

Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра автоматизованих систем обробки інформації і управління

Звіт

з лабораторної роботи №1
з дисципліни «Основи програмування 2. Модульне програмування»
«Файли даних.
Бінарні файли»

Виконав
студент

ІП-15, Волинець Кирило Михайлович

(шифр, прізвище, ім'я, по батькові)

Перевірів

Вечерковська Анастасія Сергіївна

(прізвище, ім'я, по батькові)

Бінарні файли

Мета - вивчити особливості створення і обробки бінарних файлів даних.

Завдання

Варіант 6

Створити файл з інформацією про товари меблевого магазину: назва товару, його вид (наприклад, стілець кухонний, стілець барний, стілець офісний, стіл кухонний, стіл комп'ютерний тощо), колір, ціна та наявна кількість. В новому бінарному файлі сформуванати список усіх наявних у магазині стільців вказаного виду, їх кількість та вартість. Видалити з нового файлу інформацію про стільці вартістю від 300 до 500 грн.

Розв'язання

C++:

```

1  #include <iostream>
2  #include <fstream>
3  #include <vector>
4  #include <string>
5  #include <Windows.h>
6  #include "header.h"
7
8  using namespace std;
9
10
11 int main()
12 {
13     fstream base_file = fileCreateEdit("basefile.dat");
14     base_file.close();
15     base_file.open("basefile.dat", ios::in, ios::binary);
16     if (fileIsEmpty(&base_file)) {
17         exit(1);
18     }
19
20     vector<product> database = readFile(&base_file);
21     cout << "Current database:" << endl;
22     print(database);
23
24     cout << endl << "Enter type filter:" << endl;
25     string type_filter;
26     cin >> type_filter;
27     vector<product> filtered_database = filterByType(database, type_filter);
28     cout << endl << "Filtered database: " << endl;
29     print(filtered_database);
30     fstream filtered_file("filteredfile.dat", ios::out, ios::binary);
31     writeFile(&filtered_file, filtered_database);
32     filtered_file.close();
33
34     filtered_file.open("filteredfile.dat", ios::in);
35     vector<product> filteredfile_database = readFile(&filtered_file);
36     filtered_file.close();

```

```

33
34     filtered_file.open("filteredfile.dat", ios::in);
35     vector<product> filteredfile_database = readFile(&filtered_file);
36     filtered_file.close();
37
38     filtered_file.open("filteredfile.dat", ios::out);
39     cout << "Enter price deleting range" << endl;
40     float min_filter, max_filter;
41     cin >> min_filter >> max_filter;
42     filteredfile_database = deleteByPriceRange(filteredfile_database, min_filter, max_filter);
43     writeFile(&filtered_file, filteredfile_database);
44     cout << endl << "Current filteredfile database:" << endl;
45     print(filteredfile_database);
46
47     base_file.close();
48     filtered_file.close();
49
50     system("pause");
51 }

```

```

1     #pragma once
2     #include <string>
3     using namespace std;
4
5     struct product {
6         string name, type, color;
7         float price;
8         int quantity;
9     };
10
11     fstream fileCreateEdit(string filename);
12     vector<product> filterByType(vector<product> database, string filter);
13     vector<product> deleteByPriceRange(vector<product> database, float min, float max);
14
15     void print(vector<product> database);
16
17     bool checkForFile(string name);
18     bool fileIsEmpty(fstream* file);
19
20     vector<product> readFile(fstream* file);
21     void writeFile(fstream* file, vector<product> database);
22
23     product readProduct(fstream* file);
24     void writeProduct(fstream* file, product item);

```

```

1  #include <fstream>
2  #include <string>
3  #include <iostream>
4  #include <vector>
5  #include "header.h"
6
7  using namespace std;
8
9  void cinToFstream(fstream* file);
10
11 fstream fileCreateEdit(string filename) {
12     string choose;
13     fstream file;
14     if (checkForFile(filename)) {
15         cout << filename << " is already found. Clear it? (y/n)" << endl;
16         cin >> choose;
17         cin.ignore();
18         if (choose == "y" or choose == "Y" or choose == "+") file.open(filename, ios::out, ios::binary);
19         else file.open(filename, ios::app, ios::binary);
20     }
21     else file.open(filename, ios::out, ios::binary);
22     cout << "Enter " << filename << " products (CTRL+X and enter in name input - end of input):" << endl;
23     cinToFstream(&file);
24     return file;
25 }
26
27 vector<product> filterByType(vector<product> database, string filter) {
28     for (size_t i = 0; i < database.size(); i++)
29     {
30         if (database[i].type != filter) database.erase(database.begin() + i);
31     }
32     return database;
33 }
34

```

```

26
27 vector<product> filterByType(vector<product> database, string filter) {
28     for (size_t i = 0; i < database.size(); i++)
29     {
30         if (database[i].type != filter) database.erase(database.begin() + i);
31     }
32     return database;
33 }
34
35 vector<product> deleteByPriceRange(vector<product> database, float min, float max) {
36     for (size_t i = 0; i < database.size(); i++)
37     {
38         if (database[i].price >= min and database[i].price <= max) database.erase(database.begin() + i);
39     }
40     return database;
41 }
42
43 void cinToFstream(fstream* file) {
44     product input;
45     while (true) {
46         cout << "Enter name of product:" << endl;
47         getline(cin, input.name);
48         if (input.name[0] == 24) break;
49         cout << "Enter product type:" << endl;
50         getline(cin, input.type);
51         cout << "Enter product color:" << endl;
52         getline(cin, input.color);
53         cout << "Enter product price:" << endl;
54         cin >> input.price;
55         cout << "Enter product quantity:" << endl;
56         cin >> input.quantity;
57         cin.ignore();
58         writeProduct(file, input);
59         cout << input.name << " was added to file" << endl;
60     }
61 }

```

```

62
63 void print(vector<product> database) {
64     for (size_t i = 0; i < database.size(); i++)
65     {
66         cout << "Product name - " << database[i].name << endl;
67         cout << "Type - " << database[i].type << endl;
68         cout << "Color - " << database[i].color << endl;
69         cout << "Price - " << database[i].price << endl;
70         cout << "Quantity - " << database[i].quantity << endl;
71         cout << endl;
72     }
73 }

```

```

1  #include <fstream>
2  #include <string>
3  #include <iostream>
4  #include <vector>
5  #include "header.h"
6
7  using namespace std;
8
9  string readString(fstream* file);
10 void writeString(fstream* file, string str);
11
12 int getFileSize(fstream* file);
13
14 vector<product> readFile(fstream* file) {
15     vector<product> result;
16     product cell;
17     int file_size = getFileSize(file);
18     while (file->tellg() < file_size) {
19         result.push_back(readProduct(file));
20     }
21     return result;
22 }
23
24 void writeFile(fstream* file, vector<product> database) {
25     for (size_t i = 0; i < database.size(); i++)
26     {
27         writeProduct(file, database[i]);
28     }
29 }
30

```

```

30
31  product readProduct(fstream* file) {
32      product result;
33      result.name = readString(file);
34      result.type = readString(file);
35      result.color = readString(file);
36      file->read((char*)&result.price, sizeof(result.price));
37      file->read((char*)&result.quantity, sizeof(result.quantity));
38      return result;
39  }
40
41  void writeProduct(fstream* file, product item) {
42      writeString(file, item.name);
43      writeString(file, item.type);
44      writeString(file, item.color);
45      file->write((char*)&item.price, sizeof(item.price));
46      file->write((char*)&item.quantity, sizeof(item.quantity));
47  }
48
49  string readString(fstream* file) {
50      int size;
51      file->read((char*)&size, sizeof(size));
52      char* buf = new char[size + 1];
53      buf[size] = '\0';
54      file->read(buf, size);
55      string str = buf;
56      delete[] buf;
57      return str;
58  }
59
60  void writeString(fstream* file, string str) {
61      int size = str.size();
62      file->write((char*)&size, sizeof(size));
63      file->write(str.c_str(), size);
64  }
65

```

```
66
67     bool fileIsEmpty(fstream* file) {
68         string temp;
69         *file >> temp;
70         if (temp == "") return true;
71     else {
72         file->seekg(0);
73         return false;
74     }
75 }
76
77     bool checkForFile(string name) {
78         ifstream file(name);
79     if (file) {
80         file.close();
81         return true;
82     }
83     else {
84         file.close();
85         return false;
86     }
87 }
88
89     int getFileSize(fstream* file) {
90         int current_ptr = file->tellg();
91         file->seekg(0, ios::end);
92         int size = file->tellg();
93         file->seekg(current_ptr, ios::beg);
94         return size;
95     }
```

C:\Users\kiril\source\repos\Lab1cpp_b\x64\Release\Lab1cpp_b.exe

```
basefile.dat is already found. Clear it? (y/n)
n
Enter basefile.dat products (CTRL+X and enter in name input - end of input):
Enter name of product:
Expensive chair
Enter product type:
Chair
Enter product color:
White
Enter product price:
4000
Enter product quantity:
5
Expensive chair was added to file
Enter name of product:
Default table
Enter product type:
Table
Enter product color:
Brown
Enter product price:
500
Enter product quantity:
10
Default table was added to file
Enter name of product:
^X
Current database:
Product name - Default chair
Type - Chair
Color - Black
Price - 300
Quantity - 35

Product name - Expensive chair
Type - Chair
Color - White
Price - 4000
Quantity - 5

Product name - Default table
Type - Table
Color - Brown
Price - 500
Quantity - 10

Enter type filter:
```


Enter type filter:

Chair

Filtered database:

Product name - Default chair

Type - Chair

Color - Black

Price - 300

Quantity - 35

Product name - Expensive chair

Type - Chair

Color - White

Price - 4000

Quantity - 5

Enter price deleting range

200 400

Current filteredfile database:


Product name - Expensive chair

Type - Chair

Color - White

Price - 4000

Quantity - 5

Для продолжения нажмите любую клавишу . . . 

python:

```

import os.path
import pickle

import fileFunctions as ff
import otherFunctions as of

def inputToFile(file):
    product = of.productDict()
    while True:
        product["name"] = str(input("Enter name of product\n"))
        if ord(product["name"][0]) == 24:
            break
        product["type"] = str(input("Enter product type\n"))
        product["color"] = str(input("Enter product color\n"))
        product["price"] = float(input("Enter product price\n"))
        product["quantity"] = int(input("Enter product quantity\n"))
        pickle.dump(product, file)
        print(product["name"]+" was added to file")

def fileCreateEdit(filename):
    if os.path.isfile(filename):
        print(filename + " is already found. Clear it? (y/n)")
        choose = input()
        if choose == "y" or choose == "Y" or choose == "+" : file = open(filename, 'wb')
        else: file = open(filename, 'ab')
    else: file = open(filename, 'wb')
    print("Enter "+filename+" text (CTRL+X and enter - end of file):")
    inputToFile(file)
    return file

base_file = fileCreateEdit("basefile_py.dat")
base_file.close()
base_file = open("basefile_py.dat", 'rb')
database = ff.readFile(base_file)
print("Current database:\n")
of.printDB(database)

type_filter = str(input("\nEnter type filter:\n"))
filtered_database = of.keyFilter(database, "type", type_filter)
print("\nFiltered database: ")
of.printDB(filtered_database)
filtered_file = open("filteredfile_py.dat", 'wb')
ff.writeFile(filtered_file, filtered_database)
filtered_file.close()

filtered_file = open("filteredfile_py.dat", 'rb')

filtered_file = open("filteredfile_py.dat", 'rb')
filteredfile_database = ff.readFile(filtered_file)
filtered_file.close()

filtered_file = open("filteredfile_py.dat", 'wb')
print("Enter price deleting range")
min_filter = float(input("Min value:\n"))
max_filter = float(input("Max value:\n"))
filteredfile_database = of.deleteByKeyRange(filteredfile_database, "price", min_filter, max_filter)
ff.writeFile(filtered_file, filteredfile_database)
print("\nCurrent filteredfile database:")
of.printDB(filteredfile_database)

base_file.close()
filtered_file.close()
input("Enter to close")

```

fileFunctions.py - C:\Users\kiril\Desktop\Ппора\Course1_s2\Lab1b\

File Edit Format Run Options Window Help

```
import pickle

def writeFile(file, database):
    for i in range(len(database)):
        pickle.dump(database[i], file)

def readFile(file):
    database = []
    file.seek(0,2)
    file_size = file.tell()
    file.seek(0,0)
    while file.tell() < file_size:
        database.append(pickle.load(file))
    return database
```

```
def productDict():
    product = {"name": "",
               "type": "",
               "color": "",
               "price": 0.0,
               "quantity": 0}
    return product

def printProduct(product):
    print("Product name - " + product["name"])
    print("Type - " + product["type"])
    print("Color - " + product["color"])
    print("Price - " + str(product["price"]))
    print("Quantity - " + str(product["quantity"]) + '\n')

def printDB(database):
    for i in range(len(database)):
        printProduct(database[i])

def keyFilter(database, key, key_filter):
    i = 0
    while i < len(database):
        if database[i][key] != key_filter:
            database.pop(i)
            i-=1
        i+=1
    return database

def deleteByKeyRange(database, key, min_key, max_key):
    i = 0
    while i < len(database):
        if database[i][key] >= min_key and database[i][key] <= max_key:
            database.pop(i)
            i-=1
        i+=1
    return database
```

C:\Windows\py.exe

basefile_py.dat is already found. Clear it? (y/n)

n

Enter basefile_py.dat text (CTRL+X and enter - end of file):

Enter name of product

Дорогий стілець

Enter product type

Стілець

Enter product color

Білий

Enter product price

2300

Enter product quantity

4

Дорогий стілець was added to file

Enter name of product

Звичайний стіл

Enter product type

Стіл

Enter product color

Коричневий

Enter product price

700

Enter product quantity

12

Звичайний стіл was added to file

Enter name of product

^X

Current database:

Product name - Звичайний стілець

Type - Стілець

Color - Чорний

Price - 300.0

Quantity - 23

Product name - Дорогий стілець

Type - Стілець

Color - Білий

Price - 2300.0

Quantity - 4

Product name - Звичайний стіл

Type - Стіл

Color - Коричневий

Price - 700.0

Quantity - 12

Enter type filter:

Enter type filter:

Стілець

Filtered database:

Product name - Звичайний стілець

Type - Стілець

Color - Чорний

Price - 300.0

Quantity - 23

Product name - Дорогий стілець

Type - Стілець

Color - Білий

Price - 2300.0

Quantity - 4

Enter price deleting range

Min value:

200

Max value:

400

Current filteredfile database:

Product name - Дорогий стілець

Type - Стілець

Color - Білий

Price - 2300.0

Quantity - 4

Enter to close_