# Описание

Целью данного проекта являлась разработка и реализация API, который позволяет осуществлять работу с базой данных, фактически создание API для простейшей СУБД.

Реализация проекта велась на языке программирования C(Си).

API включает в себя следующие возможности для организации работы с БД на программном уровне:

* **Работа с узлами:**
* [Создание нового узла в базе данных (в т.ч. создание новой базы данных)](#_Функция_inputTree)
* [Удаление узла из базы данных (в т.ч. удаление базы данных)](#_Функция_delete)
* [Поиск узла в базе данных](#_Функция_findNode)
* [Поиск среди потомков узла](#_Функция_goToNode)
* [Переход в узел по пути](#_Функция_goToPath)
* Сохранение базы данных в файл
* Чтение базы данных из файла
* [Вывод всех потомков узла](#_Функция_directory)
* [Вывод всех ветвей дерева БД (от опр. вершины)](#_Функция_pathList)
* [Вывод пути от вершины дерева до узла](#_Функция_way)
* **Работа со значениями и ключами(спецификаторами):**
* [Добавление значения в узел](#_Функция_add_value)
* [Удаление значения из узла](#_Функция_delete_val)
* [Удаление всех значений (значений определенного типа) из узла](#_Функция_delete_all_values)
* [Копирование значения](#_Функция_copy)
* [Вставка значения](#_Функция_paste)
* Поиск значения в узле
* Вывод всех значений узла
* Вывод значений определенного типа в узле
* Изменение значения и/или типа у данного спецификатора
* Изменение спецификатора у значения
* Вывод спецификатора у значения

Подробное описание функций в приложениях.

**Реализация**

Для работы с функционалом необходимо использовать среду разработки Microsoft Visual Studio.

Поддерживается версия Microsoft Visual Studio 2015.

* Заголовочный файл DataBase15.h
* Библиотека DataBase15.h

Совместимость с другими версиями не гарантируется.

# Функция input\_tree

*Прототип функции input\_tree*

NODE\* input\_tree(NODE\* currPtr, char\* newName)

*Заголовочный файл*

|  |  |
| --- | --- |
| Название и библиотека | Язык |
| DataBase15.h  DataBase15.lib | C |

*Описание*

Функция input\_tree создает в базе данных новый узел, являющийся потомком для текущего узла.

*Параметры*

* **currPtr**

Указатель на узел типа NODE, для которого требуется создать узел-потомок. Если передать NULL, то функция создаёт корень нового дерева

* **newName**

Имя в виде си-строки, которое будет присвоено вновь созданному узлу. Не может превышать 255 символов по длине. Имена узлов-братьев не могут совпадать

*Возвращаемое значение*

В случае успеха возвращается указатель на созданный объект типа NODE.

В случае ошибки возвращается NULL.

*Пример: исходный код программы Возвращаемый результат:*

**#include "DataBase15.h"

int main()

{

NODE\* root = NULL;

NODE\* newNode = NULL;

root = input\_tree(NULL, "root");

directory(root);

newNode = input\_tree(root, "NewNode");

directory(root);

getch();

}

# Функция delete

*Прототип функции delete*

int delete(NODE\*\* currentNode, NODE\*\* root)

*Заголовочный файл*

|  |  |
| --- | --- |
| Название и библиотека | Язык |
| DataBase15.h  DataBase15.lib | C |

*Описание*

Функция delete удаляет из базы данных узел со всеми его потомками. Удаление рекурсивно (т.е. удаляются также все потомки потомков и так далее)

*Параметры*

* **currentFolder**

Адрес указателя на узел типа NODE, к которому требуется применить операцию удаления

* **root**

Адрес указателя на корневой узел типа NODE

*Возвращаемое значение*

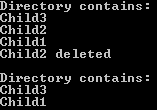
1 в случае успешного удаления

0 в случае ошибки (попытка удаления несуществующего узла)

*Пример: исходный код программы: Возвращаемый результат:*

**#include "DataBase15.h"

int main()

{

NODE\* root = NULL;

NODE\* temp = NULL;

int i;

root = input\_tree(NULL, "root");

input\_tree(root, "Child1");

temp = input\_tree(root, "Child2");

input\_tree(root, "Child3");

directory(root);

input\_tree(temp, "Child1");

i = delete(&temp, &root);

directory(root);

getch();

}

# Функция find\_node

*Прототип функции find\_node*

NODE\* find\_node(char\* fileName, NODE\* beginf)

*Заголовочный файл*

|  |  |
| --- | --- |
| Название и библиотека | Язык |
| DataBase15.h  DataBase15.lib | C |

*Описание*

Функция find\_node осуществляет поиск узла с именем fileName, среди всех узлов-потомков (в т.ч. их потомков и так далее, т.е. рекурсивно).

*Параметры*

* **beginf**

Указатель на узел типа NODE, на уровнях ниже которого среди связанных с ним узлов необходимо обнаружить искомый.

* **fileName**

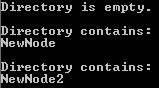
Си-строка с искомым именем объекта

*Возвращаемое значение*

В случае успеха возвращается указатель на искомый объект типа NODE.

В случае ошибки возвращается NULL.

*Пример: исходный код программы Возвращаемый результат:*

**#include "DataBase15.h"

int main()

{

NODE\* root = NULL;

NODE\* NewNode = NULL;

root = input\_tree(NULL, "root");

directory(root);

NewNode = input\_tree(root, "NewNode");

directory(root);

NewNode = input\_tree(NewNode, "NewNode2");

NewNode = find\_node("NewNode", root);

directory(NewNode);

getch();

}

# Функция go\_to\_node

*Прототип функции go\_to\_node*

NODE\* go\_to\_node(char\* s1, NODE\* node)

*Заголовочный файл*

|  |  |
| --- | --- |
| Название и библиотека | Язык |
| DataBase15.h  DataBase15.lib | C |

*Описание*

Функция go\_to\_node осуществляет поиск узла с именем s1, среди всех узлов-потомков текущего узла.

*Параметры*

* **node**

Указатель на узел типа NODE, среди потомков которого необходимо обнаружить искомый узел.

* **s1**

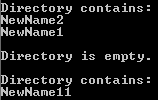
Си-строка с искомым именем объекта

*Возвращаемое значение*

В случае успеха возвращается указатель на искомый объект типа NODE.

В случае ошибки возвращается NULL.

*Пример: исходный код программы Возвращаемый результат:*

**#include "DataBase15.h"

int main()

{

NODE\* root = NULL;

NODE\* temp = NULL;

root = input\_tree(NULL, "root");

temp = input\_tree(root, "NewName1");

input\_tree(root, "NewName2");

temp = input\_tree(temp, "NewName11");

directory(root);

directory(temp);

temp = go\_to\_node("NewName1", root);

directory(temp);

getch();

}

# Функция go\_to\_path

*Прототип функции go\_to\_path*

NODE\* go\_to\_path(NODE \* root, char\* path)

*Заголовочный файл*

|  |  |
| --- | --- |
| Название и библиотека | Язык |
| DataBase15.h  DataBase15.lib | C |

*Описание*

Функция go\_to\_path возвращает указатель на узел базы данных расположенный по указанному пути.

*Параметры*

* **root**

Указатель на узел типа NODE, адрес которого нужно получить

* **path**

Си-строка, содержащая путь к узлу Root. Узлы в пути нужно разделять “.” (прим. MilkyWay.SolarSystem.Earth). Необходимо передавать указатель на начало строки. Передача самой строки непосредственно (прим. “MilkyWay.SolarSystem.Earth” вместо char s[]=“MilkyWay.SolarSystem.Earth”) недопустимо.

*Возвращаемое значение*

В случае успеха возвращается указатель на узел базы данных расположенный по указанному пути.

В случае ошибки возвращается NULL

*Пример: исходный код программы Возвращаемый результат:*

#include "DataBase15.h"

int main()

{

NODE\* root = NULL;

NODE\* temp = NULL;

char s[] = "MilkyWay.SolarSystem";

root = input\_tree(NULL, "MilkyWay");

temp=input\_tree(root, "SolarSystem");

add\_value(temp, "Wide", INT, "100");

temp = input\_tree(temp, "Earth");

temp=go\_to\_path(root, s);

print\_values(temp, ALL);

getch();

}

# Функция directory

*Прототип функции directory*

void directory(NODE \*current)

*Заголовочный файл*

|  |  |
| --- | --- |
| Название и библиотека | Язык |
| DataBase15.h  DataBase15.lib | C |

*Описание*

Функция directory выводит на экран всех потомков узла.

*Параметры*

* **current**

Указатель на узел типа NODE, для которого требуется распечатать потомков

*Возвращаемое значение*

Функция типа void. Нет возвращаемых значений

*Пример: исходный код программы Возвращаемый результат:*

**#include "DataBase15.h"

int main()

{

NODE\* root = NULL;

NODE\* NewNode = NULL;

root = input\_tree(NULL, "root");

directory(root);

NewNode = input\_tree(root, "NewNode");

directory(root);

getch();

}

# Функция path\_list

*Прототип функции path\_list*

int path\_list(NODE \* currentNode)

*Заголовочный файл*

|  |  |
| --- | --- |
| Название и библиотека | Язык |
| DataBase15.h  DataBase15.lib | C |

*Описание*

Функция path\_list выводит на экран представление БД в виде дерева .

*Параметры*

* **currentNode**

Указатель на узел типа NODE на вершину (корневой узел) дерева.

*Возвращаемое значение*

0 если дерево пустое

1 в случае успешного вывода на экран

*Пример: исходный код программы Возвращаемый результат:*

**#include "DataBase15.h"

int main()

{

NODE\* root = NULL;

NODE\* temp = NULL;

root = input\_tree(NULL, "MilkyWay");

temp=input\_tree(root, "SolarSystem");

input\_tree(temp, "Earth");

input\_tree(root, "System503");

path\_list(root);

getch();

}

# Функция way

*Прототип функции way*

void way(NODE \* beginf)

*Заголовочный файл*

|  |  |
| --- | --- |
| Название и библиотека | Язык |
| DataBase15.h  DataBase15.lib | C |

*Описание*

Функция way распечатывает путь от корневого узла до текущего.

*Параметры*

* **beginf**

Указатель на узел типа NODE, для которого требуется распечатать путь.

*Возвращаемое значение*

Функция типа void, нет возвращаемого значения.

*Пример: исходный код программы Возвращаемый результат:*

#include "DataBase15.h"

int main()

{

NODE\* root = NULL;

NODE\* temp = NULL;

int i = 1;

root = input\_tree(NULL, "root");

temp = input\_tree(root, "NewNode");

temp = input\_tree(temp, "NewNode2");

way(temp);

getch();

}

# Функция add\_value

*Прототип функции add\_value*

VALUE\* add\_value(NODE\* currentPtr, char\* qualf, TYPE type, char\* info)

*Заголовочный файл*

|  |  |
| --- | --- |
| Название и библиотека | Язык |
| DataBase15.h(VS2015) DataBase13.h(VS2013)  DataBase15.lib(VS2015) DataBase13.lib(VS2013) | C |

*Описание*

Функция add\_value добавляет новое значение для узла в базе данных.

*Параметры*

* **currentPtr**

Указатель на узел типа NODE, для которого требуется добавить значение.

* **qualf**

Спецификатор в виде си-строки, который будет присвоен вновь созданному значению. Не может превышать 255 символов по длине

* **type**

Тип значения (INT, CHAR, FLOAT, DOUBLE)

* **info**

Си-строка содержащая само значение

*Возвращаемое значение*

В случае успеха возвращается указатель на созданный объект типа VALUE.

В случае ошибки возвращается NULL.

*Пример: исходный код программы Возвращаемый результат:*

**#include "DataBase15.h"

int main()

{

NODE\* root = NULL;

root = input\_tree(NULL, "root");

add\_value(root, "Value1", INT, "2016");

add\_value(root, "Value2", INT, "15PI");

print\_values(root,ALL);

getch();

}

# Функция delete\_val

*Прототип функции delete\_val*

int delete\_val(VALUE\* toDelete, NODE\* start)

*Заголовочный файл*

|  |  |
| --- | --- |
| Название и библиотека | Язык |
| DataBase15.h  DataBase15.lib | C |

*Описание*

Функция delete\_val удаляет все значения/значения определённого типа из узла в БД.

*Параметры*

* **toDelete**

Указатель на значение, которое требуется удалить.

* **start**

Указатель на узел типа NODE, для которого требуется удалить значение.

*Возвращаемое значение*

0 в случае ошибки

1 в случае успешного удаления

*Пример: исходный код программы Возвращаемый результат:*

**#include "DataBase15.h"

int main()

{

NODE\* root = NULL;

VALUE\* temp = NULL;

root = input\_tree(NULL, "MilkyWay");

temp = add\_value(root, "1", INT, "259");

print\_values(root, ALL);

delete\_val(temp,root);

print\_values(root, ALL);

getch();

}

# Функция delete\_all\_values

*Прототип функции delete\_all\_values*

void delete\_all\_values(NODE \*currPtr, TYPE type)

*Заголовочный файл*

|  |  |
| --- | --- |
| Название и библиотека | Язык |
| DataBase15.h  DataBase15.lib | C |

*Описание*

Функция delete\_all\_values удаляет все значения/значения определённого типа из узла в БД.

*Параметры*

* **currPtr**

Указатель на узел типа NODE, для которого требуется удалить значения.

* **type**

Тип значения (INT, CHAR, FLOAT, DOUBLE)

*Возвращаемое значение*

Функция типа void. Нет возвращаемого значения

*Пример: исходный код программы Возвращаемый результат:*

#include "DataBase15.h"

**

int main()

{

NODE\* root = NULL;

root = input\_tree(NULL, "MilkyWay");

add\_value(root, "1", INT, "259");

add\_value(root, "2", CHAR, "Value");

add\_value(root, "3", INT, "346");

print\_values(root, ALL);

delete\_all\_values(root, INT);

print\_values(root, ALL);

getch();

}

# Функция copy

*Прототип функции copy*

VALUE\* copy(VALUE\* valueFrom)

*Заголовочный файл*

|  |  |
| --- | --- |
| Название и библиотека | Язык |
| DataBase15.h  DataBase15.lib | C |

*Описание*

Функция copy копирует значение из узла в буфер. Буфер должен соответствовать типу VALUE.

*Параметры*

* **valueFrom**

Указатель на значение типа VALUE, которое требуется скопировать

*Возвращаемое значение*

В случае успеха возвращается указатель типа VALUE на буфер со скопированным значением.

В случае неудачи возвращается NULL

*Пример: исходный код программы Возвращаемый результат:*

**#include "DataBase15.h"

int main()

{

NODE\* root = NULL;

NODE\* rootN = NULL;

VALUE\* temp = NULL;

root = input\_tree(NULL, "MilkyWay");

temp = add\_value(root, "1", INT, "259");

temp = copy(temp);

rootN = input\_tree(root, "Solar");

print\_values(rootN, ALL);

paste(rootN, temp);

print\_values(root, ALL);

print\_values(rootN, ALL);

getch();

}

# Функция paste

*Прототип функции paste*

int paste(NODE\* pasteTo, VALUE\* copy)

*Заголовочный файл*

|  |  |
| --- | --- |
| Название и библиотека | Язык |
| DataBase15.h  DataBase15.lib | C |

*Описание*

Функция paste добавляет указанное значение в узел

*Параметры*

* **pasteTo**

Указатель на узел типа NODE, в который требуется поместить значение

* **copy**

Указатель на значение типа VALUE, которое требуется поместить в узел

*Возвращаемое значение*

В случае успеха возвращается 2

При попытке вставить значение в узел, где уже существует значение с таким спецификатором возвращается 1

*Пример: исходный код программы Возвращаемый результат:*

#include "DataBase15.h"

int main()

{

NODE\* root = NULL;

NODE\* rootN = NULL;

VALUE\* temp = NULL;

root = input\_tree(NULL, "MilkyWay");

temp = add\_value(root, "1", INT, "259");

temp = copy(temp);

rootN = input\_tree(root, "Solar");

print\_values(rootN, ALL);

paste(rootN, temp);

print\_values(root, ALL);

print\_values(rootN, ALL);

getch();

}