Описати масив, який зберігає інформацію про телефони (фірма, рік випуску, об’єм оперативної пам»яті, розмір екрана. Організувати пошук за довільним запитом.

// ConsoleApplication65.cpp : Defines the entry point for the console application.

//

#include "stdafx.h"

#include<iostream>

#include<string>

using namespace std;

//Опис структури, що описує телефон

struct Phone

{

string firm;

int year;

int ram;

double screenSize;

};

//Функція для введення інформації про один телефон (заповнюємо поля однієї структури)

void inputPhoneInfo(Phone& phone)

{

cout << "Firm :";

cin >> phone.firm;

cout << "Year :";

cin >> phone.year;

cout << "RAM :";

cin >> phone.ram;

cout << "Size :";

cin >> phone.screenSize;

}

//Функція для виведення інформації про один телефон (вивдення полів структури)

void printPhoneInfo(const Phone& phone)

{

printf("%s, %d, %f, %d\n", phone.firm.data(), phone.ram, phone.screenSize, phone.year);

}

//Функція для створення масиву телефонів (структур)

Phone\* createPhonesArray(int phonesCount)

{

Phone\* phones = new Phone[phonesCount];

for (int i = 0; i < phonesCount; i++)

{

printf("----------------------\n");

inputPhoneInfo(phones[i]);

}

return phones;

}

//==================

bool ruleByYear(const Phone& phone, int year)

{

return phone.year == year;

}

bool ruleByRAM(const Phone& phone, int ram)

{

return phone.ram == ram;

}

//-----------------------

void printByRule(Phone\* phones, int phonesCount, int searchValue, bool (\*rule)(const Phone& , int))

{

bool contains = false;

for (int i = 0; i < phonesCount; i++)

{

if (rule(phones[i], searchValue) )

{

printPhoneInfo(phones[i]);

contains = true;

}

}

if (!contains)

{

printf("No phones");

}

}

//--------------------------------------------

void printByTemplate(Phone\* phones, int phonesCount, Phone& \_template)

{

bool contains = false;

for (int i = 0; i < phonesCount; i++)

{

if ((\_template.firm=="-" || \_template.firm==phones[i].firm) &&

(\_template.ram==0 || \_template.ram==phones[i].ram) &&

(\_template.screenSize == 0 || \_template.screenSize== phones[i].screenSize)&&

(\_template.year == 0 || \_template.year== phones[i].year))

{

printPhoneInfo(phones[i]);

contains = true;

}

}

if (!contains)

{

printf("No phones");

}

}

int main()

{

int phonesCount;

cout << "phones count : ";

cin >> phonesCount;

printf("Input phones \n");

Phone\* phones = createPhonesArray(phonesCount);

printf("1. Seach by year. \n 2.Search by RAM.\n 3.By template \n Your choise:");

int c;

cin >> c;

switch (c)

{

case 1:printByRule(phones, phonesCount, 2000, ruleByYear);

break;

case 2:printByRule(phones, phonesCount, 512, ruleByRAM);

break;

case 3:

printf("Imput search template(Firm or -, year of 0, ...)\n");

Phone temp;

inputPhoneInfo(temp);

printByTemplate(phones, phonesCount, temp);

break;

}

system("pause");

return 0;

}

// ConsoleApplication65.cpp : Defines the entry point for the console application.

//

#include "stdafx.h"

#include<iostream>

#include<string>

using namespace std;

//Опис структури, що описує телефон

struct Phone

{

string firm;

int year;

int ram;

double screenSize;

};

//Функція для введення інформації про один телефон (заповнюємо поля однієї структури)

void inputPhoneInfo(Phone& phone)

{

cout << "Firm :";

cin >> phone.firm;

cout << "Year :";

cin >> phone.year;

cout << "RAM :";

cin >> phone.ram;

cout << "Size :";

cin >> phone.screenSize;

}

//Функція для виведення інформації про один телефон (вивдення полів структури)

void printPhoneInfo(const Phone& phone)

{

printf("%s, %d, %f, %d\n", phone.firm.data(), phone.ram, phone.screenSize, phone.year);

}

//Функція для створення масиву телефонів (структур)

Phone\* createPhonesArray(int phonesCount)

{

Phone\* phones = new Phone[phonesCount];

for (int i = 0; i < phonesCount; i++)

{

printf("----------------------\n");

inputPhoneInfo(phones[i]);

}

return phones;

}

//==================

template<typename T>

bool ruleByFieldValue(const Phone& phone, string fieldName, T value)

{

if (fieldName=="firm")

{

return phone.firm == value;

}

if (fieldName=="year")

{

return phone.year == value;

}

if (fieldName == "RAM")

{

return phone.ram == value;

}

return phone.screenSize== value;

}

//====================================

//Пошук за назвою поля і значенням поля

template<typename T>

void printByFieldValue(Phone\* phones, int phonesCount, string fieldName,T value)

{

bool contains = false;

for (int i = 0; i < phonesCount; i++)

{

if (ruleByFieldValue(phones[i], fieldName,value))

{

printPhoneInfo(phones[i]);

contains = true;

}

}

if (!contains)

{

printf("No phones");

}

}

//-----------------------

bool ruleByYear(const Phone& phone, int year)

{

return phone.year == year;

}

bool ruleByRAM(const Phone& phone, int ram)

{

return phone.ram == ram;

}

void printByRule(Phone\* phones, int phonesCount, int searchValue, bool (\*rule)(const Phone& , int))

{

bool contains = false;

for (int i = 0; i < phonesCount; i++)

{

if (rule(phones[i], searchValue) )

{

printPhoneInfo(phones[i]);

contains = true;

}

}

if (!contains)

{

printf("No phones");

}

}

//--------------------------------------------

//Виведення за шаблоном

void printByTemplate(Phone\* phones, int phonesCount, Phone& \_template)

{

bool contains = false;

for (int i = 0; i < phonesCount; i++)

{

if ((\_template.firm=="-" || \_template.firm == phones[i].firm) &&

(\_template.ram==0 || \_template.ram==phones[i].ram) &&

(\_template.screenSize == 0 || \_template.screenSize== phones[i].screenSize)&&

(\_template.year == 0 || \_template.year== phones[i].year))

{

printPhoneInfo(phones[i]);

contains = true;

}

}

if (!contains)

{

printf("No phones");

}

}

//----------------------------------------

int main()

{

int phonesCount;

cout << "phones count : ";

cin >> phonesCount;

printf("Input phones \n");

Phone\* phones = createPhonesArray(phonesCount);

printf("1. Seach by year. \n 2.Search by RAM.\n 3.By template \n 4.By field Your choise:");

int c;

cin >> c;

switch (c)

{

case 1:printByRule(phones, phonesCount, 2000, ruleByYear);

break;

case 2:printByRule(phones, phonesCount, 512, ruleByRAM);

break;

case 3:

{

printf("Imput search template(Firm or -, year of 0, ...)\n");

Phone temp;

inputPhoneInfo(temp);

printByTemplate(phones, phonesCount, temp);

}

break;

case 4:

printByFieldValue(phones, phonesCount, "year", 1995);

printByFieldValue(phones, phonesCount, "size", 3.5);

printByFieldValue(phones, phonesCount, "firm", "Nokia");

break;

}

system("pause");

return 0;

}