Министерство образования Республики Беларусь

Учреждение образования

БелорусскиЙ государственный университет

информатики и радиоэлектроники

Факультет инженерно-экономический

Кафедра экономической информатики

**Отчет к лабораторной работе №7**

**по теме**

**«АССОЦИАТИВНЫЕ КЛАССЫ-КОНТЕЙНЕРЫ БИБЛИОТЕКИ STL»**

|  |  |
| --- | --- |
| Выполнила: | Вдовенко Н. Д.  студент группы 272303 |
| Проверила: | Салапура М. Н.  ст. преподаватель |

Минск 2023

Общая постановка задачи

**Цель работы:** изучить ассоциативные классы-контейнеры библиотеки STL.

**Индивидуальное задание:**

Необходимо создать контейнеры, которые будут хранить объекты классов по предметной области «разработка игр». Для контейнера реализовать добавление, удаление, редактирование, вывод содержимого контейнера на экран и в файл, поиск и сортировку элементов. Необходимо создать удобное пользовательское меню. Необходимо реализовать работу с контейнерами map, set, multimap, multiset, unordered\_map, unordered\_set, unordered\_multimap, unordered\_multiset.

**Краткие теоретические сведения:**

Контейнеры — это объекты, которые содержат в себе набор (совокупность) других объектов.

Ассоциативные контейнеры кардинально отличаются от последовательных: элементы в ассоциативном контейнере хранятся и предоставляются по ключу. Элементы последовательного контейнера хранятся и предоставляются последовательно, по их позиции в контейнере.

Ассоциативные контейнеры (associative container) обеспечивают быстрый поиск и предоставление элементов по ключу. Двумя первичными типами ассоциативных контейнеров являются map (карта) и set (набор). Элементами контейнера map являются пары ключ-значение (keyvalue pair): ключ выступает в роли индекса, а значение представляет собой хранимые в контейнере данные. Контейнер set содержит только ключи и предоставляет эффективные способы запроса на проверку наличия определенного ключа. Set можно использовать, например, для хранения слов, которые следует проигнорировать при обработке текста. Map можно использовать для создания словаря: слово будет ключом, а его определение — значением для этого ключа.

Библиотека STL предоставляет ассоциативные контейнеры, которые различаются по трем параметрам: − они являются набором (set) или картой (map); − они требуют уникальных ключей или допускает их совпадение; − они хранят элементы упорядочено или нет. В именах контейнеров, допускающих совпадение ключей, присутствует слово multi; имена контейнеров, не упорядочивающих хранимые ключи начинаются со слова unordered.

Типы ассоциативных контейнеров:

− map (ассоциативный массив, хранящий пары ключ-значение);

− set (контейнер, в котором ключ является значением);

− multimap (карта, допускающая совпадение ключей);

− multiset (набор, допускающий совпадение ключей);

− unordered\_map (карта, организованная по хеш-функции);

− unordered\_set (набор, организованный по хеш-функции);

− unordered\_multimap (хешированная карта; ключи могут повторяться);

− unordered\_multiset (хешированный набор; ключи могут повторяться).

**Диаграмма классов:**

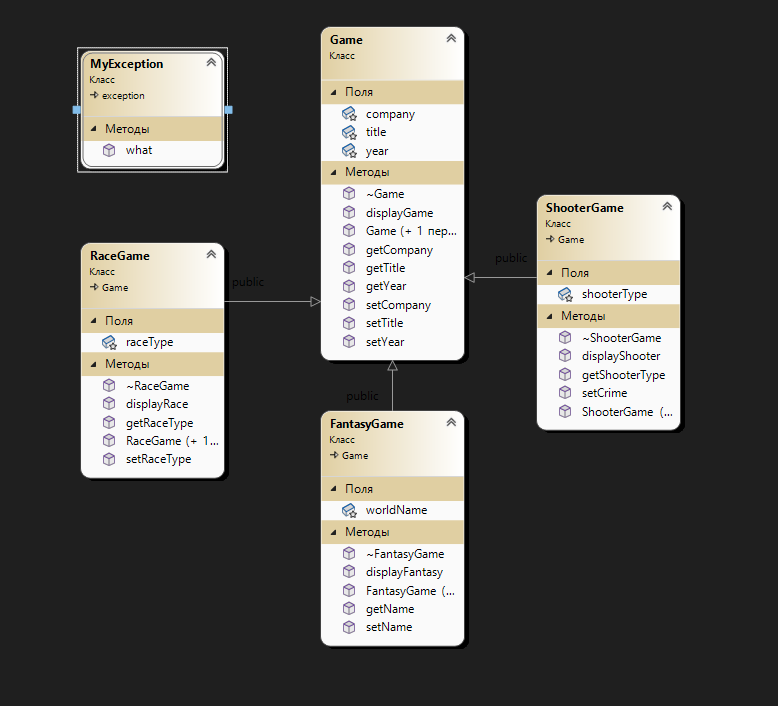


Рисунок 1 – Диаграмма классов

**Контрольные вопросы:**

1. Например, в языке Java имеется библиотека, аналогичная STL в C++, называемая Java Collections Framework. В C# существует пространство имен System.Collections, которое предоставляет структуры данных и алгоритмы, аналогичные функционалу STL. Кроме того, в Python есть различные встроенные структуры данных и библиотеки, такие как `collections` и `itertools`, которые обеспечивают сходный функционал STL (хотя концепция языка Python отличается от C++ и STL).
2. Шаблонный класс в языке программирования C++ представляет собой класс, который может работать с различными типами данных без необходимости создания отдельного класса для каждого типа. Шаблоны позволяют писать универсальный код.
3. STL предоставляет оптимизированные, проверенные временем структуры данных и алгоритмы, которые позволяют удобную и эффективную работу с данными. Вместо написания пользовательских функций и структур пользования стандартной библиотекой позволяет использовать готовые универсальные компоненты.
4. Итератор и указатель похожи в том, что оба предоставляют доступ к элементам коллекции. Однако итератор абстрагирует этот доступ от конкретной структуры данных, что позволяет использовать общие методы для перебора различных типов коллекций. Пример:

int\* ptr;

std::vector<int>::iterator it;

1. Метод `lower\_bound()` используется для поиска первого элемента, равного или большего, указанного значения в отсортированной последовательности. `upper\_bound()` находит первый элемент, больший указанного значения. Разница в том, что `lower\_bound()` возвращает позицию элемента в пределах последовательности, равного указанному значению, или итератор, указывающий на первый элемент, который больше указанного. `upper\_bound()` возвращает итератор, указывающий на первый элемент, который больше указанного значения в последовательности.
2. "Пара" в STL представляет структуру данных для хранения пары значений. Она широко используется для хранения двух значений различных типов вместе. Создание пары может быть сделано, например, с помощью конструктора `std::make\_pair`, либо напрямую передавая значения в конструктор `std::pair`.
3. `std::map` представляет отсортированное отображение, где каждый ключ связан со значением. `std::set` содержит отсортированный набор уникальных элементов.
4. `std::multimap` аналогичен `map`, но допускает хранение не уникальных ключей. `std::multiset` аналогичен `set`, но позволяет хранить не уникальные элементы.
5. `std::unordered\_map` и `std::unordered\_set` представляют аналоги `map` и `set`, но поддерживают хэшированные таблицы для реализации, что обеспечивает быстрое получение доступа к элементам.
6. std::unordered\_multimap` и `std::unordered\_multiset` представляют аналоги `multimap` и `multiset`, но также используют хэшированные таблицы и позволяют хранить не уникальные ключи или значения.

**Листинг кода:**

#include <list>

#include <iostream>

#include <string>

#include <vector>

#include <array>

#include <set>

#include <map>

#include <unordered\_set>

#include <windows.h>

#include <fstream>

#include <stdlib.h>

#include <conio.h>

#include <iomanip>

#include <Windows.h>

#define GAMESFILENAME "files\\games.txt"

using namespace std;

class Game {

protected:

string title;

string company;

int year;

public:

Game() {}

Game(string title, string company, int year)

{

this->title = title;

this->company = company;

this->year = year;

}

~Game() {}

string getTitle() const {

return title;

}

void setTitle(string newTitle) {

title = newTitle;

}

string getCompany() const {

return company;

}

void setCompany(string company) {

this->company = company;

}

int getYear() const {

return year;

}

void setYear(int newYear) {

year = newYear;

}

void displayGame()

{

cout << "\nНазвание: " << title << endl;

cout << "Издатель: " << company << endl;

cout << "Год: " << year << endl;

}

};

class RaceGame : public Game {

protected:

string raceType;

public:

RaceGame() : Game("", "", 0), raceType("") {}

RaceGame(string title, string company, int year, string raceType) :Game(title, company, year)

{

this->raceType = raceType;

}

string getRaceType() const {

return raceType;

}

void setRaceType(string raceType) {

this->raceType = raceType;

}

~RaceGame() {}

void displayRace()

{

cout << "\nИгры про гонки" << endl;

Game::displayGame();

cout << "Тип гонок: " << raceType << endl;

}

};

class ShooterGame : public Game {

protected:

string shooterType;

public:

ShooterGame() : Game("", "", 0), shooterType("") {}

ShooterGame(string title, string company, int year, string shooterType) :Game(title, company, year)

{

this->shooterType = shooterType;

}

string getShooterType() const {

return shooterType;

}

void setCrime(string newCrime) {

this->shooterType = shooterType;

}

~ShooterGame() {}

void displayShooter()

{

cout << "\nШутеры" << endl;

Game::displayGame();

cout << "Вид шутера: " << shooterType << endl;

}

};

class FantasyGame : public Game {

protected:

string worldName;

public:

FantasyGame() : Game("", "", 0), worldName("") {}

FantasyGame(string title, string author, int year, string worldName) :Game(title, company, year)

{

this->worldName = worldName;

}

string getName() const {

return worldName;

}

void setName(string worldName) {

worldName = worldName;

}

~FantasyGame() {}

void displayFantasy()

{

cout << "\nФантастика" << endl;

Game::displayGame();

cout << "Название мира: " << worldName << endl;

}

};

class MyException : public exception {

public:

const char\* what() const throw() {

return "Ошибка: Максимальное количество игр было достигнуто!";

}

};

class MyExceptionComputer : public exception

{

public:

const char\* what() const throw() {

return "Игр нет!";

}

};

void termFunc()

{

cout << "Сработала собственная функция завершения\n";

exit(-1);

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

const wchar\_t\* directoryPath = L"files";

if (!CreateDirectory(directoryPath, NULL)) {

if (GetLastError() == ERROR\_ALREADY\_EXISTS) {

std::wcout << L"Directory 'files' already exists.\n";

}

else {

std::cerr << "Error creating directory 'files'.\n";

}

}

else {

std::wcout << L"Directory 'files' created successfully.\n";

}

ofstream outfile(GAMESFILENAME);

if (outfile.is\_open()) {

outfile.close();

}

RaceGame\* race = NULL;

std::set<RaceGame\*> raceGames;

int current\_race = 0;

ShooterGame\* shooter = NULL;

multimap<int, ShooterGame\*> shooterGames;

int current\_shooter = 0;

FantasyGame\* fantasy = NULL;

unordered\_set<FantasyGame\*> fantasyGames;

int current\_fantasy = 0;

int choice;

do {

try {

cout << "\nМеню выбора" << endl;

cout << "1 - Добавить игру" << endl;

cout << "2 - Просмотр игр" << endl;

cout << "3 - Поиск игры" << endl;

cout << "4 - Удалить игру" << endl;

cout << "5 - Изменить игру" << endl;

cout << "6 - Сортировать игры" << endl;

cout << "7 - Сохранить в файл" << endl;

cout << "0 - Выйти" << endl;

while (true) {

cout << "Выберите действие: ";

cin >> choice;

if (cin.good() && choice >= 0 && choice <= 7) {

break;

}

cin.clear();

cout << "Неправильный ввод данных\n";

cin.ignore(100, '\n');

}

switch (choice) {

case 1: {

system("cls");

int choice1;

do {

cout << "\nМеню выбора" << endl;

cout << "1 - Добавить гонку" << endl;

cout << "2 - Добавить шутер" << endl;

cout << "3 - Добавить фэнтези" << endl;

cout << "0 - Выйти" << endl;

while (true) {

cout << "Выберите действие: ";

cin >> choice1;

if (cin.good() && choice1 >= 0 && choice1 <= 3) {

break;

}

cin.clear();

cout << "Неправильный ввод данных\n";

cin.ignore(100, '\n');

}

switch (choice1) {

case 1: {

system("cls");

string title;

cout << "Введите название игры" << endl;

cin >> title;

string company;

cout << "Введите издателя игры" << endl;

cin >> company;

int year;

while (true) {

cout << "Введите год выпуска" << endl;

cin >> year;

if (cin.good() && year >= 1000 && year <= 2023) {

break;

}

cin.clear();

cout << "Неправильный ввод данных\n";

cin.ignore(100, '\n');

}

string raceType;

cout << "Введите тип гонки" << endl;

cin >> raceType;

race = new RaceGame(title, company, year, raceType);

raceGames.insert(race);

current\_race++;

cout << "Игра успешно добавлена" << endl;

break;

}

case 2: {

system("cls");

string title;

cout << "Введите название игры" << endl;

cin >> title;

string company;

cout << "Введите издателя игры" << endl;

cin >> company;

int year;

while (true) {

cout << "Введите год выпуска" << endl;

cin >> year;

if (cin.good() && year >= 1000 && year <= 2023) {

break;

}

cin.clear();

cout << "Неправильный ввод данных\n";

cin.ignore(100, '\n');

}

string shooterType;

cout << "Введите вид шутера" << endl;

cin >> shooterType;

shooter = new ShooterGame(title, company, year, shooterType);

shooterGames.insert(make\_pair(year, shooter));

cout << "Игра успешно добавлена" << endl;

current\_shooter++;

break;

}

case 3: {

system("cls");

if (current\_fantasy < 6)

{

string title;

cout << "Введите название игры" << endl;

cin >> title;

string company;

cout << "Введите издателя игры" << endl;

cin >> company;

int year;

while (true) {

cout << "Введите год выпуска" << endl;

cin >> year;

if (cin.good() && year >= 1000 && year <= 2023) {

break;

}

cin.clear();

cout << "Неправильный ввод данных\n";

cin.ignore(100, '\n');

}

string worldName;

cout << "Введите название мира" << endl;

cin >> worldName;

fantasy = new FantasyGame(title, company, year, worldName);

fantasyGames.insert(fantasy);

cout << "Игра успешно добавлена" << endl;

current\_fantasy++;

}

cout << "\n";

break;

}

}

} while (choice1 != 0);

break;

}

case 2: {

system("cls");

if (!raceGames.empty())

{

cout << "-------------------------------------------------------------------------------------\n";

cout << "| Гонки |\n";

cout << "-------------------------------------------------------------------------------------\n";

cout << '|' << setw(25) << left << "Название игры" << '|' << setw(20) << "Издатель " << '|' << setw(10) << "Год" << '|' << setw(25) << "Тип гонок" << '|' << endl;

cout << "-------------------------------------------------------------------------------------\n";

for (auto& item : raceGames)

{

cout << '|' << setw(25) << left << item->getTitle() << '|' << setw(20)

<< item->getCompany() << '|' << setw(10) << item->getYear() << "|" << setw(25) << item->getRaceType() << "|" << endl;

cout << "-------------------------------------------------------------------------------------\n";

}

}

else

{

cout << "Отсутствуют гонки" << endl;

}

std::cout << "\n";

if (!shooterGames.empty())

{

cout << "-------------------------------------------------------------------------------------\n";

cout << "| Шутеры |\n";

cout << "-------------------------------------------------------------------------------------\n";

cout << '|' << setw(25) << left << "Название игры" << '|' << setw(20) << "Издатель " << '|' << setw(10) << "Год" << '|' << setw(25) << "Тип шутера" << '|' << endl;

cout << "-------------------------------------------------------------------------------------\n";

for (auto& item : shooterGames)

{

cout << '|' << setw(25) << left << item.second->getTitle() << '|' << setw(20) << item.second->getCompany() << '|' << setw(10) << item.second->getYear() << "|" << setw(25) << item.second->getShooterType() << "|" << endl;

cout << "-------------------------------------------------------------------------------------\n";

}

}

else

{

cout << "Отсутсвуют шутеры" << endl;

}

std::cout << "\n";

if (!fantasyGames.empty())

{

cout << "-------------------------------------------------------------------------------------\n";

cout << "| Фэнтези |\n";

cout << "-------------------------------------------------------------------------------------\n";

cout << '|' << setw(25) << left << "Название игры" << '|' << setw(20) << "Издатель " << '|' << setw(10) << "Год" << '|' << setw(25) << " Название мира" << '|' << endl;

cout << "-------------------------------------------------------------------------------------\n";

for (auto& item : fantasyGames)

{

cout << '|' << setw(25) << left << item->getTitle() << '|' << setw(20) << item->getCompany() << '|' << setw(10) << item->getYear() << "|" << setw(25) << item->getName() << "|" << endl;

cout << "-------------------------------------------------------------------------------------\n";

cout << "\n";

}

}

else

{

cout << "Отсутсвует фэнтези" << endl;

}

cout << "\n";

break;

}

case 3: {

system("cls");

int choice1;

do {

cout << "\nМеню выбора" << endl;

cout << "1 - Поиск гонок" << endl;

cout << "2 - Поиск шутера" << endl;

cout << "3 - Поиск фэнтези" << endl;

cout << "0 - Выйти" << endl;

while (true) {

cout << "Выберите действие: ";

cin >> choice1;

if (cin.good() && choice1 >= 0 && choice1 <= 3) {

break;

}

cin.clear();

cout << "Неправильный ввод данных\n";

cin.ignore(100, '\n');

}

switch (choice1) {

case 1: {

system("cls");

if (raceGames.empty()) {

cout << "Нет доступных игр для отображения" << endl;

}

else

{

cout << "-------------------------------------------------------------------------------------\n";

cout << "| Гонки |\n";

cout << "-------------------------------------------------------------------------------------\n";

cout << '|' << setw(25) << left << "Название игры" << '|' << setw(20) << "Издатель " << '|' << setw(10) << "Год" << '|' << setw(25) << "Тип гонок" << '|' << endl;

cout << "-------------------------------------------------------------------------------------\n";

for (auto& item : raceGames)

{

cout << '|' << setw(25) << left << item->getTitle() << '|' << setw(20)

<< item->getCompany() << '|' << setw(10) << item->getYear() << "|" << setw(25) << item->getRaceType() << "|" << endl;

cout << "-------------------------------------------------------------------------------------\n";

}

cout << "\n";

string name\_poi;

cout << "Введите название игры, которую хотите удалить" << endl;

cin >> name\_poi;

int x = 0;

for (auto& item : raceGames)

{

if (item->getTitle() == name\_poi) {

raceGames.erase(item);

cout << "Игра удалена" << endl;

x++;

break;

}

}

if (x == 0)

{

cout << "Такой игры нет" << endl;

}

cout << "\n";

}

break;

}

case 2: {

system("cls");

if (shooterGames.empty()) {

cout << "Нет доступных игр для отображения" << endl;

}

else {

cout << "-------------------------------------------------------------------------------------\n";

cout << "| Шутеры |\n";

cout << "-------------------------------------------------------------------------------------\n";

cout << '|' << setw(25) << left << "Название игры" << '|' << setw(20) << "Издатель " << '|' << setw(10) << "Год" << '|' << setw(25) << "Тип шутера" << '|' << endl;

cout << "-------------------------------------------------------------------------------------\n";

for (auto& entry : shooterGames)

{

cout << '|' << setw(25) << left << entry.second->getTitle() << '|' << setw(20) << entry.second->getCompany() << '|' << setw(10) << entry.first << "|" << setw(25) << entry.second->getShooterType() << "|" << endl;

cout << "-------------------------------------------------------------------------------------\n";

}

string name\_poi;

cout << "Введите название игры, которую вы хотите удалить" << endl;

cin >> name\_poi;

int flag = 0;

int yaer;

for (auto& item : shooterGames)

{

yaer = item.first;

if (item.second->getTitle() == name\_poi)

{

flag = 1;

shooterGames.erase(yaer);

cout << "Игра удалена" << endl;

break;

}

}

if (flag == 0)

{

cout << "Такой игры нет" << endl;

}

cout << "\n";

}

break;

}

case 3: {

system("cls");

if (fantasyGames.empty()) {

cout << "Нет доступных игр для отображения" << endl;

}

else {

cout << "-------------------------------------------------------------------------------------\n";

cout << "| Фэнтези |\n";

cout << "-------------------------------------------------------------------------------------\n";

cout << '|' << setw(25) << left << "Название игры" << '|' << setw(20) << "Издатель " << '|' << setw(10) << "Год" << '|' << setw(25) << " Название мира" << '|' << endl;

cout << "-------------------------------------------------------------------------------------\n";

for (auto& it : fantasyGames)

{

cout << '|' << setw(25) << left << it->getTitle() << '|' << setw(20) << it->getCompany() << '|' << setw(10) << it->getYear() << "|" << setw(25) << it->getName() << "|" << endl;

cout << "-------------------------------------------------------------------------------------\n";

}

cout << "\n";

string name\_poi;

cout << "Введите название игры, который хотите удалить" << endl;

cin >> name\_poi;

int flag = 0;

for (auto& item : fantasyGames)

{

if (item->getTitle() == name\_poi)

{

flag = 1;

fantasyGames.erase(item);

cout << "Игра удалена" << endl;

break;

}

}

if (flag == 0)

{

cout << "Такой игры нет" << endl;

}

cout << "\n";

}

break;

}

}

} while (choice1 != 0);

break;

}

case 5: {

system("cls");

int choice1;

do {

cout << "\nМеню выбора" << endl;

cout << "1 - Изменить название гонки" << endl;

cout << "2 - Изменить название шутера" << endl;

cout << "3 - Изменить название фэнтези" << endl;

cout << "0 - Выйти" << endl;

while (true) {

cout << "Выберите действие: ";

cin >> choice1;

if (cin.good() && choice1 >= 0 && choice1 <= 3) {

break;

}

cin.clear();

cout << "Неправильный ввод данных\n";

cin.ignore(100, '\n');

}

switch (choice1) {

case 1: {

system("cls");

if (raceGames.empty()) {

cout << "Нет доступных игр для отображения" << endl;

}

else

{

cout << "-------------------------------------------------------------------------------------\n";

cout << "| Гонки |\n";

cout << "-------------------------------------------------------------------------------------\n";

cout << '|' << setw(25) << left << "Название игры" << '|' << setw(20) << "Издатель " << '|' << setw(10) << "Год" << '|' << setw(25) << "Тип гонок" << '|' << endl;

cout << "-------------------------------------------------------------------------------------\n";

for (auto& item : raceGames)

{

cout << '|' << setw(25) << left << item->getTitle() << '|' << setw(20)

<< item->getCompany() << '|' << setw(10) << item->getYear() << "|" << setw(25) << item->getRaceType() << "|" << endl;

cout << "-------------------------------------------------------------------------------------\n";

}

cout << "\n";

string name\_poi;

cout << "Введите название, которое хотите изменить:" << endl;

cin >> name\_poi;

int poi = 0;

for (auto& item : raceGames)

{

if (item->getTitle() == name\_poi)

{

poi = 1;

std::string name\_new;

cout << "Введите новое название игры:" << endl;

cin >> name\_new;

item->setTitle(name\_new);

cout << "Название изменено" << endl;

break;

}

}

if (poi == 0)

{

cout << "Такой игры нет" << endl;

}

cout << "\n";

}

break;

}

case 2: {

system("cls");

if (shooterGames.empty()) {

cout << "Нет доступных игр для отображения" << endl;

}

else {

cout << "-------------------------------------------------------------------------------------\n";

cout << "| Шутеры |\n";

cout << "-------------------------------------------------------------------------------------\n";

cout << '|' << setw(25) << left << "Название игры" << '|' << setw(20) << "Издатель " << '|' << setw(10) << "Год" << '|' << setw(25) << "Тип шутера" << '|' << endl;

cout << "-------------------------------------------------------------------------------------\n";

for (auto& entry : shooterGames)

{

cout << '|' << setw(25) << left << entry.second->getTitle() << '|' << setw(20) << entry.second->getCompany() << '|' << setw(10) << entry.first << "|" << setw(25) << entry.second->getShooterType() << "|" << endl;

cout << "-------------------------------------------------------------------------------------\n";

}

cout << "\n";

string name\_poi;

cout << "Введите название игры, которое хотите изменить:" << endl;

cin >> name\_poi;

int poi = 0;

for (auto& item : shooterGames)

{

if (item.second->getTitle() == name\_poi)

{

poi = 1;

string name\_new;

cout << "Введите новое название" << endl;

cin >> name\_new;

item.second->setTitle(name\_new);

cout << "Название изменено" << endl;

break;

}

}

if (poi == 0)

{

cout << "Такой игры нет" << endl;

}

cout << "\n";

}

break;

}

case 3: {

system("cls");

if (fantasyGames.empty()) {

cout << "Нет доступных игр для отображения" << endl;

}

else {

cout << "-------------------------------------------------------------------------------------\n";

cout << "| Фэнтези |\n";

cout << "-------------------------------------------------------------------------------------\n";

cout << '|' << setw(25) << left << "Название игры" << '|' << setw(20) << "Издатель " << '|' << setw(10) << "Год" << '|' << setw(25) << " Название мира" << '|' << endl;

cout << "-------------------------------------------------------------------------------------\n";

for (auto& it : fantasyGames)

{

cout << '|' << setw(25) << left << it->getTitle() << '|' << setw(20) << it->getCompany() << '|' << setw(10) << it->getYear() << "|" << setw(25) << it->getName() << "|" << endl;

cout << "-------------------------------------------------------------------------------------\n";

}

cout << "\n";

string name\_poi;

cout << "Введите название, которое хотите изменить:" << endl;

cin >> name\_poi;

int poi = 0;

for (auto& item : fantasyGames)

{

if (item->getTitle() == name\_poi)

{

string name\_new;

poi = 1;

cout << "Введите новое название:" << endl;

cin >> name\_new;

item->setTitle(name\_new);

cout << "Название изменено" << endl;

break;

}

}

if (poi == 0)

{

cout << "Такой игры нет" << endl;

}

cout << "\n";

}

break;

}

}

} while (choice1 != 0);

break;

}

case 6: {

system("cls");

int choice1;

do {

cout << "\nМеню выбора" << endl;

cout << "1 - Отсортировать гонки по названию" << endl;

cout << "2 - Отсортировать шутеры по году выпуска" << endl;

cout << "3 - Отсортировать фэнтези по названию мира" << endl;

cout << "0 - Выйти" << endl;

while (true) {

cout << "Выберите действие: ";

cin >> choice1;

if (cin.good() && choice1 >= 0 && choice1 <= 3) {

break;

}

cin.clear();

cout << "Неправильный ввод данных\n";

cin.ignore(100, '\n');

}

switch (choice1) {

case 1: {

system("cls");

if (raceGames.empty()) {

cout << "Нет доступных игр" << endl;

}

else

{

std::string\* temporary = new std::string[raceGames.size()];

int i = 0;

for (auto& item : raceGames) {

temporary[i] = item->getTitle();

i++;

}

for (int k = 0; k < raceGames.size(); k++) {

for (int j = raceGames.size() - 1; j > k; j--) {

if (temporary[j - 1] > temporary[j]) {

swap(temporary[j - 1], temporary[j]);

}

}

}

cout << "Отсортированный список:" << endl;

cout << "-------------------------------------------------------------------------------------\n";

cout << '|' << setw(25) << left << "Название игры" << '|' << setw(20) << "Издатель " << '|' << setw(10) << "Год" << '|' << setw(25) << "Тип гонок" << '|' << endl;

cout << "-------------------------------------------------------------------------------------\n";

for (int j = 0; j < raceGames.size(); j++) {

for (auto& item : raceGames)

{

if (temporary[j] == item->getTitle())

{

cout << '|' << setw(25) << left << item->getTitle() << '|' << setw(20)

<< item->getCompany() << '|' << setw(10) << item->getYear() << "|" << setw(25) << item->getRaceType() << "|" << endl;

cout << "-------------------------------------------------------------------------------------\n";

}

}

}

cout << "\n";

}

break;

}

case 2: {

system("cls");

if (shooterGames.empty()) {

cout << "Нет доступных игр" << endl;

}

else {

std::list<int> sort;

for (auto& item : shooterGames)

{

int G;

G = item.first;

sort.push\_back(G);

}

sort.sort();

std::cout << "Отсортированный список:" << std::endl;

cout << "-------------------------------------------------------------------------------------\n";

cout << '|' << setw(25) << left << "Название игры" << '|' << setw(20) << "Издатель " << '|' << setw(10) << "Год" << '|' << setw(25) << "Тип шутера" << '|' << endl;

cout << "-------------------------------------------------------------------------------------\n";

for (std::list<int>::iterator it = sort.begin(); it != sort.end(); it++)

{

for (auto& item : shooterGames)

{

float av;

av = item.first;

if (av == \*it)

{

cout << '|' << setw(25) << left << item.second->getTitle() << '|' << setw(20) << item.second->getCompany() << '|' << setw(10) << item.first << "|" << setw(25) << item.second->getShooterType() << "|" << endl;

cout << "-------------------------------------------------------------------------------------\n";

}

}

}

}

cout << "\n";

break;

}

case 3: {

system("cls");

if (fantasyGames.empty()) {

cout << "Нет доступных игр" << endl;

}

else {

std::list<int> sort;

for (auto& item : fantasyGames)

{

sort.push\_back(item->getYear());

}

sort.sort();

std::cout << "Отсортированный список:" << std::endl;

cout << "-------------------------------------------------------------------------------------\n";

cout << '|' << setw(25) << left << "Название игры" << '|' << setw(20) << "Издатель " << '|' << setw(10) << "Год" << '|' << setw(25) << "Название мира" << '|' << endl;

cout << "-------------------------------------------------------------------------------------\n";

for (std::list<int>::iterator it = sort.begin(); it != sort.end(); it++)

{

for (auto& item : fantasyGames)

{

if (item->getYear() == \*it)

{

cout << '|' << setw(25) << left << item->getTitle() << '|' << setw(20) << item->getCompany() << '|' << setw(10) << item->getYear() << "|" << setw(25) << item->getName() << "|" << endl;

cout << "-------------------------------------------------------------------------------------\n";

}

}

}

}

cout << "\n";

break;

}

}

} while (choice1 != 0);

break;

}

case 7: {

system("cls");

std::ofstream File;

File.open(GAMESFILENAME, std::ios::binary);

if (File.is\_open())

{

File << "-------------------------------------------------------------------------------------\n";

File << "| Гонки |\n";

File << "-------------------------------------------------------------------------------------\n";

File << '|' << setw(25) << left << "Название игры" << '|' << setw(20) << "Издатель " << '|' << setw(10) << "Год" << '|' << setw(25) << "Тип гонок" << '|' << endl;

File << "-------------------------------------------------------------------------------------\n";

for (auto& item : raceGames)

{

File << '|' << setw(25) << left << item->getTitle() << '|' << setw(20)

<< item->getCompany() << '|' << setw(10) << item->getYear() << "|" << setw(25) << item->getRaceType() << "|" << endl;

File << "-------------------------------------------------------------------------------------\n";

}

File << "-------------------------------------------------------------------------------------\n";

File << "| Шутеры |\n";

File << "-------------------------------------------------------------------------------------\n";

File << '|' << setw(25) << left << "Название игры" << '|' << setw(20) << "Издатель " << '|' << setw(10) << "Год" << '|' << setw(25) << "Тип шутера" << '|' << endl;

File << "-------------------------------------------------------------------------------------\n";

for (auto& item : shooterGames)

{

File << '|' << setw(25) << left << item.second->getTitle() << '|' << setw(20) << item.second->getCompany() << '|' << setw(10) << item.first << "|" << setw(25) << item.second->getShooterType() << "|" << endl;

File << "-------------------------------------------------------------------------------------\n";

}

File << "-------------------------------------------------------------------------------------\n";

File << "| Фэнтези |\n";

File << "-------------------------------------------------------------------------------------\n";

File << '|' << setw(25) << left << "Название игры" << '|' << setw(20) << "Издатель " << '|' << setw(10) << "Год" << '|' << setw(25) << " Название мира" << '|' << endl;

File << "-------------------------------------------------------------------------------------\n";

for (auto& item : fantasyGames)

{

File << '|' << setw(25) << left << item->getTitle() << '|' << setw(20) << item->getCompany() << '|' << setw(10) << item->getYear() << "|" << setw(25) << item->getName() << "|" << endl;

File << "-------------------------------------------------------------------------------------\n";

File << "\n";

}

std::cout << "Данные успешно записаны в файл\n";

}

else std::cout << "Ошибка открытия файла\n";

File.close();

break;

}

}

}

catch (invalid\_argument& e) {

cout << e.what() << endl;

}

catch (MyException& e) {

cout << e.what() << endl;

}

catch (MyExceptionComputer& e) {

cout << e.what() << endl;

}

} while (choice != 0);

return 0;

}

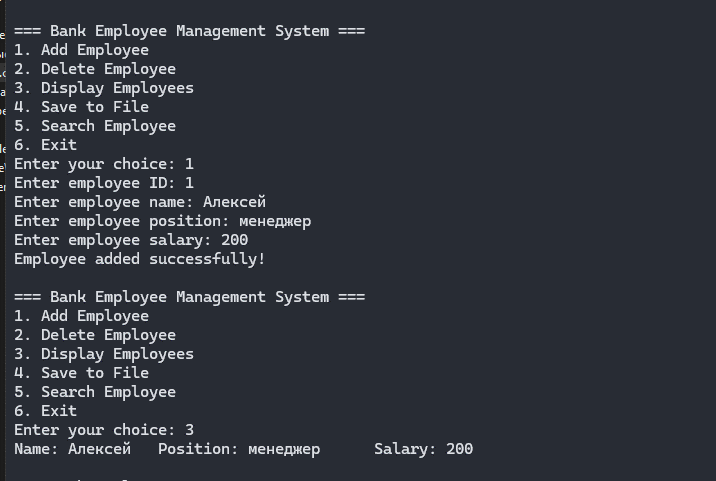


Рисунок 2 – Ввод информации

**Вывод**: в ходе выполнения данной лабораторной работы были изучены и применены основные принципы работы с ассоциативными контейнерами.