. Учимся программировать на языке C++ / К. Джамса. – М.: Мир, 2022. – 320 с.

5. Ишкова, Э. А. Изучаем C++ на задачах и примерах / Э.А. Ишкова. – М.: Наука и техника, 2018. – 240 с.

6. Кетков, Александр Практика программирования: Бейсик, Си, Паскаль. Самоучитель / Александр Кетков, Юлий Кетков.–М.: БХВ–Петербург, 2021. – 480 с.

7. Кузнецов, М.В. C++. Мастер–класс в задачах и примерах (+ CD–ROM) / М.В. Кузнецов. – М.: БХВ–Петербург, 2020. – 956 с.

8. Культин, Н. C/C++ в задачах и примерах / Н. Культин. – М.: БХВ–Петербург, 2022. – 368 с.

9. Культин, Н.Б. C/C++ в задачах и примерах (+ CD–ROM) / Н.Б. Культин. – М.: БХВ–Петербург, 2019. – 738 с.

10. Лав, Роберт Ядро Linux. Описание процесса разработки / Роберт Лав. – М.: Вильямс, 2019. – 496 с.

11. Лафоре, Роберт Объектно–ориентированное программирование в C++ / Роберт Лафоре. – М.: Питер, 2020. – 928 с.

12. Мартынов, Н. Н. Информатика. C для начинающих / Н.Н. Мартынов. – М.: КУДИЦ–Образ, 2022. – 304 с.