МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»

(БГТУ им. В.Г. Шухова)

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Дисциплина: Алгоритмы и структуры данных

Лабораторная работа № 6

**Структуры данных «стек» и «очередь» (Pascal/С)**

|  |  |
| --- | --- |
|  | Выполнила: ст. группы ПВ-21  Зановская А.И.  Проверил: Синюк В.Г. |

Белгород

2017

Л а б о р а т о р н а я р а б о т а № 8

**Структуры данных «таблица» (Pascal/С)**

**Цель работы:** изучить СД «таблица», научиться их программно реализовывать и использовать.

З а д а н и е

1. Для СД «таблица» определить:

1.1. Абстрактный уровень представления СД:

1.1.1. Характер организованности и изменчивости.

Динамическая структура

1.1.2. Набор допустимых операций.

Инициализация, включение элемента, исключение элемента с заданным ключом, чтение элемента с заданным ключом, изменение элемента с заданным ключом, проверка пустоты таблицы, уничтожение таблицы.

1.2. Физический уровень представления СД:

1.2.1. Схему хранения. Связная

1.2.2. Объем памяти, занимаемый экземпляром СД. Зависит от представления и базового типа.

1.2.3. Формат внутреннего представления СД и способ его интерпретации. Зависит от представления.

1.2.4. Характеристику допустимых значений. Зависит от базового типа.

1.2.5. Тип доступа к элементам. Последовательный.

1. 3. Логический уровень представления СД:

Способ описания СД и экземпляра СД на языке программирования.

Table T

2. Реализовать СД «таблица» в соответствии с вариантом индивидуального задания (табл.18) в виде модулей на языках Pascal и С.

3. Разработать программы на языках Pascal и С для решения задачи в соответствии с вариантом индивидуального задания (см. табл.18) с использованием модулей, полученных в результате выполнения пункта 2 задания.

Таблица 18

**Варианты индивидуальных заданий**

|  |  |  |
| --- | --- | --- |
| Номер варианта | Номер модуля | Задача |
| 10 | 5 | 10 |

**Задачи**

10. Текстовый файл содержит формулу химического соединения (не обязательно существующего). Название элемента начинается с большой буквы. Если количество некоторых элементов в соединении больше одного, то после названия указывается число элементов. Примеры формул:

H2O , HCl , O2 , H2SO4 , O4H2S , NaCl .

Для правильных формул получить молекулярный вес соединения. Использовать таблицу, ключ элемента которой представляет собой название химического элемента, а информационная часть — вес элемента. Информацию в таблицу загрузить из текстового файла.

**Модуль**

5. Упорядоченная таблица на односвязном линейном списке.

**Спецификация СД на языке C:**

#if !defined(\_\_TABLE5\_H)

#define \_\_TABLE5\_H

#include "List.h"

const TableOk = 0;

const TableNotMem = 1;

const TableUnder = 2;

const TableNoKey = 3;

typedef char[3] TKey; // Определить тип ключа

int TableError; // 0..3

void InitTable(Table \*T);

int EmptyTable(Table \*T); // Возвращает 1 , если таблица пуста, иначе — 0

int PutTable(Table \*T, BaseType E); /\* Включение элемента в таблицу. Возвращает 1 , если элемент включен в таблицу, иначе — 0 (если в таблице уже есть элемент с заданным ключем или нехватает памяти) \*/

int GetTable(Table \*T, BaseType \*E, T\_Key Key); /\* Исключение элемента. Возвращает 1 , если элемент с ключем Key был в таблице, иначе — 0 \*/

int ReadTable(Table \*T, BaseType \*E, T\_Key Key); /\* Чтение элемента. Возвращает 1 , если элемент с ключем Key есть в таблице, иначе — 0 \*/

int WriteTable(Table \*T, BaseType \*E, T\_Key Key); /\* Изменение элемента. Возвращает 1 , если элемент с ключем Key есть в таблице, иначе — 0 \*/

#endif

**TABLE5.h**

#include <stdio.h>

#include <stdlib.h>

#include "TABLE5.h"

typedef char[3] TKey;

typedef int TableBaseType;

typedef struct element \*ptrel;

typedef struct

{

TableBaseType data;

TKey key;

ptrel next;

} element;

typedef struct

{

ptrel Start;

ptrel ptr;

}

void TableInit(Table \*T)

{

T->Start = (ptrel)malloc(sizeof(element));

if (T->Start==NULL)

{

TableListError=TableListNotMem;

return;

}

T->ptr=T->Start;

T->ptr->next=NULL;

TableListError=tableOk;

}

//Включение элемента в таблицу

int TablePut(Table \*T, element \*E, TKey Key)

{

int c;

ptrel tmp = (ptrel)malloc(sizeof(element));

if (tmp==NULL)

{

TableListError=TableListNotMem;

return 0;

}

T->ptr=T->Start;

while (T->ptr->next!=NULL && c=strcmp(T->ptr->next->key, Key)<0)

T->ptr=T->ptr->next;

if (c<0)

{

tmp->data=E->data;

tmp->key=E->key;

tmp->next=T->ptr->next;

T->ptr->next=tmp;

T->ptr=T->ptr->next;

TableListError=TableListOk;

return 1;

}

else

return 0;

}

//Исключение элемента из таблицы

int TableGet(Table \*T, element \*E, TKey Key)

{

int c;

if (TableEmpty(T))

{

TableListError=TableListEmpty;

Return 0;

}

T->ptr=T->Start;

while (T->ptr->next!=NULL && c=strcmp(T->ptr->next->key, Key)<0)

T->ptr=T->ptr->next;

if (c==0)

{

ptrel tmp;

tmp=T->ptr->next;

T->ptr->next=tmp->next;

E->data = tmp->data;

E->key = tmp->key;

free(tmp);

return 1;

}

else

return 0;

}

int TableRead(Table \*T, element \*E, TKey Key)

{

int c;

if (TableEmpty(T))

{

TableListError=TableListEmpty;

return 0;

}

T->ptr=T->Start;

while (T->ptr->next!=NULL && c=strcmp(T->ptr->next->key, Key)<=0)

T->ptr=T->ptr->next;

if (c==0)

{

E->data = T->ptr->data;

E->key = T->ptr->key;

return 1;

}

else

{

TableListError=TableNoKey;

return 0;

}

}

int TableWrite(Table \*T, element \*E, TKey Key)

{

int c;

if (TableEmpty(T))

{

TableListError=TableListEmpty;

return 0;

}

T->ptr=T->Start;

while (T->ptr->next!=NULL && c=strcmp(T->ptr->next->key, Key)<=0)

T->ptr=T->ptr->next;

if (c==0)

{

T->ptr->data = E->data;

T->ptr->key = E->key;

return 1;

}

else

return 0;

}

//Таблица пуста

int TableEmpty (Table \*T)

{

return (T->Start->next == NULL);

}

void DoneTable(Table \*T)

{

DoneList(T);

TableError = ListError;

}

int strcmp(char \*s, char \*t)

{

for (;\*s == \*t; s++, t++)

if (\*s == '\0')

return 0;

return \*s - \*t;

}

**main.c**

int TableInput (Table \*T)

{

FILE \*f;

char name[]="1.txt";

f=fopen(name,"r+b");

int i, fl; element E;

if(!TableInit(T))

return 0;

while (!feof(f) && TableListError!=TableNotMem)

{

fscanf(f, "%s",E.key);

fscanf(f, "%d",&E.data);

TablePut(T, E, E.Key);

}

return TableListError!=TableNotMem;

}

int MolMass (Table \*T, char \*s)

{

char t[5];

int i = 0, fl = 1, k = 1, M = 0, j;

element E;

while (fl)

{

j = 0; k = 1;

t[j] = s[i];

i++;

if (s[i]>='a' && s[i]<='z')

while (s[i]>='a' && s[i]<='z')

{

t[j] = s[i];

j++; i++;

}

t[j]='\0';

if (s[i] >= '0' && s[i] <= '9')

while (s[i]>='0' && s[i]<='9')

{

k = s[i] - '0' + k\*10;

i++;

}

fl=TableRead(T, &E, t);

if(fl)

M+=k\*E->data;

else

return;

fl = s[i]!='\0';

}

return M;

}

int main ()

{

char name[]="data.txt";

char Form[15];

FILE \*f;

f=fopen(name,"r+b");

Table T;

TableInput(&T);

int Mass;

while (!feof(f))

{

fscanf(f, "%s",Form);

Mass=MolMass(T,Form);

if (fabs(Mass)<0.01)

printf ("Incorrect formul\n");

else

printf ("%s %d\n", Form, Mass);

}

}



