МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Московский государственный технологический университет «СТАНКИН»

(ФГБОУ ВО «МГТУ «СТАНКИН»)

Институт информационных технологий

Кафедра информационных систем

КУРСОВАЯ РАБОТА

по дисциплине «Объектно-ориентированное программирование» Тема: «Игра города»

Студент
группа ИДБ–21–06

Музафаров К.Р.

_____
подпись

Руководитель
к.т.н., доцент

РазумовскийА.И.

_____
подпись

Москва

2023 г.

ВВЕДЕНИЕ

Целью данной курсовой работы будет являться написание игры города на языке C++. Игра в города – это популярная игра, которая позволяет развивать кругозор и узнавать новые города. Она может быть интересной и полезной не только для детей, но и для взрослых. Однако, в современном мире существует множество вариаций этой игры, что может привести к путанице и недопониманию при выборе правил и параметров игры.

В данной курсовой работе будет разработана игра в города на языке программирования C++ с применением ООП (объектно-ориентированного программирования), которое позволяет создавать более эффективные и надежные программы.В данной курсовой работе будут освещены вопросы реализации пользовательского интерфейса и логики игры.

Целью данной работы является изучение и применение паттернов ООП при разработке игры в города на языке C++. В результате будет получен полноценный программный продукт, который может быть использован не только в качестве досуга, но и в образовательных целях, например при изучении географии городов России.

Работа состоит из двух частей. В первой части будет рассмотрена теория ООП, во второй – разработка логики игры и реализация пользовательского интерфейса.

ГЛАВА 1. СРЕДСТВА РАЗРАБОТКИ

1.1. ОСНОВНЫЕ ПРИНЦИПЫ ОБЪЕКТНО ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ

В настоящее время объектно-ориентированный подход при разработке систем различной степени сложности общепризнан. Более того, он применяется не только при разработке, но и при использовании широко распространённых объектно-ориентированных систем.

Далее мы перейдем к принципам ООП. Основные принципы объектно-ориентированного программирования (ООП) - это базовые концептуальные структуры, которые используются при проектировании и написании программного обеспечения. Вот более подробное описание каждого из основных принципов ООП:

1. Инкапсуляция - это процесс скрытия внутренней реализации объекта от внешнего мира. Каждый объект в системе должен быть скрыт от других объектов и должен предоставлять только те методы и свойства, которые необходимы для общения с другими объектами. Это обеспечивает защиту объектов от неправильного использования и изменения другими компонентами системы.

2. Полиморфизм - это возможность объекта при определенных условиях обладать разным поведением. При использовании полиморфизма, объект может представлять собой более общий тип, но иметь разные функции в зависимости от конкретного контекста. Полиморфизм делает код более гибким и уменьшает количество необходимого кода.

3. Абстракция - это выделение существенных характеристик объекта и отброс лишних подробностей. Абстракция позволяет создавать упрощенную модель объекта, которая может быть использована для создания еще более сложных систем. При использовании абстракции, мы сосредотачиваемся на сильных сторонах объекта, игнорируя его слабые стороны, что делает моделирование системы более удобным и понятным.

Эти три основных принципа ООП образуют основу для проектирования систем, которые могут быть расширены, поддерживаемы и возможно, более эффективными и надежными по сравнению с системами, созданными без использования ООП.

1.2. ИСПОЛЬЗОВАНИЕ WINDOWS FORMS

Для создания оконного приложения я использовал Windows forms. Windows Forms - новый стиль построения приложения на базе классов .NET Framework class library. Они имеют собственную модель программирования, которая более совершеннее, чем модели, основанные на Win32 API или MFC, и они выполняются в управляемой среде .NET Common Language Runtime (CLR). Эта статья дает представление о том, что такое Windows Forms, рассматривая ее от модели программирования до Microsoft Intermediate Language и JIT-транслятора. Главная выгода от написания Windows-приложений с использованием Windows Forms — это то, что Windows Forms создают более однородную структуру программную модель и устраняют многие ошибки и противоречия от использования Windows API. Например, каждый опытный программист под Windows знает, что некоторые стили окна могут применяться только к окну, когда оно уже создано. Windows Forms в значительной степени устраняют такое противоречие. Если вы хотите существующему окну задать стиль, который может быть присвоен только в момент создания окна, то Windows Forms спокойно уничтожит окно и вновь создаст его с указанным стилем. Кроме того, .NET Framework class library намного более богаче, чем Windows API, и когда вы будете писать приложения, используя Windows Forms, вы получите в распоряжение больше возможностей. Написание приложения с использованием Windows Forms потребует меньшего количества кода, чем приложения, которые используют Windows API или MFC.

1.3. ИСПОЛЬЗОВАНИЕ DLL, ЕГО ПРЕИМУЩЕСТВА И НЕДОСТАТКИ

Библиотека DLL - это библиотека, содержащая код и данные, которые могут использоваться несколькими программами одновременно. Например, в

Windows операционных системах библиотека DLL Comdlg32 выполняет общие функции, связанные с диалогом. Каждая программа может использовать функции, содержащиеся в этой библиотеке DLL, для реализации диалогового окна "Открыть". Это помогает повысить эффективность повторного использования кода и эффективного использования памяти.

С помощью библиотеки DLL программу можно разделить на отдельные компоненты. Например, программа учета может быть продана модулем. Каждый модуль можно загрузить в основную программу во время выполнения, если он установлен. Так как модули являются отдельными, время загрузки программы ускоряется. Модуль загружается только при запросе этой функции.

Кроме того, обновления проще применять к каждому модуле, не затрагивая другие части программы. Например, у вас может быть программа заработной платы, и налоговые ставки меняются каждый год. Если эти изменения изолированы в библиотеке DLL, можно применить обновление, не требуя сборки или установки всей программы еще раз.

В следующем списке описаны некоторые преимущества, которые предоставляются, когда программа использует библиотеку DLL:

- Использует меньше ресурсов

Если несколько программ используют ту же библиотеку функций, библиотека DLL может уменьшить дублирование кода, загруженного на диск и в физической памяти. Это может значительно повлиять на производительность не только программы, выполняемой на переднем плане, но и других программ, работающих в Windows операционной системе.

- Повышение уровня модульной архитектуры

Библиотека DLL помогает повысить уровень разработки модульных программ. Она помогает разрабатывать крупные программы, для которых требуется несколько языковых версий, или программы, для которых требуется модульная архитектура. Примером модульной программы является программа учета с множеством модулей, которые можно динамически загрузить во время выполнения.

• Упрощает развертывание и установку

Если функции в библиотеке DLL требуется обновление или исправление, для развертывания и установки библиотеки DLL не требуется повторное связывание программы с библиотекой DLL. Кроме того, если несколько программ используют ту же библиотеку DLL, все эти программы будут пользоваться преимуществами обновления или исправления. Эта проблема может возникать чаще при использовании библиотеки DLL сторонних разработчиков, которая регулярно обновляется или исправлена.

Единственным крупным недостатком можно считать тот факт, что программа, использующая dll-библиотеку, не может иметь полный функционал в ее отсутствии.

Если приложение использует функцию в библиотеке, или одна библиотека использует функцию из другой библиотеки, то получается
зависимость, из-за которой приложение или библиотека становятся зависимыми, теряют свою самодостаточность. Соответственно при следующих событиях происходят ошибки:

- Соответствующая библиотека обновилась до новой версии;
- Внесены изменения в зависимую dll-библиотеку;
- Соответствующий dll-файл перезаписывался с более ранней версией;
- Соответствующий dll-файл не найден системой или удален с компьютера.

Обычно эти действия называются конфликтами dll-библиотек. Если не обеспечивается обратная совместимость, программа не может быть успешно запущена. Такие действия называют конфликтом dll-библиотек. Программа может быть успешно запушена и нормально использоваться только в случае обеспечения обратной совместимости.

1.4 ОБЫЧНЫЕ И АССОЦИАТИВНЫЕ КОНТЕЙНЕРЫ

C++ предоставляет программисту две основные категории структур данных - это обычные и ассоциативные контейнеры.

Обычные контейнеры и ассоциативные контейнеры в C++ отличаются своей основной функцией. Обычные контейнеры используются для хранения элементов данных в последовательности или коллекции, и имеют доступ к этим элементам через итераторы или индексы. Ассоциативные контейнеры используют пары ключ-значение для хранения элементов данных и имеют доступ к ним через ключ.

Вот некоторые основные отличия между этими двумя типами контейнеров:

1. Доступ к элементам: Обычные контейнеры имеют доступ к элементам по индексу или итератору, тогда как ассоциативные контейнеры имеют доступ к ним только через ключ.

2. Сортировка: Ассоциативные контейнеры сортируют ключи автоматически для обеспечения более быстрого поиска, тогда как обычные контейнеры не сортируют элементы автоматически.

3. Уникальность: В ассоциативных контейнерах ключи должны быть уникальными, тогда как в обычных контейнерах могут быть дубликаты элементов.

4. Скорость доступа: Ассоциативные контейнеры обеспечивают поиск элементов по ключу за время O(log n), тогда как обычные контейнеры обеспечивают доступ к элементам за время O(1).

5. Размер: Обычные контейнеры лучше подходят для хранения большого количества элементов, тогда как ассоциативные контейнеры лучше подходят для небольшого количества элементов.

6. Использование: Обычные контейнеры могут использоваться для различных целей, тогда как ассоциативные контейнеры часто используются для организации данных в структуры типа словарей и таблиц.

Оба типа контейнеров предоставляют разработчикам гибкость и эффективность при работе с элементами данных, но выбор конкретного типа контейнера зависит от задачи, которую нужно решить.

К обычным контейнерам относятся:

- Vector - Контейнер, который хранит элементы в виде динамического массива.

- Deque - Контейнер, который представляет собой двустороннюю очередь (двунаправленный список) элементов.

- List - Контейнер, который хранит элементы в виде двунаправленного списка.

- Array - Контейнер, который хранит фиксированное количество элементов в массиве заданного размера.

- Forward_list - Контейнер, который хранит элементы в виде однонаправленного списка.

К ассоциативным контейнерам относятся:

- Map - Контейнер, который хранит элементы в отсортированном порядке по ключу.
- unordered_map - Контейнер, который хранит элементы в неупорядоченном списке, где доступ к элементам осуществляется по ключу с помощью хэш-функции.
- set - Контейнер, который хранит уникальные ключи в отсортированном порядке.
- unordered_set - Контейнер, который хранит уникальные ключи в неупорядоченном списке с помощью хэш-функции.

1.5. АЛГОРИТМ STD::EQUAL_RANGE

Возвращает диапазон, содержащий все элементы, эквивалентные value в диапазоне [first, last).

Диапазон [first, last) должен быть хотя бы частично упорядочен относительно value , то есть он должен удовлетворять всем следующим требованиям:

• секционируется относительно element < value или comp(element, value) (то есть все элементы, для которых выражение является true предшествуют всем элементам, для которых выражение является false)

- разбит на части по отношению к !(value < element) или !comp(value, element)
- для всех элементов, если element < value или comp(element, value) - true , тогда !(value < element) или !comp(value, element) - также true

Полностью отсортированный диапазон отвечает этим критериям. Возвращаемый диапазон определяется двумя итераторами, один из которых указывает на первый элемент, которым является не менее чем value а другой указывает на первый элемент greater чем value

1.6. АЛГОРИТМ STD::SET_UNION

Алгоритм std::set_union выполняет операцию объединения двух отсортированных последовательностей. Он переводит входные итераторы в начальную и конечную позиции обеих последовательностей, а выходной итератор — в начальную позицию целевого диапазона. Указанный компаратор определяет порядок элементов в целевом диапазоне. Если компаратор не указан, элементы сравниваются по умолчанию. operator<.

ГЛАВА 2.ПРОГРАМНАЯ РЕАЛИЗАЦИЯ.

2.1. ОПИСАНИЕ РАБОТЫ ПРОГРАММЫ

Рассматриваемый в данной курсовой работе программный продукт, является игрой в города. Эта игра поможет как подтянуть знания городов(особенно на сложном уровне), так и просто скрасить время.

Для переключения уровней сложности используется конструкция switch case. Города хранятся в файле txt. В процессе игры, так же города хранятся в обычных и ассоциативных контейнерах. Интерфейс реализован по средствам windows forms.

2.2. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

Код заголовочного файла DLL.h:

```
#ifdef DLL_EXPORTS
```

```
#define DLL_API __declspec(dllexport)
#else
#define DLL_API __declspec(dllimport)
#endif

#include "Words.h"
#include <vector>
#include "DB.h"

extern DLL_API AnswerWord aword;

DLL_API void initWords(int levl);
DLL_API bool isWordUsed(std::string inputWord);
DLL_API void changeAnswerWord(int i);
DLL_API std::vector<Word*>& getDB();
DLL_API std::string isFindWordStartsOnLetter(std::string input);
DLL_API std::vector<std::string>& getDBUser();
DLL_API void addWordUser(std::string word);
DLL_API std::vector<std::string>& getCheckDB();
DLL_API bool isCurrentWord(std::string inputWord);
DLL_API void countWords(char symbol);
```

Код DLL.cpp:

```
// DLL.cpp : Определяет экспортируемые функции для DLL.
//
#include "DLL.h"

struct Leaks { ~Leaks() { _CrtDumpMemoryLeaks(); } };
Leaks _l;

DLL_API DB db;
```

```
DLL_API AnswerWord aword;

DLL_API void initWords(int levl)
{
   std::ifstream WordsIN(L"Towns.txt");
   std::string word;
   int skip = 0;

   switch (levl)
   {
   case 1: {
      // easy
      while (getline(WordsIN, word)) {
         std::locale loc("rus_rus.1251");
                              std::transform(word.begin(),  word.end(),  word.begin(),
uppercaseCustom(loc));
         db.addCheckDB(word);
         if (skip == 3) {
            db.addWordsDB(new Word(word));
            skip = 0;
         }
         skip++;
      }

      break;
   }

   case 2: {
      // meduim

      while (getline(WordsIN, word)) {
```

```
            std::locale loc("rus_rus.1251");
                                    std::transform(word.begin(),  word.end(),  word.begin(),
uppercaseCustom(loc));
            db.addCheckDB(word);
            if (skip == 2) {
               db.addWordsDB(new Word(word));
               skip = 0;
            }
            skip++;
         }

         break;
      }
      default: {
         // hard

         while (getline(WordsIN, word)) {
            std::locale loc("rus_rus.1251");
                                    std::transform(word.begin(),  word.end(),  word.begin(),
uppercaseCustom(loc));
            db.addCheckDB(word);
            db.addWordsDB(new Word(word));
         }

         break;
      }
      }

   db.shuffle();
}
```

```
DLL_API void changeAnswerWord(int i)
{
   aword = AnswerWord(getDB()[i]->getWord());
}

DLL_API bool isWordUsed(std::string inputWord)
{
   std::locale loc("rus_rus.1251");
            std::transform(inputWord.begin(),  inputWord.end(),  inputWord.begin(),
uppercaseCustom(loc));

        if (std::find(getDBUser().begin(), getDBUser().end(), inputWord) == get-
DBUser().end()) {
       addWordUser(inputWord);
       return true;
   }
   return false;
}

DLL_API bool isCurrentWord(std::string inputWord)
{
   std::locale loc("rus_rus.1251");
            std::transform(inputWord.begin(),  inputWord.end(),  inputWord.begin(),
uppercaseCustom(loc));

    return std::find(getCheckDB().begin(), getCheckDB().end(), inputWord) != get-
CheckDB().end();
}

DLL_API void countWords(std::string symbol)
{
```

```
    std::vector<std::string> wordsVector;
    std::ifstream WordsIN(L"Towns.txt");
    std::string word;
    while (getline(WordsIN, word)) {
        std::locale loc("rus_rus.1251");
                                std::transform(word.begin(),   word.end(),   word.begin(),
uppercaseCustom(loc));

    }
    std::pair<std::vector<std::string>::iterator, std::vector<std::string>::iterator> it;
     std::sort(wordsVector.begin(), wordsVector.end(),[](const auto& lh, const auto&
rh)
        {
            return std::stod(lh) < std::stod(rh);
        });

     it = equal_range(wordsVector.begin(), wordsVector.end(), symbol, []( std::string
a, std::string  b) { return (a > b); });
}

DLL_API std::vector<Word*>& getDB()
{
    return db.getDB();
}

DLL_API std::string isFindWordStartsOnLetter(std::string input)
{
    return db.isFindWordStartsOnLetter(input);
}

DLL_API std::vector<std::string>& getDBUser() {
```

```
    return db.getDBUser();
}

DLL_API void addWordUser(std::string word) {
    db.addWordUser(word);
}

DLL_API std::vector<std::string>& getCheckDB() {
    return db.getCheckDB();
}
```

Код заголовочного файла DB.h:

```
#pragma once
#include "framework.h"
#include "Words.h"
#include <fstream>
#include <vector>
#include <set>
#include <algorithm>
#include <random>


class DB
{
private:
        std::vector<Word*> WordsDB;
        std::vector<std::string> CheckDB;
        std::vector<std::string> DBUser;
        std::vector<Word*> sity_temp;
        std::vector<Word*> sity_res;
        std::set<Word*> sity;
```

```
public:
	DB();
	std::vector<Word*>& getDB();
	std::vector<std::string>& getDBUser();
	std::vector<std::string>& getCheckDB();
	std::string isFindWordStartsOnLetter(std::string input);
	void addWordUser(std::string word);
	void addCheckDB(std::string word);
	void addWordsDB(Word* word);
	void determenate();
	void shuffle();
	DB& operator = (const DB& new_obj);
	~DB();
};
```

Код DB.cpp:

```
// DLL.cpp : Определяет экспортируемые функции для DLL.
//
#include "DLL.h"

struct Leaks { ~Leaks() { _CrtDumpMemoryLeaks(); } };
Leaks _l;

DLL_API DB db;
DLL_API AnswerWord aword;

DLL_API void initWords(int levl)
{
   std::ifstream WordsIN(L"Towns.txt");
```

```
    std::string word;
    int skip = 0;

    switch (levl)
    {
    case 1: {
        // easy
        while (getline(WordsIN, word)) {
            std::locale loc("rus_rus.1251");
                                std::transform(word.begin(),  word.end(),  word.begin(),
uppercaseCustom(loc));
            db.addCheckDB(word);
            if (skip == 3) {
                db.addWordsDB(new Word(word));
                skip = 0;
            }
            skip++;
        }

        break;
    }

    case 2: {
        // meduim

        while (getline(WordsIN, word)) {
            std::locale loc("rus_rus.1251");
                                std::transform(word.begin(),  word.end(),  word.begin(),
uppercaseCustom(loc));
            db.addCheckDB(word);
            if (skip == 2) {
```

```
            db.addWordsDB(new Word(word));
            skip = 0;
        }
        skip++;
    }

    break;
  }
  default: {
    // hard

    while (getline(WordsIN, word)) {
        std::locale loc("rus_rus.1251");
                        std::transform(word.begin(),  word.end(),  word.begin(),
uppercaseCustom(loc));
        db.addCheckDB(word);
        db.addWordsDB(new Word(word));
    }

    break;
  }
  }

  db.shuffle();
}

DLL_API void changeAnswerWord(int i)
{
  aword = AnswerWord(getDB()[i]->getWord());
}
```

```
DLL_API bool isWordUsed(std::string inputWord)
{
   std::locale loc("rus_rus.1251");
            std::transform(inputWord.begin(),  inputWord.end(),  inputWord.begin(),
uppercaseCustom(loc));

        if (std::find(getDBUser().begin(), getDBUser().end(), inputWord) == get-
DBUser().end()) {
       addWordUser(inputWord);
       return true;
   }
   return false;
}

DLL_API bool isCurrentWord(std::string inputWord)
{
   std::locale loc("rus_rus.1251");
            std::transform(inputWord.begin(),  inputWord.end(),  inputWord.begin(),
uppercaseCustom(loc));

    return std::find(getCheckDB().begin(), getCheckDB().end(), inputWord) != get-
CheckDB().end();
}

DLL_API void countWords(std::string symbol)
{
   std::vector<std::string> wordsVector;
   std::ifstream WordsIN(L"Towns.txt");
   std::string word;
   while (getline(WordsIN, word)) {
      std::locale loc("rus_rus.1251");
```

```
                    std::transform(word.begin(),  word.end(),  word.begin(),
uppercaseCustom(loc));

  }
  std::pair<std::vector<std::string>::iterator, std::vector<std::string>::iterator> it;
   std::sort(wordsVector.begin(), wordsVector.end(),[](const auto& lh, const auto&
rh)
      {
         return std::stod(lh) < std::stod(rh);
      });

   it = equal_range(wordsVector.begin(), wordsVector.end(), symbol, []( std::string
a, std::string  b) { return (a > b); });
}

DLL_API std::vector<Word*>& getDB()
{
  return db.getDB();
}

DLL_API std::string isFindWordStartsOnLetter(std::string input)
{
  return db.isFindWordStartsOnLetter(input);
}

DLL_API std::vector<std::string>& getDBUser() {
  return db.getDBUser();
}

DLL_API void addWordUser(std::string word) {
  db.addWordUser(word);
```

```
}

DLL_API std::vector<std::string>& getCheckDB() {
   return db.getCheckDB();
}
```

Код полиморфной иерархии Words.h:

```
#pragma once
#include <string>
#include <algorithm>
#include <locale>

struct uppercaseCustom
{
	uppercaseCustom(const std::locale& loc) :
		loc(loc)
	{}

	char operator()(const char& c)
	{
		return std::toupper(c, loc);
	}

private:
	std::locale loc;
};

class Word {
protected:
	std::string word;

public:
```

```
	Word(){}
	Word(std::string word) {
		std::locale loc("rus_rus.1251");
		std::transform(word.begin(), word.end(), word.begin(), uppercase-
Custom(loc));
		this->word = word;
	}

	virtual Word* copyPointer() {
		return new Word(this->word);
	}

	virtual std::string copy() {
		return this->word;
	}

	virtual const std::string& getWord() const {
		return word;
	}

	virtual ~Word()
	{
		word.~basic_string();
	}
};

class AnswerWord : public Word {
public:
	AnswerWord(){}
	AnswerWord(std::string word) : Word(word) {}
```

```
    const std::string& getWord() const override {
        return word;
    }

    bool isValid(std::string inputWord) {
        std::string reverseWord = word;
        reverse(reverseWord.begin(), reverseWord.end());

        std::locale loc("rus_rus.1251");
        std::transform(inputWord.begin(), inputWord.end(),
inputWord.begin(), uppercaseCustom(loc));

        if (reverseWord[0] == inputWord[0]) {
            return true;
        }

        return false;
    }

    ~AnswerWord() {}
};
```

2.3. СКРИНШОТЫ РАБОТЫ ПРОГРАММЫ

При открытии нас встречает меню выбора сложности, а так же кнопка играть (рис 1 - 3).

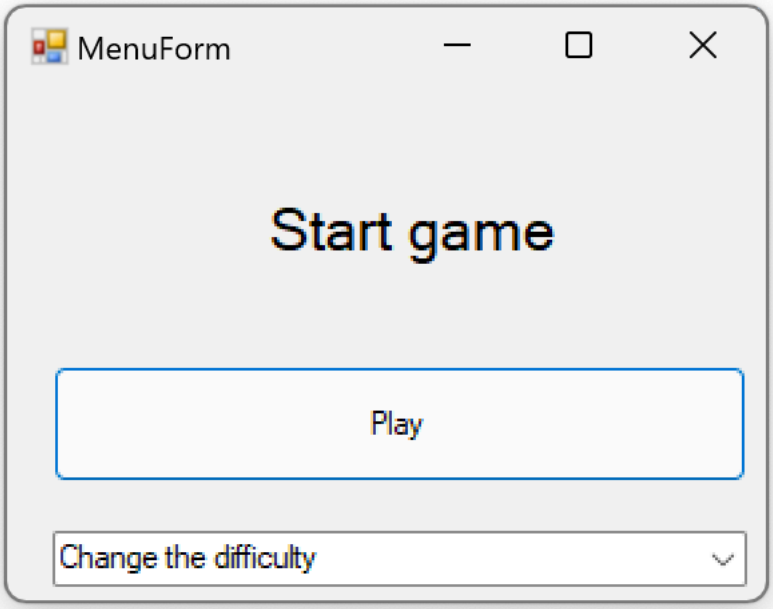

Рис.1. Стартовое меню.

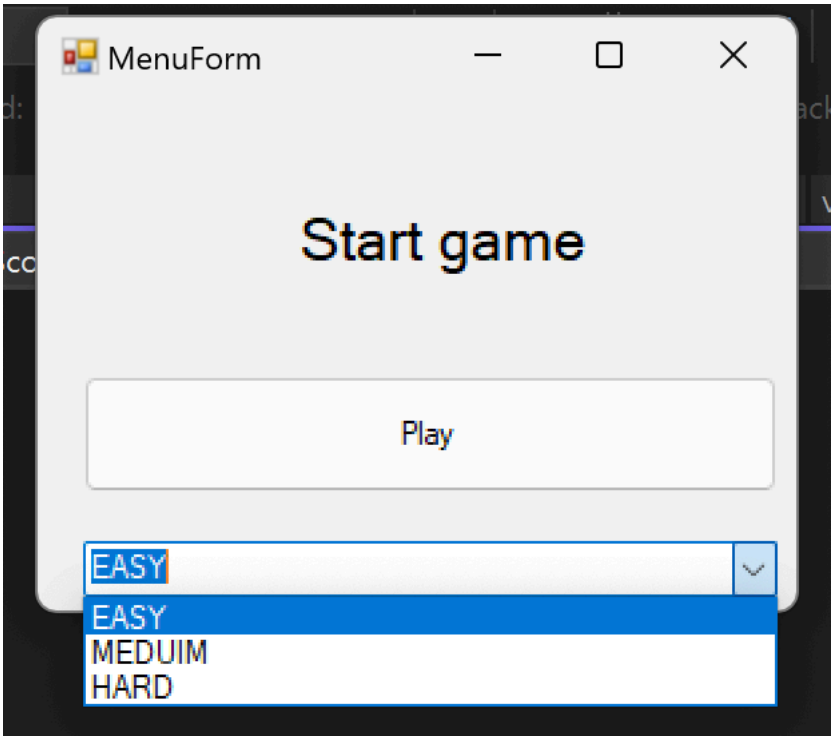

Рис.2. Выбор уровня сложности.

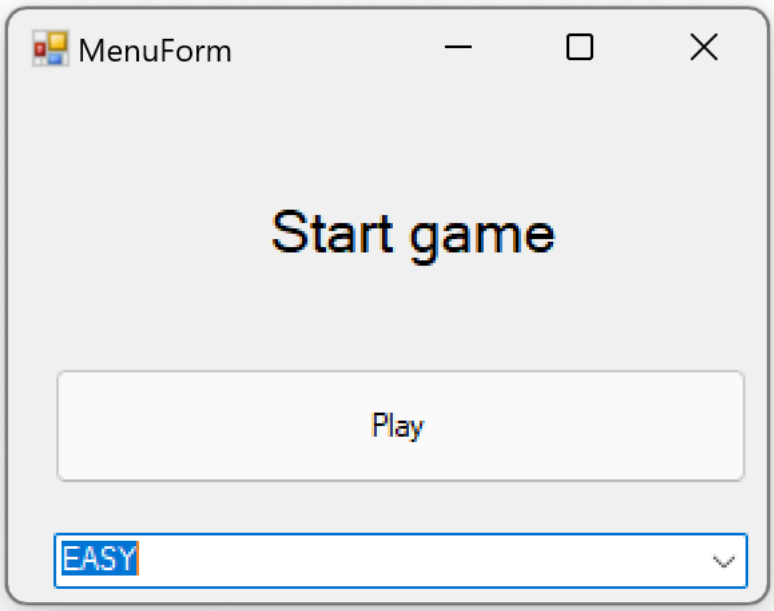

Рис.3. Стартовое меню с выбранной сложностью.

В процессе игры мы будем наблюдать 2 поля, одно для компьютера(туда будут выбираться города из списка всех городов России, представленном ниже, и вставляться специальным алгоритмом), другое для того чтобы ответил игрок. Так де там есть каунтер оставшихся городов и кнопка «Next», для того чтобы перейти к следующему после ответа.

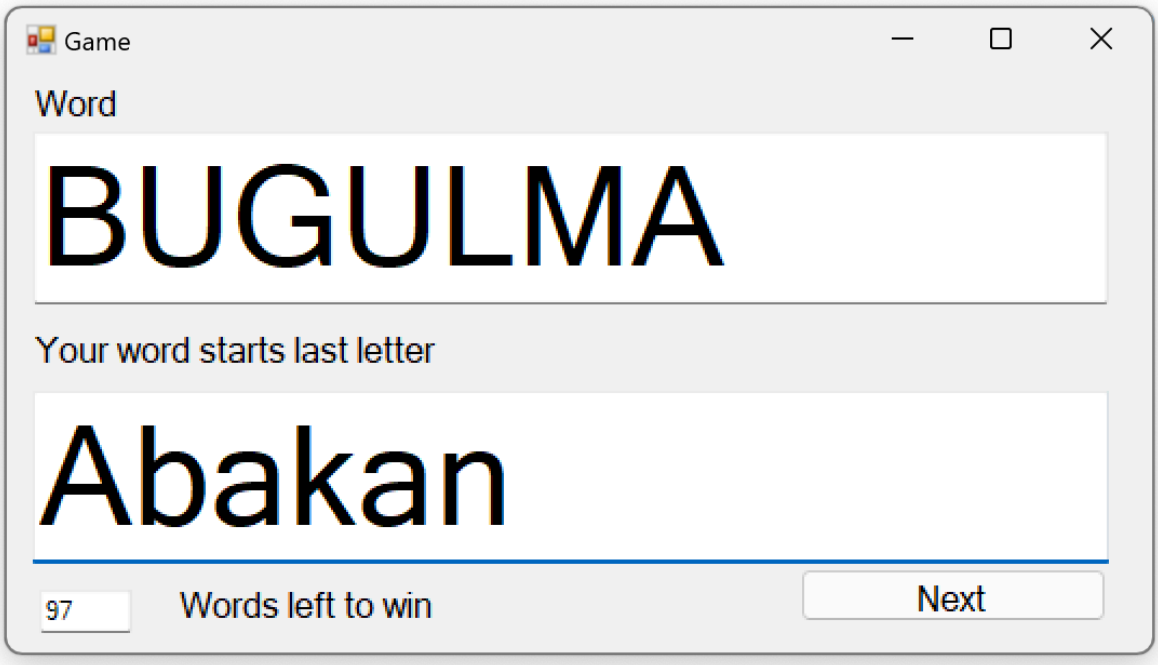

Рис.4. Геймплей.

Список городов:

Abakan, Achinsk, Aginskoye, Akademgorodok, Alapayevsk, Aleksandrovsk-Sakhalinsky, Aleksin, Almetyevsk, Amursk ,Anadyr, Angarsk, Anzhero-Sudzhensk, Arkhangelsk, Armavir, Artyom, Arzamas, Asbest, Asino, Astrakh, Azov, Balakovoa, Balashikha, Baltiysk, Barnaul, Bataysk ,Belgorod, Belogorsk, Belovo, Berdsk, Berezniki, Birobidzhan, Biysk, Blagoveshchensk, Bor, Borisoglebsk, Borovichi, Bratsk, Bryansk, Bugulma, Buguruslan, Buzuluk, Chapayevsk, Cheboksary, Chelyabinsk, Cheremkhovo, Cherepovets, Cherkessk, Chernogorsk, Chistopol, Chita, Chusovoy, Derbent, Dimitrovgrad, Dolgoprudny, Dubna, Dudinka, Dzerzhinsk, Elektrostal, Elista, Engels, Gatchina, Glazov, Gorno, Altaysk, Grozny, Gubkin, Gukovo, Gus-Khrustalny, Igarka, Irkutsk, Ishimbay, Ivanovo, Izhevsk, Kaliningrad, Kaluga, Kamen-na-Obi, Kamensk-Shakhtinsky, Kamensk-Uralsky, Kamyshin, Kansk, Karabulak, Karachayevsk, Kazan, Kemerovo, Khabarovsk, Khanty-Mansiysk, Khasavyurt, Khimki, Kholmogory, Kimry, Kineshma, Kirov, Kirovsk, Kiselyovsk6 Kislovodsk, Kizel, Klin, Kolomenskoye, Kolomna, Komsomolsk-na-Amure, Kopeysk, Korkino, Korolyov, Korsakov, Kostroma, Kotlas, Kovrov, Krasnodar, Krasnogorsk, Krasnokamsk, Krasnoturinsk, Krasnoyarsk, Krasnoye Selo, Kronshtadt, Kropotkin, Kudymkar, Kulebaki, Kungur, Kurgan, Kursk, Kushva, Kyakhta, Kyzyl, Labinsk,Leninsk-Kuznetsky, Lesosibirsk, Lipetsk, Liski, Lomonosov, Lysva, Lyubertsy, Magadan, Magas, Magnitogorsk, Makhachkala, Malgobek, Maykop, Mezhdurechensk ,Miass ,Michurinsk Mikhaylovka ,Mineralnye Vody, Monchegorsk, Moscow, Murmansk, Murom, Mytishchi,Naberezhnye Chelny, Nakhodka, Nalchik, Naro-Fominsk, Naryan-Mar, Naryan-Mar, Nazran, Nerchinsk, Nevinnomyssk, Nikolayevsk-na-Amure, Nizhnekamsk, Nizhnevartovsk, Nizhny, Novgorod, Nizhny Tagil, Noginsk, Norilsk, Novocherkassk, Novokuybyshevsk, Novokuznetsk, Novomoskovsk, Novorossiysk, Novoshakhtinsk, Novosibirsk, Novotroitsk, Oktyabrsky, Omsk, Orekhovo-Zuyevo, Orenburg, Orsk, Oryol, Osa, Osinniki, Palana, Partizansk, Pavlovo, Pavlovsk, Pavlovsky Posad, Pechenga, Penza, Perm, Pervouralsk, Peterhof, Petropavlovsk-Kamchatsky, Petrozavodsk, Podolsk, Polevskoy,

Prokopyevsk, Pskov, Pushkin, Pyatigorsk, Ramenskoye, Revda, Rostov, Rostov-na-Donu, Rubtsovsk, Ryazan, Rybinsk, Rzhev, Saint Petersburg, Salavat, Salekhard, Samara, Saransk, Sarapul, Saratov, Sergiyev Posad, Serov, Serpukhov, Severodvinsk, Shadrinsk, Shakhty, Shchyokino, Shchyolkovo, Shlisselburg, Shuya, Slavyansk-na-Kubani, Smolensk, Sochi, Sokol, Solikamsk, Sovetsk, Sovetskaya Gavan, Staraya Russa, Stary Oskol, Stavropol, Sterlitamak, Stupino, Surgut, Svobodny, Syktyvkar, Syzran,Taganrog,Tambov,Tavda, Teberda, Tikhoretsk, Tobolsk,Tolyatti, Tomsk, Torzhok, Troitsk, Tuapse, Tula, Tulun, Tura, Tver, Tyumen, Ufa, Ukhta, Ulan-Ude, Ulyanovsk, Usolye-Sibirskoye, Ussuriysk, Ust-Dzheguta, Ust-Ilimsk, Ust-Ordynsky, Velikiye Luki, Veliky Novgorod, Veliky Ustyug, Verkhoyansk, Vichuga, Vladikavkaz, Vladimir, Vladivostok, Volgograd, Vologda, Volsk, Volzhsky, Vorkuta, Voronezh, Voskresensk, Votkinsk, Vyborg, Yakutsk, Yaroslavl, Yasnaya Polyana, Yegoryevsk, Yekaterinburg, Yelets
Yessentuki, Yeysk, Yoshkar-Ola, Yuzhno-Sakhalinsk, Zelenodolsk, Zheleznogorsk, Zlatoust, Zvenigorod

ЗАКЛЮЧЕНИЕ

В результате проведенной работы была разработана игра "Города", основанная на принципах объектно-ориентированного программирования. При разработки данного программного продукта были использованы принципы ООП и технологии C++/CLI .NET framework. Несмотря на свои скромные размеры, благодаря применению объектно-ориентированного подхода к проектированию, функционал данной программы всегда можно расширить, прибегая к минимальным изменениям существующего кода. Использование DLL библиотеки позволяет более гибко разрабатывать и расширять интерфейс программы.

Таким образом мы на практике научились проектировать и реализовывать программные продукты с использованием объектно-ориентированного подхода, изучили особенности работы с DLL библиотеками и разработали современный интерфейс на платформе .NET.

Программа работает с использованием интерфейса, написанного на windows forms, что позволяет игрокам легко освоить ее правила и начать играть сразу же. Было учтено большое количество возможных вариантов ввода пользователей, что предотвращает ошибки и помогает сохранять высокую степень удовлетворенности игроков.

Также были реализованы необходимые методы для проверки корректности введенных слов, что позволяет игрокам сосредоточиться на игре и насладиться ее процессом без отвлекающих факторов. В целом, разработанная игра "Город" демонстрирует принципы объектно-ориентированного программирования и представляет собой интересное приложение для развития интеллектуальных способностей игроков.

СПИСОК ЛИТЕРАТУРЫ

1. Официальный сайт Microsoft [Электронный ресурс] Режим доступа: http://docs.microsoft.com (дата обращения: 02.03.2023)

2. STL: стандартная библиотека шаблонов C++ [Электронный ресурс/статья] Режим доступа: https://tproger.ru/articles/stl-cpp (дата обращения: 27.04.2023)

3. Романов С.С. Ключевые понятия и особенности объектно- ориентированного программирования // Таврический научный обозреватель. 2016. No12-2 (17). URL: https://cyberleninka.ru/article/n/klyuchevye-ponyatiya-i-osobennosti-obektno-orientirovannogo-programmirovaniya (дата обращения: 23.03.2023)

4. Большая российская энциклопедия [Электронный ресурс] Режим доступа: https://bigenc.ru/technology_and_technique/text/3958439

5. Официальный сайт Cppreference [Электронный ресурс] Режим доступа: https://ru.cppreference.com (дата обращения: 27.03.2023)

6. Страуструп, Б. Язык программирования C++ / Б. Страуструп. - М.: Радио и связь, 2017. – 1136 с. [Бумажный ресурс]

7. Лафоре Р. Объектно-ориентированное программирование в C++ [Текст] / Р. Лафоре. — 4-е изд. — СПб.: Питер, 2004. — 928 с.