**C++ Workshop – 150018**

**Homework Assignment #11**

**Files**

**Before you begin, make sure that your program follows the following rules:**

1. Your program should be easy to read. This means to make sure that you are using correct indentation and blank lines to make your code more readable
2. Break your program down into smaller sub-problems and use functions to solve each sub-problem.
3. Document your code as you were taught in the lecture. You are to write an abstract at the beginning of the program, comments before each method and function, and comment complicated sections of code.
4. Make sure that you understand the assignment and write a program that does exactly what the assignment requires.
5. Add an example of the output of the program in comments at the end of your program.
6. All solutions must be submitted in to Moodle according to the date that your instructor gave you which may not be the date listed in Moodle. You are given exactly one week to complete the assignment. You may work in teams of two. If you choose to do so, then one student must upload the program. The comments on the program must contain the names of both students. The other student must upload a Word document that lists the names of both students who worked on the program.

**Question 1**

In this assignment we will perform random accesses to binary files. On part of the data we will need to perform bitwise operations and we will also need exception handling for bad data.

**About the exceptions:**

In this exercise you are required to throw exceptions using the exception class and print the exception type using the inbuilt message (rather than throwing a constant string as was done in previous exercises).

**About the file:**

The first thing that the program will do is create a binary file for 100 records. The binary file (heretofor called the database) contains 100 identical sized records, each with a unique key (numbered from 1 to 100) corresponding to the record’s location in the file (the numbering is from 1 to 100 so that the record whose key is 1 is found in the beginning of the file).

Initially all records are empty, and during execution direct access to a record is done using its key. Since the entries are identical in size, the location of each entry in the file can be computed from the given key. It can be assumed that if the key stored in a record is 0, the record has not yet been created. Conversely, if the key has a value other than 0, then the record has already been created and contains actual information.(In other words, when we say that a family exists in the system then we mean that the family exists in the file. Aka the family’s id is present in the place where it should be and there is not a 0 in that place.).

**About the data:**

In the neighborhood a family can enroll its children to the following activities: swimming, gymnastics, dance, art, self-defence, music, drama and basketball.

Information about the activities will be saved in a byte which will be manipulated using bitwise arithmetic. Every activity has a corresponding bit that is “on” (1) if a family is enrolled in the activity and “off” (0) if not, as described in the following table (bit numbers are from least significant to most significant):

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Activity | swimming | gymnastics | dance | art | self-defence | music | drama | basketball |
| Bit no. | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

For example:

The number 1 in binary is 00000001 which signifies enrollment in the swimming activity.

The number 2 in binary is 00000010 which signifies enrollment in the gymnastics activity.

The number 68 in binary is 01000100 which signifies enrollment in both the dance activity and the drama activity.

For simplicity, we define the following enum:

enum ACTIVITY {

NONE,

SWIMMING,

GYMNATSTICS,

DANCE = 4,

ART = 8,

SELF\_DEFENSE = 16,

MUSIC = 32,

DRAMA = 64,

BASKETBALL = 128

};

Write a program that will manage the enrollment of family members in activities offered in the neighborhood they live.

1. Define a class **Family** to store the information of each family:
   1. family id (int) – in the range 1-100 (this is the key of the record)
   2. last name (up to 20 chars)
   3. number of family members
   4. phone number (int)
   5. The activities that the family enrolled for. They are to be stored in a **single short int** using bitwise manipulations as will be described below.
2. Write a function **setFile** that receives a reference to a binary file and initializes the file to store the data of 100 families. To do this, 100 blank family records should be placed in the file with the following data:
   1. family id: 0
   2. last name: an empty 20 length character string
   3. number of family members: 0
   4. phone number: 0
   5. activities: NONE (signifying no enrollment to any activity)
3. Write the following functions for maintaining the database:
   1. Function **add** which adds a family. The function receives as input a reference to a binary file stream. It then reads data for a family from standard input (console). Initially no children in the family are registered for any of the activities so the activities field should be initialized to NONE. If the family id given is legal but has so far not been registered in the database (i.e., the corresponding record has a family id of 0) a new family record is added to the database with the information that was read in.  
        
      If the family already exists in the database, then the exception ERROR: Family is already in the file should be thrown. If the family id is not valid (1-100), then the exception ERROR: Invalid family number should be thrown.
   2. Function **del** which deletes a family from the database. The function receives two parameters, a reference to a binary file stream and a family id and deletes the corresponding record from the file whose id matches the given parameter. (Note – the implication of deleting an entry is to reset its family id to 0. There is no need to change other fields).  
      If the family id is not valid (1-100), then the exception ERROR: Invalid family number should be thrown.

In the even that the family doesn’t exist in the system (in the file) then the exception:

ERROR: Family is not in the file should be thrown.

* 1. Helper function **count**. The function receives a reference to a binary file stream and an instance of ACTIVITY that represents a specific activity.

(Note the formal parameter (the parameter that is found in the function header) will hold the activity and should be of type int. The actual parameter (the parameter that is being passed to the function) should be one of the values of type ACTIVITIES (one of the enum values)). The function should iterate over the families in the file, count how many are enrolled to the given activity (that was passed as a parameter) and return that amount. This function will be called by the next function.  
The implementation of this function must be done using only bitwise operators.  
Hint: A family is enrolled to an activity if the result of & is not 0. For example: a family enrolled to dance and drama will have 68 in the activities field which in binary is 01000100. If we are checking if the family is enrolled to SWIMMING whose value 1 (00000001), the result of & is 0 which means the family is not enrolled. If we are checking if the family is enrolled to DANCE whose value is 4 (00000100), the result of & is 00000100 which is not 0 and means the family is enrolled as required.

* 1. Function **update** to update a family’s record data. The function receives three parameters: a reference to a binary file stream, a family id, and an STL queue.  
       
     If the family does not exist in the database, then the exception ERROR: Family is not in the file should be thrown. If the family id is not valid (1-100), then the exception ERROR: Invalid family number should be thrown.  
       
     If the input is valid, the function will ask for every activity if the family is interested in enrolling for that activity and will input Y or y if they are interested and N or n if not.  
     For any other response ERROR: Invalid response should be thrown.  
       
     The output for this stage will be:  
       
     Do you want to register for swimming?  
     Do you want to register for gymnastics?  
     Do you want to register for dance?  
     Do you want to register for art?  
     Do you want to register for self defense?  
     Do you want to register for music?  
     Do you want to register for drama?  
     Do you want to register for basketball?  
       
     Activities are limited to 10 children. Before performing the update, the program first checks that there is indeed room in a chosen activity (a call to the previous function would be prudent here). If there is space in the activity, the function updates the list of selected activities of the family. Note: A partial update can also be done; if a family is interested in 3 activities but only 2 have space then the family should be enrolled to those two.  
       
     The updated data about the family should then be written into the binary file.

If there is no space for a requested activity, the function should add the family to a waiting list. The waiting list is in the STL queue the was received as a parameter. (It is up to you how that queue is to be handled: you may store the family instance in the queue once with the activities field set to all the activities that are being waited for (similar to the one way that you store the family in the binary file) or you may store the family instance many times and in each time the activities field set to one activity that is being waited for).

* 1. Function **waiting** that handles waiting lists. The function receives the waiting queue and prints the family name and ‘phone number of the family followed by all the activities that the family is waiting for. The function will print (to the screen) the following details: last name, number of persons, telephone number and the names of all the activities the family is enrolled to.
  2. A function **print** for printing family information. The function receives as a parameter a reference to the binary file stream, and a second parameter that is a family id. If the family id exists in the data file, the function prints the family information to the standard output (screen) as follows: last name, number of persons, telephone number and the names of all the activities the family is enrolled to.  
       
     If the family does not exist in the database, then the exception ERROR: Family is not in the file should be thrown. If the family id is not valid (1-100), then the exception ERROR: Invalid family number should be thrown.

Here is the main program which tests the correctness of your functions:

#include "Family.h"

#include <queue>

#include <iostream>

#include <fstream>

#include <exception>

using namespace std;

enum { EXIT, ADD, DEL, COUNT, UPDATE, WAITING, PRINT };

enum ACTIVITY {

NONE, // טרם בחר חוג

SWIMMING, // שחיה

GYMNATSTICS, // התעמלות קרקע

DANCE = 4, // מחול

ART = 8, // אומנות

SELF\_DEFENSE = 16, // הגנה עצמית

MUSIC = 32, // מוסיקה

DRAMA = 64 // דרמה

BASKETBALL = 128; // כדור סל

};

void handleCount(fstream& file) {

int snum;

cout << "enter activity number to count:\n";

cout << "Choices are:\n1 SWIMMING\n2 GYMNASTICS\n3 DANCE\n4 ART\n5 SELF DEFENSE \n6 MUSIC \n7 DRAMA\n8 BASKETBALL\n";

cin >> snum;

cout << "there are ";

switch (snum) {

case 1:

cout << count(file, SWIMMING);

break;

case 2:

cout << count(file, GYMNASTICS);

break;

case 3:

cout << count(file, DANCE);

break;

case 4:

cout << count(file, ART);

break;

case 5:

cout << count(file, SELF\_DEFENSE);

break;

case 6:

cout << count(file, MUSIC);

break;

case 7:

cout << count(file, DRAMA);

break;

case 8:

cout << count(file, BASKETBALL);

break;

}

cout << " families registered\n";

}

int main()

{

queue<Family> q;

fstream file;

file.open("families.txt", ios::binary | ios::out);

if (!file)

{

cout << "ERROR: couldn't open file\n";

return 0;

}

setFile(file);

file.close();

file.open("families.txt", ios::binary | ios::in | ios::out);

int choice;

int snum;

int cnum;

cout << "Choices are:\n0 to exit\n1 to add a family\n2 to delete a family\n3 to count number of families signed up\n4 to update the activities of a family \n5 print the waiting list \n6 to print a family\n";

cout << "enter 0-6:\n";

cin >> choice;

while (choice)

{

try {

switch (choice)

{

case ADD://add to the file

add(file);

break;

case DEL://delete from file

cout << "enter number of family to delete:\n";

cin >> snum;

del(file, snum);

break;

case COUNT:

handleCount(file);

break;

case UPDATE://update the list of classes of a family

cout << "enter number of family to update:\n";

cin >> snum;

update(file, snum, q);

break;

case WAITING://update the list of classes for waiting fam

waiting(q);

break;

case PRINT://print the details of a specific family

cout << "enter number of family to print:\n";

cin >> snum;

print(file, snum);

break;

default:

cout << "ERROR: invalid choice\n";

}

}// try

catch (const char\* msg) {

cout << msg;

}

cout << "\nenter 0-6:\n";

cin >> choice;

}

file.close();

return 0;

}

דוגמה:

Choices are:

0 to exit

1 to add a family

2 to delete a family

3 to count number of families signed up

4 to update the activities of a family

5 print the waiting list

6 to print a family

enter 0-6:

1

enter the family's: id, name, amount of people and a phone number:

5 aaron 5 0505551235

enter 0-6:

1

enter the family's: id, name, amount of people and a phone number:

123 brian 7 0505551123

ERROR: Invalid family number

enter 0-6:

1

enter the family's: id, name, amount of people and a phone number:

5 aaron 5 0505551235

ERROR: Family is already in the file

enter 0-6:

1

enter the family's: id, name, amount of people and a phone number:

6 carly 8 0505551236

enter 0-6:

1

enter the family's: id, name, amount of people and a phone number:

7 daniella 9 0505551237

enter 0-6:

2

enter number of family to delete:

6

enter 0-6:

4

enter number of family to update:

6

ERROR: Family is not in the file

enter 0-6:

4

enter number of family to update:

5

Do you want to register for swimming?

Y

Do you want to register for gymnastics?

Y

Do you want to register for dance?

Y

Do you want to register for art?

Y

Do you want to register for self defense?

N

Do you want to register for music?

N

Do you want to register for drama?

N

Do you want to register basketball?

N

enter 0-6:

4

enter number of family to update:

7

Do you want to register for swimming?

Y

Do you want to register for gymnastics?

N

Do you want to register for dance?

Y

Do you want to register for art?

N

Do you want to register for self defense?

Y

Do you want to register for music?

N

Do you want to register for drama?

Y

Do you want to register basketball?

N

enter 0-6:

3

enter activity number to count:

Choices are:

1 SWIMMING

2 GYMNASTICS

3 DANCE

4 ART

5 SELF DEFENSE

6 MUSIC

7 DRAMA

8 BASKETBALL

1

there are 2 families registered

enter 0-6:

6

enter number of family to print:

5

family name: aaron

number of persons: 5

phone number: 505551235

!2!15

SWIMMING

GYMNASTICS

DANCE

ART

enter 0-6:

6

enter number of family to print:

6

ERROR: Family is not in the file

enter 0-6:

6

enter number of family to print:

7

family name: Daniella

number of persons: 9

phone number: 505551237

SWIMMING

DANCE

SELF DEFENSE

DRAMA

