Національний Технічний Університет України

“Київский Політехнічний Інститут”

Факультет Інформатики та Обчислювальної Техніки

Кафедра обчислювальної техніки

**Лабораторна робота №1**

по курсу «Системне програмування»

Виконав:

студент 3 курсу

ФІОТ гр. ІО-92

Петрук Вадим

Киев 2011

МЕТОДИ РОБОТИ З ТАБЛИЦЯМИ, ЇХ ВПОРЯДКУВАННЯ І ПРИСКОРЕНА ОБРОБКА

**Мета роботи:** Вивчення типів таблиць в системних прог­ра­мах і конструкцій базової мови програмування для їх визначення. Пошук за прямою адресою. Основні типи залежностей та відно­шень, які реалізуються через пошук в таблицях системних програм. Лінійний та двійковий пошук. Вимоги до унікальності ключів.

Завдання на роботу

Завдання на підготовку до роботи на комп’ютері

1. Визначити варіант завдання для основних і додаткової задачі за Табл.1.2.

2. Розробити структуру даних для елементу таблиці згідно з варіантом за Табл.1.2. Якщо за варіантом треба використати тип даних, який визначається програмістом (enum, struct, union), то цей тип необхідно визначити перед використанням. Перед початком заняття показати викладачу *текст опису струк­тури і необхідних типів даних функціональної частини поля*.

3. Ознайомитись з шаблоном програмного проекту spLb1. Настроїти відповідні дані в програмному проекті на мові С.

4. Настроїти функції пошуку за прямою адресою, а також лінійного та двійкового пошуку, відмітки про вилучення, упаку­ван­ня таблиці, впорядкування таблиці і вставку до таблиці за значен­ням ключових полів з використанням різних методів.

* для таблиці з доступом за прямою адресою (1 бал за нала­годжену програму) – налагодження виконати в файлах tables.h, table.cpp і vistab.cpp;
* для таблиці з доступом за ключовим полем або групою полів за методом лінійного пошуку (1 бал за налагоджену програму) – налагодження виконати в файлі tables.h і vistab.cpp;

5. Настроїти програми відображення з модуля vistab так, щоб вони да­вали можливість відтворювати потрібні типи даних. Перед по­чат­ком заняття показати викладачу *текст оператора виве­ден­ня функціональних полів*.

6. Скласти алгоритми функцій вставки, вилучення і корекції в таблицях на базі методу двійкового пошуку і підготувати прог­рам­ні модулі, які забезпечують вставку, вилучення і корекцію еле­мен­та таблиці за значенням ключового елемента: для таблиці з досту­пом за ключовим полем або групою полів за методом двій­ко­вого пошуку (по 1 балу за кожну налагоджену функцію) – заго­тов­ки для функцій insBin, delBin та updBin в кінці модуля tables.cpp.

7. Підготувати програмні модулі для додаткової задачі: пошуку з порівнянням за мірою близькості за варіантом з таблиці 1.3 (1 бал за налагоджену програму).

8. Підготувати програмний модуль контрольної задачі, який виконує задані варіанти програм пошуку і вставки в таблицю за значенням ключового елемента, а також додаткових задач і дозволяє перевірити коректність виконання програм.

Завдання на роботу на комп’ютері

9. Побудувати програмний проект, ввівши програмні модулі у відповідні файли проекту і налагодити синтаксис.

10. Побудувати виконавчий модуль тестової програми і нала­го­дити змістовне виконання програми.

11. Одержати результати виконання, проаналізувати їх і зро­би­ти висновки.

Порядок вибору варіанту:

За останньою цифрою номера залікової книжки або за поряд­ковим номером студента в списку підгрупи визначте варіант завдання для задач за табл. 1.2 та додаткового завдання за табл.1.3.

*Таблиця* 1.2

Варіанти завдань для виконання пошуку

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № вар. | Тип ключа для прямої адреси | Тип ключа для інших видів пошуку | Тип функціо­нального поля | Тип вибірки |
| 19 | unsigned long | char\*\_ unsigned long | enum | Всі |

Варіанти типу вибірки означають:

* “Всі” – складання програми багатоваріантного пошуку всіх можливих результатів для ключа з повтореннями значень;

**Лістинг програми:**

**Test**

**package** lab1\_;

**public** **class** Test {

**public** **static** **void** main(String[] args){

**try**{

String[] columns = {"Key","Day of week"}; // columns

Long[] keys = {(**long**) 5,(**long**)6, (**long**)3, (**long**)4}; //keys

EnumField[] days = {EnumField.*SATURDAY*, EnumField.*WEDNESDAY*,

EnumField.*THURSDAY*, EnumField.*WEDNESDAY*}; //fields

Table table1 = **new** Table(columns,keys,days); // new table

table1.printTable(); // print table

Row row1 =table1.getElement(0); // get element

System.*out*.println("key = "+row1.getKey()+"; func = "+ row1.getFuncField());

Long a = **new** Long(5);

EnumField day = EnumField.*MONDAY*;

table1.addElement(a, day, 0); // add element

table1.deleteElement(1); // remove element

table1.updateElement(a, day, 0); // update element

table1.printTable(); // print table

Row[] someRows1 = table1.selectByLin(EnumField.*WEDNESDAY*); // select by lineary search by field

**for** (**int** i=0; i<someRows1.length; i++){

System.*out*.println(someRows1[i].toString());

}

Row someRow = table1.selectByLin((**long**)1); // select by -"- bu key

System.*out*.println(someRow.toString());

table1.sort(); // table sort

table1.printTable();

Row someRow1 = table1.selectKeyByBin((**long**)8); // select by bin search by key

System.*out*.println("key = "+someRow1.getKey()+"; func = "+ someRow1.getFuncField());

}**catch** (TableFormatException e){

System.*out*.println(e.getMessage());

}

}

}

**Table**

**package** lab1\_;

**import** java.util.ArrayList;

**public** **class** Table {

**private** ArrayList<String> columns = **new** ArrayList<String>();

**private** ArrayList<Long> keys = **new** ArrayList<Long>();

**private** ArrayList<EnumField> fStrs = **new** ArrayList<EnumField>();

**public** Table(String[] columns,Long[] keyStr, EnumField[] fStr) **throws** TableFormatException{

**if** (columns.length == 2){

**for** (**int** i=0; i<columns.length; i++){

**this**.columns.add(columns[i]);

}

}**else**{

**throw** **new** TableFormatException(" WrongTableFormat!Number of columns must be 2!");

}

**if** (keyStr.length == fStr.length){

**for** (**int** i=0; i<keyStr.length; i++){

keys.add(keyStr[i]);

fStrs.add(fStr[i]);

}

}**else**{

System.*out*.println("Number of keys dont equels number of functional fields!");

**throw** **new** TableFormatException("Number of keys dont equels number of functional fields!");

}

}

// вибірка за прямою адресою елемента № nElm

**public** Row getElement(**int** nElm){

Row row = **new** Row(keys.get(nElm), fStrs.get(nElm));

**return** row;

}

// включення за прямою адресою в позицію(адресу) № nElm

**public** **void** addElement(Long keyStr, EnumField fStr, **int** nElm){

// System.out.println(isKey(keyStr));

**if** (!(isKey(keyStr))){

keys.add(nElm,keyStr);

fStrs.add(nElm, fStr);

}**else**{

System.*out*.println("Element with this key '"+ keyStr+"' is in this table!!!");

System.*out*.println("Element not added!");

//throw TableFormatExeption("Element with this key '"+ keyStr+"' is in this table!!!")

// Apply automatic key change?Y/N

}

}

// включення за прямою адресою в позицію(адресу) № nElm

**public** **void** addElement(Row row, **int** nElm){

**this**.addElement(row.getKey(), row.getFuncField(), nElm);

}

// вилучення за прямою адресою елемента № nElm

**public** **void** deleteElement(**int** nElm){

keys.remove(nElm);

fStrs.remove(nElm);

}

// корекція за прямою адресою елемента № nElm

**public** **void** updateElement(Long keyStr, EnumField fStr, **int** nElm){

**if** ((isKey(keyStr)) & (keys.get(nElm) != keyStr)){

System.*out*.println("Correction is impossible! Element with this key is already in table!");

}**else**{

keys.remove(nElm);

fStrs.remove(nElm);

keys.add(nElm, keyStr);

fStrs.add(nElm, fStr);

}

}

**public** **void** updateElement(Long keyStr, **int** nElm){

**this**.updateElement(keyStr, fStrs.get(nElm), nElm);

}

**public** **void** updateElement(EnumField fStr, **int** nElm){

**this**.updateElement(keys.get(nElm), fStr, nElm);

}

**public** **void** updateElement(Row row, **int** nElm){

**this**.updateElement(row.getKey(), row.getFuncField(), nElm);

}

// вибірка за лінійним пошуком по ключу

**public** Row selectByLin(Long key){

Row row = **new** Row();

**for** (**int** i=0; i<keys.size(); i++){

**if** (keys.get(i)==key){

row.setKey(key);

row.setFuncField(fStrs.get(i));

**return** row;

}

}

**return** row;

}

// вибірка за лінійним пошуком по функціональному полю

**public** Row[] selectByLin(EnumField day){

ArrayList<Row> tempRows = **new** ArrayList<Row>();

**for** (**int** i=0; i<fStrs.size(); i++){

**if** (fStrs.get(i)==day){

tempRows.add(**new** Row(keys.get(i),day));

}

}

Row[] rows = **new** Row[tempRows.size()];

**for** (**int** i=0; i<rows.length; i++){

rows[i] = tempRows.get(i);

}

**return** rows;

}

//сортування таблиці по ключу

**public** **void** sort(){

**for** (**int** i=0; i < (keys.size()-1); i++){

**int** minIndex = i;

**for** (**int** j=i; j<keys.size(); j++){

**if** (keys.get(j) < keys.get(minIndex)){

minIndex = j;

}

}

**if** (minIndex != i){

Long minItem = keys.get(minIndex);

keys.set(minIndex, keys.get(i));

keys.set(i, minItem);

}

}

}

// вибірка за двійковим пошуком по ключу

**public** Row selectKeyByBin(Long key){

sort();

Row row = **new** Row();

**if** (keys.size() == 0)

{

**return** row;

}

**if** (keys.get(0) > key)

{

**return** row;

}

**if** (keys.get(keys.size()-1) < key)

{

**return** row;

}

**int** leftIndex = 0;

**int** rightIndex = keys.size();

**int** mid;

**while** (leftIndex < rightIndex)

{

mid = leftIndex + (rightIndex - leftIndex) / 2;

**if** (key <= keys.get(mid))

{

rightIndex = mid;

}

**else**

{

leftIndex = mid + 1;

}

}

**if** (keys.get(rightIndex) == key){

row.setKey(key);

row.setFuncField(fStrs.get(rightIndex));

**return** row;

}**else**{

**return** row;

}

}

//----- some other importent methods -----//

//is key in this table ?

**private** **boolean** isKey(Long key){

**for** (**int** i=0; i<keys.size(); i++){

**if** (keys.get(i).equals(key))

**return** **true**;

}

**return** **false**;

}

// print table to console

**public** **void** printTable(){

**for** (**int** i=0; i<columns.size();i++){

System.*out*.print(columns.get(i)+" | ");

}

System.*out*.println();

System.*out*.println("---------------------");

**for** (**int** i=0; i<keys.size();i++){

System.*out*.print(" "+keys.get(i)+" | "+fStrs.get(i));

System.*out*.println();

}

System.*out*.println("---------------------");

}

}

**EnumField**

**package** lab1\_;

**public** **enum** EnumField {

*SUNDAY*, *MONDAY*, *TUESDAY*, *WEDNESDAY*,*THURSDAY*, *FRIDAY*, *SATURDAY*

}

**TableFormatException**

**package** lab1\_;

**public** **class** TableFormatException **extends** Exception{

TableFormatException(){

System.*out*.println("TableFormatException");

}

TableFormatException (String massage){

**super**(massage);

}

}