

Лабораторная работа 7 (4 часа)
Языки программирования

**Структурная обработка ошибок, препроцессор,
модель памяти**

1. Используйте материал лекции № 7.
2. Создайте проект-приложение с именем **LP_Lab07**.
3. Разработайте набор функций (API – application program interface), обеспечивающих работу словаря. Весь код должен располагаться в файлах **Dictionary.h** и **Dictionary.cpp**.
4. Перечень функций:

Create	Создать экземпляр словаря. Словарь: массив элементов (структур) Entry . Элементы имеют уникальный идентификатор, который не может дублироваться в словаре. Максимальная емкость словаря (максимальное количество элементов): DICTMAXSIZE . Параметры: имя экземпляра словаря, емкость экземпляра словаря. Возврат: экземпляр словаря (Instance). Исключения: превышена длина имени словаря, превышен размер максимальной емкости словаря.
AddEntry	Добавить элемент словаря. Параметры: экземпляр словаря, элемент словаря (Entry). Исключения: переполнение словаря, дублирование идентификатора.
DelEntry	Удалить элемент словаря. Параметры: экземпляр словаря, идентификатор элемента. Исключения: не найден элемент.
GetEntry	Получить элемент словаря. Параметры: экземпляр словаря, идентификатор элемента. Возврат: элемент словаря (Entry). Исключения: не найден элемент.
UpdEntry	Изменить элемент словаря. Параметры: экземпляр словаря, идентификатор элемента, элемент словаря (Entry). Исключения: не найден элемент, дублирование идентификатора.
Delete	Удалить элементы словаря. Параметры: экземпляр словаря.
Print	Распечатать элементы словаря. Параметры: экземпляр словаря.

5. Все функции должны располагаться в пространстве имен Dictionary.

6. Содержимое файла Dictionary.h представлено на следующем рисунке. Исследуйте его и примените.

```
#include <cstring>
#define DICTNAMEMAXSIZE 20 // маскиальный размер имени словаря
#define DICTMAXSIZE 100 // маскиальная емкость словаря
#define ENTRYNAMEMAXSIZE 30 // маскиальная длина имени в словаре
#define THROW01 "Create: превышен размер имени словаря"
#define THROW02 "Create: превышен размер максимальной емкости словаря"
#define THROW03 "AddEntry: переполнение словаря"
#define THROW04 "AddEntry: дублирование идентификатора"
#define THROW05 "GetEntry: не найден элемент"
#define THROW06 "DelEntry: не найден элемент"
#define THROW07 "UpdEntry: не найден элемент"
#define THROW08 "UpdEntry: дублирование идентификатора"

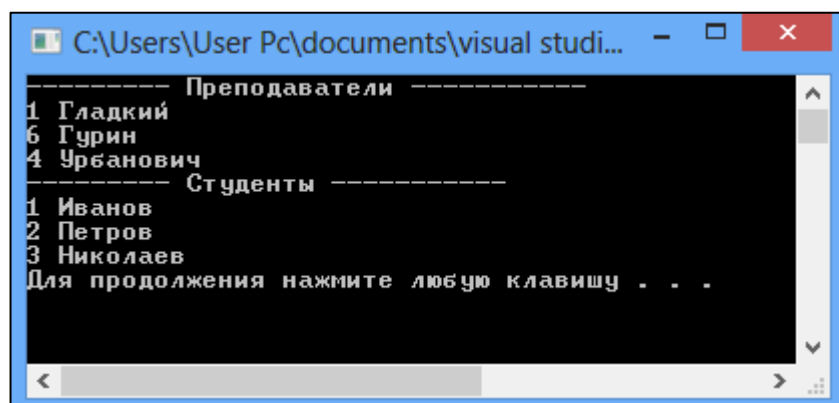
namespace Dictionary //
{
    struct Entry // элемент словаря
    {
        int id; // идентификатор (уникальный)
        char name[ENTRYNAMEMAXSIZE]; // символьная информация
    };
    struct Instance // экземпляр словаря
    {
        char name[DICTNAMEMAXSIZE]; // наименование словаря
        int maxsize; // максимальная емкость словаря
        int size; // текущий размер словаря < DICTNAMEMAXSIZE
        Entry* dictionary; // массив элементов словаря
    };
    Instance Create( // создать словарь
        char name[DICTNAMEMAXSIZE], // имя словаря
        int size // емкость словаря < DICTNAMEMAXSIZE
    );
    void AddEntry( // добавить элемент словаря
        Instance& inst, // экземпляр словаря
        Entry ed // элемент словаря
    );
    void DelEntry( // удалить элемент словаря
        Instance& inst, // экземпляр словаря
        int id // идентификатор удаляемого элемента (уникальный)
    );
    void UpdEntry( // изменить элемент словаря
        Instance& inst, // экземпляр словаря
        int id, // идентификатор заменяемого элемента
        Entry new_ed // новый элемент словаря
    );
    Entry GetEntry( // получить элемент словаря
        Instance inst, // экземпляр словаря
        int id // идентификатор получаемого элемента
    );
    void Print(Instance d); // печать словаря
    void Delete(Instance& d); // удалить словарь
};
```

7. Пример применения функций. Исследуйте его.

```
#include "stdafx.h"
#include <iostream>
#include <locale>
#include "Dictionary.h"

int _tmain(int argc, _TCHAR* argv[])
{
    setlocale(LC_ALL, "rus");
    try
    {
        Dictionary::Instance d1 = Dictionary::Create("Преподаватели", 5); // создание словаря
        Dictionary::Entry e1 = {1, "Гладкий"}, e2 = {2, "Велякин"}, // элементы словаря
            e3 = {3, "Смелов"}, e4 = {4, "Урбанович"}, e5 = {5, "Пацей"};
        Dictionary::AddEntry(d1, e1); //добавление элемента словарь
        Dictionary::AddEntry(d1, e2); //добавление элемента словарь
        Dictionary::AddEntry(d1, e3); //добавление элемента словарь
        Dictionary::AddEntry(d1, e4); //добавление элемента словарь
        Dictionary::Entry ex2 = Dictionary::GetEntry(d1, 4); // найти элемент в словаре по идентификатору
        Dictionary::DelEntry(d1, 2); // удалить элемент из словаря по идентификатору
        Dictionary::Entry newentry1 = {6, "Гурин"}; // элемент словаря
        Dictionary::UpdEntry(d1, 3, newentry1); // заменить элемент словаря по идентификатору
        Dictionary::Print(d1); // распечатать элементы словаря
        Dictionary::Instance d2 = Dictionary::Create("Студенты", 5);
        Dictionary::Entry s1 = {1, "Иванов"}, s2 = {2, "Петров"}, s3 = {3, "Сидоров"};
        Dictionary::AddEntry(d2, s1);
        Dictionary::AddEntry(d2, s2);
        Dictionary::AddEntry(d2, s3);
        Dictionary::Entry newentry3 = {3, "Николаев"};
        Dictionary::UpdEntry(d2, 3, newentry3);
        Dictionary::Print(d2);
        Delete(d1);
        Delete(d2);
    }
    catch (char* e) // обработка исключений словаря
    {
        std::cout<<e<<< std::endl;
    };
    system("pause");
    return 0;
}
```

8. Пример выполнения функции **Print**, распечатывающей элементы двух словарей с наименованиями: **Преподаватели** и **Студенты**. Разработанная в лабораторной работе функция, должна осуществлять вывод в таком же формате.



9. Разработайте контрольный пример, демонстрирующий работу словаря. Компиляция контрольного примера должна управляться с помощью макросов (условная компиляция), перечисленных в представленной ниже таблице.

TEST_CREATE_01	Тест функции Create : проверка генерации исключения THROW01 (см. рисунок).
TEST_CREATE_02	Тест функции Create : проверка генерации исключения THROW02 (см. рисунок).
TEST_ADDENTRY_03	Тест функции AddEntry : проверка генерации исключения THROW03 (см. рисунок).
TEST_ADDENTRY_04	Тест функции AddEntry : проверка генерации исключения THROW04 (см. рисунок).
TEST_GETENTRY_05	Тест функции GetEntry : проверка генерации исключения THROW05 (см. рисунок).
TEST_GETENTRY_06	Тест функции DelEntry : проверка генерации исключения THROW06 (см. рисунок).
TEST_UPDENTRY_07	Тест функции UpdEntry : проверка генерации исключения THROW07 (см. рисунок).
TEST_UPDENTRY_08	Тест функции UpdEntry : проверка генерации исключения THROW08 (см. рисунок).
TEST_DICTIONARY	Демонстрирует успешное выполнение всех функций. Должны быть созданы два словаря: Студенты и Преподаватели. Каждый словарь должен содержать не менее 7 элементов. Словари должны быть распечатаны с помощью функций Print .

10. Макросы, управляющие условной компиляцией должны быть определены в файле **stdafx.h**.
11. В файле **LPLab07.cpp** напишите препроцессорное выражение, запрещающее установку более одного макроса из представленной выше списка (примените директиву **#error**).
12. Продемонстрируйте возможность установки макроса **TEST_DICTIONARY** через свойства проекта.

Ответьте на следующие вопросы:

- что такое препроцессор?
- перечислите все директивы препроцессора;
- какие типы памяти используются приложением C++?
- что такое пространство имен?
- что такое исключение?
- поясните принцип связи инструкций `throw` и `catch` при структурной обработке исключения;
- поясните смысл выражения «необработанное в функции исключение распространяется по стеку вызова функций»;
- укажите в программе место, где используется статическая память (если есть);
- укажите в программе место, где используется память из кучи (если есть);
- укажите в программе место, где используется память из стека (если есть);
- укажите в тексте программы применяемые директивы препроцессора, поясните принцип их действия;
- укажите в тексте программы место, где применяется условная компиляция;
- укажите в тексте программы место, где применяется пространство имен;
- укажите в тексте программы место, где применяется структурная обработка исключений, поясните принцип действия инструкций **`try`**, **`catch`** и **`throw`**.