

**Лабораторная работа 5 (4 часа)**  
**Конструирование программного обеспечения**

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

**Цель работы:** практическое использование директив препроцессора, структурная обработка ошибок.

**Задание.**

1. Используйте материал лекции № 4 и материалы лекции дисциплины ОПИ «Дополнительно ОПИ Л\_08. Классы\_памяти».
2. Создайте проект-приложение с именем **SE\_Lab05**.
3. Разработайте набор функций API (API – application program interface), обеспечивающих работу словаря. Весь код должен располагаться в файлах **Dictionary.h** и **Dictionary.cpp**.
4. Перечень и описание функций:

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

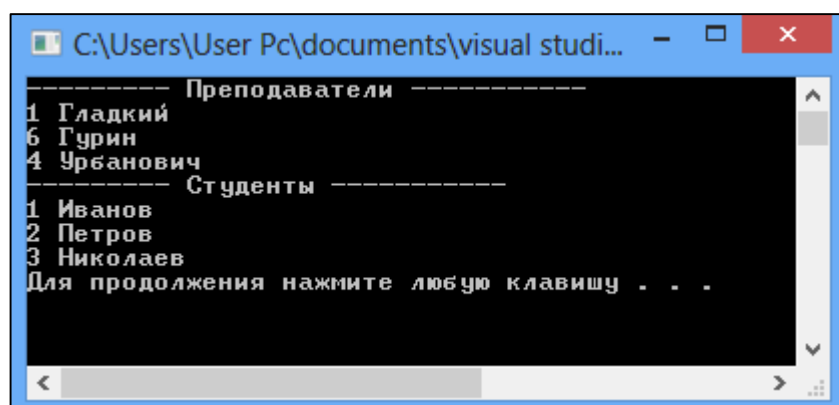
namespace Dictionary //
{
    struct Entry // элемент словаря
    {
        int id; // идентификатор (уникальный)
        char name[ENTRYNAMEMAXSIZE]; // символьная информация
    };
    struct Instance // экземпляр словаря
    {
        char name[DICTNAMEMAXSIZE]; // наименование словаря
        int 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.

Название	Назначение
<b>TEST_CREATE_01</b>	Тест функции <b>Create</b> : проверка генерации исключения <b>THROW01</b> (см. рисунок).
<b>TEST_CREATE_02</b>	Тест функции <b>Create</b> : проверка генерации исключения <b>THROW02</b> (см. рисунок).
<b>TEST_ADDENTRY_03</b>	Тест функции <b>AddEntry</b> : проверка генерации исключения <b>THROW03</b> (см. рисунок).
<b>TEST_ADDENTRY_04</b>	Тест функции <b>AddEntry</b> : проверка генерации исключения <b>THROW04</b> (см. рисунок).
<b>TEST_GETENTRY_05</b>	Тест функции <b>GetEntry</b> : проверка генерации исключения <b>THROW05</b> (см. рисунок).
<b>TEST_GETENTRY_06</b>	Тест функции <b>DelEntry</b> : проверка генерации исключения <b>THROW06</b> (см. рисунок).
<b>TEST_UPDENTRY_07</b>	Тест функции <b>UpdEntry</b> : проверка генерации исключения <b>THROW07</b> (см. рисунок).
<b>TEST_UPDENTRY_08</b>	Тест функции <b>UpdEntry</b> : проверка генерации исключения <b>THROW08</b> (см. рисунок).
<b>TEST_DICTIONARY</b>	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 <b>Print</b> .

10. Макросы, управляющие условной компиляцией должны быть определены в заголовочном файле **stdafx.h**.
11. В файле **SE\_Lab05.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;
- поясните назначение, принцип работы операторов препроцессора **#** и **##** (стрингификации и конкатенации) и приведите пример использования;
- какие типы памяти используются приложением C++?
- что такое пространство имен?
- что такое исключение?
- поясните принцип связи инструкций **throw** и **catch** при 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 операторов **try**, **catch** и **throw**.