|  |  |  |
| --- | --- | --- |
|  | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования FPMI_ngtu_neti_rgb_polya«Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра теоретической и прикладной информатики | | |
| Лабораторная работа № 4 | | |
| по дисциплине «Основы программирования» | | |
| **Операции над таблицами** | | |
|  | | |
|  | Бригада | ФИО |
| №2 | Веселый Денис Андреевич |
|  | Ворончук Илья Игоревич |
| Группа | ПМИ-32 |
|  |  |
|  |  |
| Преподаватель | Дворецкая виктория константиновна |
|  |  |
| Новосибирск,2024 | | |

**1) Цель работы**

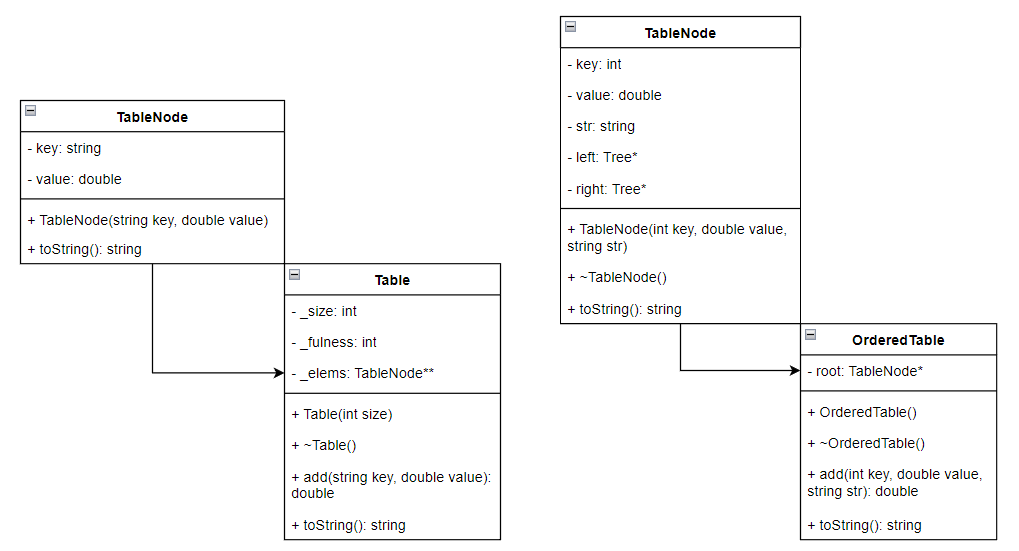
Получить практические навыки работы с таблицами и использования их при решении задач.

**2) Условие задачи**

* + 1. Элемент таблицы характеризуется двумя признаками: один – это последовательность не более чем из 15 символов и этот признак – ключ, другой признак – вещественное значение. Определить тип «таблица» и написать подпрограмму включения элемента в неупорядоченную таблицу.
    2. Элемент таблицы характеризуется тремя признаками: один –вещественное значение, второй – целое положительное число и этот признак – ключ элемента, третий – это последовательность не более чем из 10 символов. Определить тип «таблица» и написать подпрограмму включения элемента в упорядоченную таблицу.

**2.1) Алгоритм решения задачи**

* + 1. Таблица представлена в виде статического массива из 64 элементов. Коллизия решается методом рехеширования.
    2. Таблица представлена в виде упорядоченного бинарного дерева. При совпадении ключей, порядок добавления определяется по второму значению. При совпадении вторых значений, порядок определяется по третьему значению. Если элементы равны, то они добавляются друг за другом.

** 3) Стандартная схема программы**

**4) Код программы**

**Table.cpp**

#include "Table.hpp"

Table::TableNode::TableNode(std::string key, double value) : key{key}, value{value} {}

// преобразует структуру TableNode в строку

std::string Table::TableNode::toString()

{

return key + ": " + std::to\_string(value);

}

Table::Table(int size) : \_size{size + 1}, \_fulness{0}

{

\_elems = new Table::TableNode \*[\_size];

for (int i = 0; i < \_size; i++)

\_elems[i] = nullptr;

}

Table::~Table()

{

delete[] \_elems;

\_elems = nullptr;

}

// возвращает хэш в числовом виде; принимает на вход ключ и порядок хэширования

int Table::getHash(std::string key, bool h)

{

int hash = 0;

for (int i = 0; i < key.size(); i++)

hash = (hash << 5) - hash + int(pow(-1, h)) \* key[i];

return abs(hash);

}

// возвращает индекс; принимает на вход ключ

int Table::getIndex(std::string key)

{

return getHash(key, 0) % \_size;

}

// добавляет новый элемент (ключ, значение) в таблицу

double Table::add(std::string key, double value)

{

if (\_fulness >= \_size - 1)

return false;

int index = getIndex(key);

for (int i = 0; i < \_size; i++)

{

if (!\_elems[index])

{

\_elems[index] = new TableNode(key, value);

\_fulness++;

return value;

}

index = (getHash(key, 0) + i \* getHash(key, 1)) % \_size;

}

return NAN;

}

// преобразует таблицу в строку

std::string Table::toString()

{

std::string result;

for (int i = 0; i < \_size; i++)

if (\_elems[i])

result += \_elems[i]->toString() + "\n";

return result;

}

**OrderedTable.cpp**

#include "OrderedTable.hpp"

#include <functional>

OrderedTable::TableNode::TableNode(int key, double value, std::string str) : key{key}, value{value}, str{str}, left{nullptr}, right{nullptr} {}

OrderedTable::TableNode::~TableNode()

{

delete left, right;

left, right = nullptr, nullptr;

}

// преобразует TableNode в строку

std::string OrderedTable::TableNode::toString()

{

return std::to\_string(key) + ": [" + std::to\_string(value) + ", " + str + "]";

}

OrderedTable::OrderedTable() : \_root{nullptr} {}

OrderedTable::~OrderedTable()

{

delete \_root;

\_root = nullptr;

}

// добавляет новый элемент (ключ, значение, строка) в упорядоченную таблицу

double OrderedTable::add(int key, double value, std::string str)

{

if (!\_root)

{

\_root = new TableNode(key, value, str);

return value;

}

TableNode \*node = new TableNode(key, value, str);

TableNode \*cur = \_root;

TableNode \*buff;

while (cur)

{

buff = cur;

cur = key > cur->key || (key == cur->key && value > cur->value) || (value == cur->value && str > cur->str)

? cur->left

: cur->right;

}

key > buff->key || (key == buff->key && value > buff->value) || (value == buff->value && str > buff->str)

? buff->left = node

: buff->right = node;

return value;

}

// преобразует упорядоченную таблицу в строку

std::string OrderedTable::toString()

{

if (!\_root)

return "";

std::string result;

std::function<void(TableNode \*)> func;

func = [&](TableNode \*tableNode)

{

if (tableNode->right)

func(tableNode->right);

result += tableNode->toString() + "\n";

if (tableNode->left)

func(tableNode->left);

};

func(\_root);

return result;

}

**5) Тестирование программы**

|  |  |  |
| --- | --- | --- |
| № | Команды | Комментарий |
| 1 | add eloy 80  add cream 90.14  add jimihendrix 50  add moodyblues 600.8  prt | 1. Добавление элементов вида (ключ, значение) в неупорядоченную таблицу. 2. Вывод результата в консоль. |
| 2 | add --order 4 10 traffic  add --order 9 1 yes  add --order 400 11 greenday  add --order 8 3 the5:55  prt --order | 1. Добавление элементов вида (ключ, значение, строка) в упорядоченную таблицу. 2. Вывод результата в консоль. |
| 3 | add styx 89  add omega 777  add styx 41.3  add styx 12  prt | 1. Добавление элементов вида (ключ, значение) в неупорядоченную таблицу. Среди них – 3 имеют одинаковый ключ. 2. Вывод результата в консоль. |

**6) Результаты отладки**

Результаты работы программы показали, что программа работает успешно.