**Федеральное агентство по образованию РФ**

**Государственное образовательное учреждение высшего профессионального образования**

**Белгородский Государственный Технологический Университет им В. Г. Шухова**

Кафедра программного обеспечения вычислительной техники и

автоматизированных систем.

Лабораторная работа №8 по дисциплине

«Алгоритмы и структуры данных»

**Структура данных «таблица»**

Выполнил:

студент группы ПВ-21

Браткова И.О.

Белгород 2017 г.

**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.

1.1.2. Набор допустимых операций.

1.2. Физический уровень представления СД:

1.2.1. Схему хранения.

1.2.2. Объем памяти, занимаемый экземпляром СД.

1.2.3. Формат внутреннего представления СД и способ его интерпретации.

1.2.4. Характеристику допустимых значений.

1.2.5. Тип доступа к элементам.

1. 3. Логический уровень представления СД:

Способ описания СД и экземпляра СД на языке программирования.

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

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

**Таблица**

1.1. Абстрактный уровень представления СД:

1.1.1. Характер организованности и изменчивости: **динамическая, линейная структура данных представляющая собой множество.**

1.1.2. Набор допустимых операций: **инициализация, включение элемента, исключение элемента с заданным ключом, чтение элемента с заданным ключом, удаление элемента с заданным ключом, изменение элемента с заданным ключом, проверка таблицы на пустоту, уничтожение таблицы.**

1.2. Физический уровень представления СД:

1.2.1. Схему хранения: **связная или последовательная схема хранения.**

1.2.2. Объем памяти, занимаемый экземпляром СД: **Зависит от количества элементов в таблице V = k \* size; k – количество элементов, size – размер занимаемый информационной частью элемента и ключом.**

1.2.3. Формат внутреннего представления СД и способ его интерпретации: **Может храниться как в статической, так и в динамической памяти.**

1.2.4. Характеристику допустимых значений: **CAR(Table) = CAR(BaseType)0 + CAR(BaseType)1+ … + CAR(BaseType)max, где CAR(BaseType) — кардинальное число тела элемента таблицы типа BaseType, max — количество различных ключей в таблице.**

1.2.5. Тип доступа к элементам: **прямой (по ключу).**

1.3. Логический уровень представления СД.

Способ описания СД и экземпляра СД на языке программирования:

**typedef <базовый тип элемента таблицы> TableBaseType;**

**typedef <базовый тип ключа таблицы> TableKey;**

**typedef struct Element**

**{**

**TableBaseType data;**

**TableKey key;**

**};**

**typedef struct Table**

**{**

**TableEl \*buf;**

**unsigned uk;**

**unsigned N;**

**};**

“**table.h**”:

#ifndef TABLE\_H\_INCLUDED

#define TABLE\_H\_INCLUDED

#include <stdlib.h>

#define TSize 40

const short TableOk = 0;

const short TableFull = 1;

const short TableEmpty = 2;

static short TableError;

typedef struct {

int \*val; *// номера ключевых слов, которые могут стоять после текущего ключевого слова*

int kol; *// количество ключевых слов, которые могут стоять после текущего*

} BaseType;

typedef char\* T\_key;

typedef struct {

T\_key key; *//ключевое слово*

BaseType data;

} element;

typedef unsigned ptrel;

typedef struct {

element TMem[TSize];

ptrel ptr;

int N;

} Table;

void InitTable(Table \*T); *// Инициализация таблицы*

int search(Table T, T\_key key) ; *//* *Поиск элемента с заданным ключем*

int PutTable(Table \*T, T\_key key, BaseType data) ; *// Включение элемента в таблицу*

int getTable(Table \*T, T\_key key, BaseType \*data) ; *// Исключение элемента из таблицы*

int ReadTable(Table T, T\_key key, BaseType \*data) ; *// Чтение элемента таблицы*

int WriteTable(Table \*T, T\_key key, BaseType \*data); *// Изменение данных в таблице*

int isEmptyTable(Table T); *// Таблица пуста/не пуста*

int isFullTable(Table T); *// Таблица переполнена/не переполнена*

#endif // TABLE\_H\_INCLUDED

“**table.cpp**”:

#include "table.h"

void initTable(Table \*T){ T->N=0; }

int search(Table T, T\_key key)

{ int i;

for(i=0; i<T.N ;i++)

if(strcmp(T.TMem[i].key,key)==0)

return i;

return -1;

}

int PutTable(Table \*T, T\_key key, BaseType data)

{ int i;

if(isFullTable(\*T))

{ TableError=TableFull;

return 0; }

i=search(\*T,key);

if(i==-1)

{ T->N++;

T->ptr=T->N-1;

T->TMem[T->ptr].key=(char\*)malloc(strlen(key)+1);

strcpy(T->TMem[T->ptr].key,key);

T->TMem[T->ptr].data=data;

return 1;

}

else

return 0;

}

int getTable(Table \*T, T\_key key, BaseType \*data)

{ int i,k=0,j;

if(isEmptyTable(\*T))

{ TableError=TableEmpty;

return 0; }

i=search((\*T),key);

if(i!=-1)

{ \*data=T->TMem[i].data;

k=0;

for(j=0; j<T->N; j++)

if(j!=i)

{ T->TMem[k]=T->TMem[j];

k++;

}

T->N--;

return 1;

}

else

return 0;

}

int ReadTable(Table T, T\_key key, BaseType \*data)

{ int i;

if(isEmptyTable(T))

{ TableError=TableEmpty;

return 0;

}

i=search(T,key);

if(i!=-1)

{ \*data=T.TMem[i].data;

return 1;

}

else

return 0;

}

int WriteTable(Table \*T, T\_key key, BaseType \*data)

{ int i;

if(isEmptyTable(\*T))

{ TableError=TableEmpty;

return 0;

}

i=search((\*T),key);

if(i!=-1)

{ BaseType t=T->TMem[i].data;

T->TMem[i].data=(\*data);

\*data=t;

return 1;

}

else

return 0;

}

int isEmptyTable(Table T)

{ return(T.N==0);}

int isFullTable(Table T)

{ return(T.N==TSize);}

4. Текст программы на некотором алгоритмическом языке может содержать символы-разделители, служебные слова, числовые константы и идентификаторы (слова, начинающиеся не с цифры и не являющиеся служебными). Проверить ошибки в записи идентификаторов и констант и сочетаемость рядом стоящих служебных слов, символов-разделителей, числовых констант и идентификаторов. Например, пары «BEGIN x» и «ELSE BEGIN» — сочетаемы, а пары «x BEGIN» и «BEGIN ELSE» — несочетаемы. Проверку констант выполнить с помощью стандартной процедуры VAL. Проверку сочетаемости — с помощью таблицы сочетаемости. Ключ элемента таблицы — служебное слово или символ-разделитель, информационная часть — код служебного слова или символа-разделителя и множество кодов слов, которые могут непосредственно следовать за ключевым словом. Код константы — 0, идентификатора — 1. Для констант и идентификаторов также сформировать множества кодов слов, которые могут следовать за ними. Информацию для таблицы сочетаемости прочитать из файла.

“**main.cpp**”:

#include "table.h"

#include <stdio.h>

#include <string.h>

#include <locale.h>

void ReadFile(Table \*T)

{ FILE \*f;

BaseType data;

int i;

T\_key key;

f=fopen("data.txt","r");

while(!feof(f))

{ fscanf(f,"%s",key);

fscanf(f,"%i",&data.kol);

data.val=(int\*)malloc(data.kol\*sizeof(int));

for(i=0; i<data.kol; i++)

fscanf(f,"%i",&data.val[i]);

i=PutTable(T,key,data);

} }

int proverka(char \*s)

{ int i=0;

while(s[i])

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

i++;

else

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

return 0;

else

return -1;

return 1;

}

void word(char \*s, FILE \*f)

{ char c; int i,flag=0; fscanf(f,"%c",&c); i=0;

while(c==' ' || c=='\r' || c=='\n') fscanf(f,"%c",&c);

if((c>32 && c<=47) || (c>=58 && c<=61) || c==96) s[i++]=c;

if(c==':') { fscanf(f,"%c",&c);

if(c=='=') s[i++]=c;

else fseek(f,-1,SEEK\_CUR); c=s[i-2];

}

while((c>='0' && c<='9') || (c>='a' && c<='z') || (c>='A' && c<='Z'))

{ s[i++]=c; fscanf(f,"%c",&c); flag++; }

if(flag) fseek(f,-1,SEEK\_CUR); s[i]='\0';

}

int chislo(char \*s)

{ while(\*s) if(\*s>='0' && \*s<='9') s++;

else return 0; return 1; }

int sochet(Table T, char \*name)

{ char s1[20],s2[20],c;

int i,j,k,flag; FILE \*f; f=fopen(name,"rb");

word(s1,f); while(!feof(f))

{ word(s2,f); flag=0; if((j=search(T,s1))==-1)

if(proverka(s1)==-1) return 0;

else if(proverka(s1)==0)

{ if(!chislo(s1))

return 0; s1[0]='0'; s1[1]='\0';

j=search(T,s1); }

else { s1[0]='1'; s1[1]='\0'; j=search(T,s1); }

if((k=search(T,s2))==-1)

if(proverka(s2)==-1) return 0;

else

if(proverka(s2)==0)

{ if(!chislo(s2))return 0; s2[0]='0'; s2[1]='\0';

k=search(T,s2);

}

else

{ s2[0]='1'; s2[1]='\0'; k=search(T,s2);

}

for(i=0; i<T.TMem[j].data.kol;i++)

if(k==T.TMem[j].data.val[i]) flag++;

if(flag==0) if(T.TMem[j].data.kol==0) return 1;

else return 0; strcpy(s1,s2);

}

}

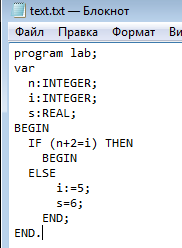
int main()

{ Table T;

int i;

char name[20];

scanf("%i",i);

 printf("input name file");

scanf("%s",name);

initTable(&T);

ReadFile(&T);

PrintTable(T);

i=sochet(T,name);

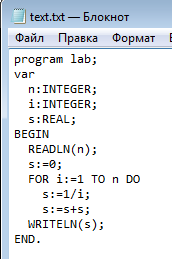
if(i==1)

printf("OK");

else

printf("ERROR");

return 1;

 }

