

МIНIСТЕРСТВО  ОСВIТИ  І  НАУКИ  УКРАЇНИ

НАЦІОНАЛЬНИЙ   ТЕХНІЧНИЙ   УНІВЕРСИТЕТ   УКРАЇНИ

“КИЇВСЬКИЙ  ПОЛІТЕХНІЧНИЙ  ІНСТИТУТ”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**Лабораторна робота №** 1

з дисципліни “ Основи програмування ”

Тема: “Декомпозиція програми на модулі, створення інтерфейсів, обробка помилок та модульне тестування. ”

|  |  |  |
| --- | --- | --- |
| Виконав(ла)  студент(ка) I курсу  групи КП-52  Cаприкін Артем Олексійович  (*прізвище, ім’я, по батькові*)  варіант №4 |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладач  Гадиняк Руслан Анатолійович  (*прізвище, ім’я, по батькові*) |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Штрафні бали:   |  |  | | --- | --- | | **Термін здачі** | **Оформлення звіту** | |  |  | | Нараховані бали:   |  |  |  | | --- | --- | --- | | **Корект. програм (2 бала)** | **Відп. на теор. питання (1 бал)** | **Відп. на прогр. питання (2 бала)** | |  |  |  | | Сумарний бал:   |  | | --- | |  | |

Київ 2016

**Мета роботи**

Освоїти основні принципи модульного програмування в мові С, навчитися розділяти програму на модулі, виділяти їх інтерфейси та приховувати реалізацію. Вміти генерувати всередині функцій та модулів помилкові ситуації та обробляти їх у клієнтському коді. Забезпечувати якість розробленого програмного коду за допомогою покриття модульними тестами.

**Постановка завдання**

На основі персонального завдання за варіантом виконати планування його реалізації із розділенням програмного коду на окремі модулі (мінімум 2 модулі) та їх інтеграцію у робочу програму.

Описати інтерфейси кожного модуля у відповідності до вимог, що наводились у лекційному матеріалі.

Додати можливість визначення статусу виконання методів модулів для обробки помилок у клієнтському коді.

Реалізувати модулі, використовуючи інкапсуляцію внутрішніх функцій та типів даних модулів.

Написати набори модульних тестів для кожного розробленого модуля, що найкраще покривають всі функції та їх параметри. Назви модульних тестів описувати у відповідності до рекомендацій, що наводились у лекційному матеріалі.

**Завдання за варіантом:**

Система бронювання квитків у кіно. Є кінозал із обмеженою кількістю місць. Є багато користувачів, кожен із яких може забронювати або викупити квитки на сеанс. При запиті на бронь місця система блокує всім іншим користувачам доступ до місця.

**Тексти коду програм**

**Модуль cinemahall.**

|  |
| --- |
| cinemahall.h |
| #ifndef CINEMAHALL\_H\_INCLUDED  #define CINEMAHALL\_H\_INCLUDED  typedef struct tagCinemaHall\_Private CinemaHall\_Private;  #define SEAT\_AVAILABLE 0  #define SEAT\_BOUGHT 1  #define SEAT\_BOOKED 2  #define SEAT\_DENIED 3  typedef struct  {  CinemaHall\_Private\* chp;  } CinemaHall;  int CinemaHall\_initialize (CinemaHall\* , int);  int CinemaHall\_deinitialize (CinemaHall\* );  int CinemaHall\_getQuantity (CinemaHall\*);  int CinemaHall\_getSeatState (CinemaHall\*, int);  int CinemaHall\_setSeatState (CinemaHall\*, int, int);  char\* CinemaHall\_errorCode\_toString (int);  #endif // CINEMAHALL\_H\_INCLUDED |

|  |
| --- |
| cinemahall.c |
| #include "cinemahall.h"  #include "errormodule.h"  #include <stdlib.h>  struct tagCinemaHall\_Private  {  int\* cinemaArray;  int cinemaSize;  };  int CinemaHall\_initialize (CinemaHall\* ch , int size)  {  int status = 0;  CinemaHall\_Private\* chp = malloc (sizeof (CinemaHall\_Private));  if (chp == NULL)  {  status = 2;  ErrorModule\_setLastError(status);  return status;  }  ch->chp = chp;  if (size <= 0 || size > 32000)  {  status = 1;  ErrorModule\_setLastError(status);  size = 10;  }  int\* array = malloc (size\*sizeof(int));  if (array == NULL)  {  status = 2;  ErrorModule\_setLastError(status);  return status;  }  memset (array, 0, size\*sizeof(int));  ch->chp->cinemaArray = array;  ch->chp->cinemaSize = size;  ErrorModule\_setLastError(status);  return status;  }  int CinemaHall\_deinitialize (CinemaHall\* ch)  {  int status = 0;  free (ch->chp->cinemaArray);  free (ch->chp);  ErrorModule\_setLastError(status);  return status;  }  int CinemaHall\_getQuantity (CinemaHall\* ch)  {  int status = 0;  ErrorModule\_setLastError(status);  return ch->chp->cinemaSize;  }  int CinemaHall\_getSeatState (CinemaHall\* ch, int where)  {  int status = 0;  if (where < 0 || where >= ch->chp->cinemaSize)  {  status = 3;  ErrorModule\_setLastError(status);  return -1;  }  ErrorModule\_setLastError(status);  return ch->chp->cinemaArray[where];  }  int CinemaHall\_setSeatState (CinemaHall\* ch, int where, int state)  {  int status = 0;  if (where < 0 || where >= ch->chp->cinemaSize)  {  status = 3;  ErrorModule\_setLastError(status);  return -1;  }  int prev = ch->chp->cinemaArray[where];  ch->chp->cinemaArray[where] = state;  ErrorModule\_setLastError(status);  return prev;  }  char\* CinemaHall\_errorCode\_toString (int code)  {  const char\* errorStrings [] =  {  "All success",  "Invalid cinema size",  "Error allocating memory",  "Invalid place",  };  const int errorSize = sizeof (errorStrings) / sizeof (char\*);  return errorStrings [code % errorSize];  } |

**Модуль client.**

|  |
| --- |
| client.h |
| **#ifndef CLIENT\_H\_INCLUDED**  **#define CLIENT\_H\_INCLUDED**  **typedef struct tagClient\_Private Client\_Private;**  **typedef struct**  **{**  **Client\_Private\* cp;**  **} Client;**  **int Client\_initialize (Client\*, int);**  **int Client\_deinitialize (Client\*);**  **int Client\_getId (Client\*);**  **int Client\_getSeatState (Client\*);**  **int Client\_getMaxSeat (Client\*);**  **int Client\_getAction (Client\*, int\*, int\*);**  **char\* Client\_errorCode\_toString (int);**  **#endif // CLIENT\_H\_INCLUDED** |

|  |
| --- |
| client.c |
| **#include "client.h"**  **#include "cinemahall.h"**  **#include <stdlib.h>**  **struct tagClient\_Private**  **{**  **int maxSeat;**  **int id;**  **int seatNumber;**  **int seatState;**  **};**  **int Client\_getMaxSeat (Client\* c)**  **{**  **int status = 0;**  **ErrorModule\_setLastError(status);**  **return c->cp->maxSeat;**  **}**  **int Client\_initialize(Client\* c, int maxSeat)**  **{**  **static count = 0;**  **int status = 0;**  **Client\_Private\* cp = malloc (sizeof (Client\_Private));**  **if (cp == NULL)**  **{**  **status = 1;**  **ErrorModule\_setLastError(status);**  **return status;**  **}**  **c->cp = cp;**  **if (maxSeat <= 0 || maxSeat >= 32000)**  **{**  **status = 2;**  **maxSeat = 10;**  **ErrorModule\_setLastError(status);**  **}**  **c->cp->maxSeat = maxSeat;**  **c->cp->id = count++;**  **c->cp->seatNumber = -1;**  **c->cp->seatState = -1;**  **ErrorModule\_setLastError(status);**  **return status;**  **}**  **int Client\_getAction (Client\* c, int\* where, int\* state)**  **{**  **int status = 0;**  **int s = rand () % 2 + 1;**  **if (s == SEAT\_BOOKED)**  **s = rand () % 2 + 1;**  **\*state = s;**  **\*where = rand () % c->cp->maxSeat;**  **ErrorModule\_setLastError(status);**  **return status;**  **}**  **int Client\_getSeatState (Client\* c)**  **{**  **int status = 0;**  **ErrorModule\_setLastError(status);**  **return c->cp->seatState;**  **}**  **int Client\_deinitialize(Client\* c)**  **{**  **int status = 0;**  **free (c->cp);**  **ErrorModule\_setLastError(status);**  **return status;**  **}**  **int Client\_getId (Client\* c)**  **{**  **int status = 0;**  **ErrorModule\_setLastError(status);**  **return c->cp->id;**  **}**  **char\* Client\_errorCode\_toString(int code)**  **{**  **const char\* errorStrings [] =**  **{**  **"All success",**  **"Error allocating memory"**  **"Error Invalid max seat"**  **};**  **const int errorSize = sizeof (errorStrings) / sizeof (char\*);**  **return errorStrings [code % errorSize];**  **}** |

**Модуль Console.**

|  |
| --- |
| console.h |
| **#ifndef CONSOLE\_H\_INCLUDED**  **#define CONSOLE\_H\_INCLUDED**  **#include "cinemahall.h"**  **#include <windows.h>**  **#define FOREGROUND\_WHITE FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_RED**  **#define BACKGROUND\_WHITE BACKGROUND\_BLUE | BACKGROUND\_GREEN | BACKGROUND\_RED**  **#define BACKGROUND\_YELLOW BACKGROUND\_RED | BACKGROUND\_GREEN | BACKGROUND\_INTENSITY**  **#define BACKGROUND\_GRAY BACKGROUND\_INTENSITY**  **typedef struct tagConsole\_Private Console\_Private;**  **typedef struct**  **{**  **Console\_Private\* cp;**  **} Console;**  **int Console\_initialize (Console\*, int, int);**  **int Console\_deinitialize (Console\*);**  **int Console\_getSize (Console\*, COORD\*);**  **int Console\_showCinema (Console\*);**  **static int Console\_setPosition (Console\*, int, int);**  **static int Console\_printSymbol (Console\*, char, int);**  **int Console\_showCinemaHall (Console\*, CinemaHall\* );**  **char\* Console\_errorCode\_toString (int);**  **#endif // CONSOLE\_H\_INCLUDED** |

|  |
| --- |
| console.c |
| **#include "console.h"**  **#include "errormodule.h"**  **#include <windows.h>**  **struct tagConsole\_Private**  **{**  **HANDLE hConsole\_Output;**  **HANDLE hConsole\_Input;**  **int width;**  **int height;**  **};**  **int Console\_initialize (Console\* c, int width, int height)**  **{**  **int status = 0;**  **Console\_Private\* cp = malloc (sizeof (Console\_Private));**  **if (cp == NULL)**  **{**  **status = 1;**  **ErrorModule\_setLastError(status);**  **return status;**  **}**  **c->cp = cp;**  **c->cp->hConsole\_Output = GetStdHandle(STD\_OUTPUT\_HANDLE);**  **c->cp->hConsole\_Input = GetStdHandle (STD\_INPUT\_HANDLE);**  **c->cp->width = width;**  **c->cp->height = height;**  **char cmd[32];**  **sprintf(cmd, "mode con: lines=%d cols=%d", height, width);**  **system(cmd);**  **ErrorModule\_setLastError(status);**  **return status;**  **}**  **int Console\_deinitialize (Console\* c)**  **{**  **int status = 0;**  **CloseHandle (c->cp->hConsole\_Output);**  **CloseHandle(c->cp->hConsole\_Input);**  **ErrorModule\_setLastError(status);**  **return status;**  **}**  **int Console\_showCinema (Console\* c)**  **{**  **int status = 0;**  **const int n = 256;**  **char szFileName[n];**  **strcpy (szFileName, "cinema.bmp");**  **HBITMAP hBmp;**  **BITMAP bmp;**  **hBmp = (HBITMAP)LoadImage (NULL, szFileName, IMAGE\_BITMAP, NULL, NULL, LR\_LOADFROMFILE);**  **if (hBmp == NULL)**  **{**  **status = 2;**  **ErrorModule\_setLastError(status);**  **return status;**  **}**  **GetObject (hBmp, sizeof (BITMAP), &bmp);**  **int BitCount = bmp.bmPlanes\*bmp.bmBitsPixel;**  **int colorSize = 0;**  **switch (BitCount)**  **{**  **case 1:**  **case 4:**  **case 8:**  **case 32:**  **colorSize = sizeof (RGBQUAD)\* (1 << BitCount);**  **break;**  **case 16:**  **case 24:**  **colorSize = NULL;**  **break;**  **}**  **int dataSize = bmp.bmWidthBytes\*bmp.bmHeight;**  **void\* pvBits = malloc (dataSize);**  **BITMAPINFO\* pBmi = (BITMAPINFO\*)malloc(sizeof (BITMAPINFOHEADER) + colorSize);**  **pBmi->bmiHeader.biSize = sizeof (BITMAPINFOHEADER);**  **pBmi->bmiHeader.biWidth = bmp.bmWidth;**  **pBmi->bmiHeader.biHeight = -bmp.bmHeight;**  **pBmi->bmiHeader.biPlanes = 1;**  **pBmi->bmiHeader.biBitCount = BitCount;**  **pBmi->bmiHeader.biCompression = BI\_RGB;**  **pBmi->bmiHeader.biSizeImage = dataSize;**  **pBmi->bmiHeader.biXPelsPerMeter = pBmi->bmiHeader.biYPelsPerMeter = 0;**  **pBmi->bmiHeader.biClrImportant = 0;**  **HDC hDC = GetDC (0);**  **GetDIBits (hDC, hBmp, 0, bmp.bmHeight, pvBits, pBmi, DIB\_RGB\_COLORS);**  **ReleaseDC (0, hDC);**  **COORD pos;**  **for (int y = 0; y < bmp.bmHeight; y++)**  **{**  **for (int x = 0; x < bmp.bmWidth; x++)**  **{**  **BYTE\* pixel = (BYTE\*)(pvBits + 4\*y\*bmp.bmWidth + x\*4);**  **pos.X = x;**  **pos.Y = y;**  **//pixel[0] - BLUE pixel[1] - GREEN pixel[2] - RED pixel[3] - alpha**  **int attr = 0;**  **if (pixel[0] > 100)**  **attr |= BACKGROUND\_BLUE;**  **if (pixel[1] > 100)**  **attr |= BACKGROUND\_GREEN;**  **if (pixel[2] > 100)**  **attr |= BACKGROUND\_RED;**  **SetConsoleTextAttribute(c->cp->hConsole\_Output, attr);**  **SetConsoleCursorPosition(c->cp->hConsole\_Output, pos);**  **printf (" ");**  **}**  **}**  **ErrorModule\_setLastError(status);**  **return 0;**  **}**  **int Console\_getSize (Console\* c, COORD\* cd)**  **{**  **int status = 0;**  **COORD coord;**  **coord.X = c->cp->width;**  **coord.Y = c->cp->height;**  **memcpy(cd, &coord, sizeof (COORD));**  **ErrorModule\_setLastError(status);**  **return status;**  **}**  **int Console\_setPosition (Console\* c, int x, int y)**  **{**  **COORD pos;**  **pos.X = x;**  **pos.Y = y;**  **SetConsoleCursorPosition(c->cp->hConsole\_Output, pos);**  **return 0;**  **}**  **int Console\_printSymbol (Console\* c, char ch, int attr)**  **{**  **SetConsoleTextAttribute(c->cp->hConsole\_Output, attr);**  **printf (" ");**  **return 0;**  **}**  **int Console\_showCinemaHall (Console\* c, CinemaHall\* ch)**  **{**  **const int marginTop = 20;**  **const int marginLeft = 10;**  **const int marginRight = 10;**  **const int secondMarginLeft = 1;**  **const int secondMarginRight = 1;**  **int status = 0;**  **int dl = 0;**  **int dr = 0;**  **for (int i = 0; i < CinemaHall\_getQuantity (ch); i++)**  **{**  **if (i != 0 && i % (c->cp->width - marginLeft - marginRight - dl - dr) == 0)**  **dl+=secondMarginLeft, dr+=secondMarginRight;**  **int y = marginTop + (i / (c->cp->width - marginLeft - marginRight - dl - dr));**  **int x = marginLeft +**  **i % (c->cp->width - marginLeft - marginRight - dl - dr);**  **Console\_setPosition (c, x, y);**  **int attr = 0;**  **switch ( CinemaHall\_getSeatState(ch, i) )**  **{**  **case SEAT\_AVAILABLE:**  **attr = BACKGROUND\_GRAY;**  **break;**  **case SEAT\_BOUGHT:**  **attr = BACKGROUND\_RED;**  **break;**  **case SEAT\_BOOKED:**  **attr = BACKGROUND\_YELLOW;**  **break;**  **}**  **Console\_printSymbol (c, ' ', attr);**  **}**  **ErrorModule\_setLastError(status);**  **return status;**  **}**  **char\* Console\_errorCode\_toString (int code)**  **{**  **const char\* errorStrings [] =**  **{**  **"All success",**  **"Error allocating memory"**  **"Error no cinema picture"**  **};**  **const int errorSize = sizeof (errorStrings) / sizeof (char\*);**  **return errorStrings [code % errorSize];**  **}** |

**Модуль ErrorModule.**

|  |
| --- |
| errormodule.h |
| **#ifndef ERRORMODULE\_H\_INCLUDED**  **#define ERRORMODULE\_H\_INCLUDED**  **static int iErrorCode = 0;**  **int ErrorModule\_setLastError (int);**  **int ErrorModule\_getLastError ();**  **#endif // ERRORMODULE\_H\_INCLUDED** |

|  |
| --- |
| errormodule.c |
| **#include "errormodule.h"**  **int ErrorModule\_setLastError (int c)**  **{**  **iErrorCode = c;**  **return 0;**  **}**  **int ErrorModule\_getLastError ()**  **{**  **return iErrorCode;**  **}** |

**Модульні тести**

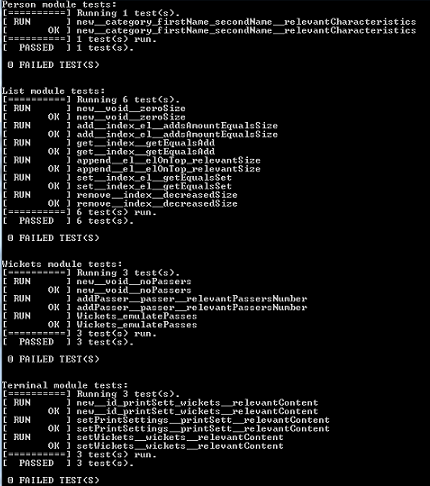
**Код тестів модуля CinemaHall.**

|  |
| --- |
| CinemaHall\_tests.c |
| **#include <stdlib.h>**  **#include <stdarg.h>**  **#include <stddef.h>**  **#include <setjmp.h>**  **#include <cmocka.h>**  **#include "tests.h"**  **#include "../cinemahall.h"**  **static void new\_\_int\_\_relevantSize (void\*\* state)**  **{**  **CinemaHall cinema;**  **CinemaHall\_initialize(&cinema, 20);**  **assert\_int\_equal (CinemaHall\_getQuantity(&cinema), 20);**  **CinemaHall\_deinitialize(&cinema);**  **}**  **static void new\_\_int\_\_invalidSize (void\*\* state)**  **{**  **CinemaHall cinema;**  **CinemaHall\_initialize(&cinema, -1);**  **assert\_int\_equal (CinemaHall\_getQuantity(&cinema), 10);**  **CinemaHall\_deinitialize(&cinema);**  **}**  **static void getQuantity\_\_void\_\_relevantSize (void\*\* state)**  **{**  **CinemaHall cinema;**  **CinemaHall\_initialize(&cinema, 5);**  **assert\_int\_equal (CinemaHall\_getQuantity(&cinema), 5);**  **CinemaHall\_deinitialize(&cinema);**  **}**  **static void getSeatState\_\_CinemaHall\_\_pos\_\_SeatState (void\*\* state)**  **{**  **CinemaHall cinema;**  **CinemaHall\_initialize(&cinema, 5);**  **CinemaHall\_setSeatState(&cinema, 3, SEAT\_BOOKED);**  **assert\_int\_equal (CinemaHall\_getSeatState(&cinema, 3), SEAT\_BOOKED);**  **CinemaHall\_deinitialize(&cinema);**  **}**  **static void setSeatState\_\_CinemaHall\_\_pos\_\_state\_\_PrevSeatState (void\*\* state)**  **{**  **CinemaHall cinema;**  **CinemaHall\_initialize(&cinema, 5);**  **CinemaHall\_setSeatState(&cinema, 3, SEAT\_BOUGHT);**  **assert\_int\_equal (CinemaHall\_setSeatState(&cinema, 3, SEAT\_BOOKED), SEAT\_BOUGHT);**  **CinemaHall\_deinitialize(&cinema);**  **}**  **void moduleTests\_CinemaHall(void) {**  **const UnitTest tests[] =**  **{**  **unit\_test(new\_\_int\_\_relevantSize),**  **unit\_test(new\_\_int\_\_invalidSize),**  **unit\_test(getQuantity\_\_void\_\_relevantSize),**  **unit\_test(getSeatState\_\_CinemaHall\_\_pos\_\_SeatState),**  **unit\_test(setSeatState\_\_CinemaHall\_\_pos\_\_state\_\_PrevSeatState)**  **};**  **return run\_tests(tests);**  **}** |

**Код тестів модуля client.**

|  |
| --- |
| client\_tests.c |
| **#include <stdlib.h>**  **#include <stdarg.h>**  **#include <stddef.h>**  **#include <setjmp.h>**  **#include <cmocka.h>**  **#include "tests.h"**  **#include "../client.h"**  **static void initialize\_\_int\_\_InvalidMaxSeat (void\*\* state)**  **{**  **Client client;**  **Client\_initialize(&client, -1);**  **assert\_int\_equal(Client\_getMaxSeat(&client), 10);**  **Client\_deinitialize(&client);**  **}**  **static void initialize\_\_int\_\_relevantMaxSeat (void\*\* state)**  **{**  **Client client;**  **Client\_initialize(&client, 5);**  **assert\_int\_equal(Client\_getMaxSeat(&client), 5);**  **Client\_deinitialize(&client);**  **}**  **static void getId\_\_void\_\_relevantId (void\*\* state)**  **{**  **Client client;**  **Client\_initialize(&client, 5);**  **assert\_int\_equal(Client\_getId(&client), 2);**  **Client\_deinitialize(&client);**  **}**  **void moduleTests\_Client(void) {**  **const UnitTest tests[] =**  **{**  **unit\_test (initialize\_\_int\_\_InvalidMaxSeat),**  **unit\_test (initialize\_\_int\_\_relevantMaxSeat),**  **unit\_test (getId\_\_void\_\_relevantId)**  **};**  **return run\_tests(tests);**  **}** |

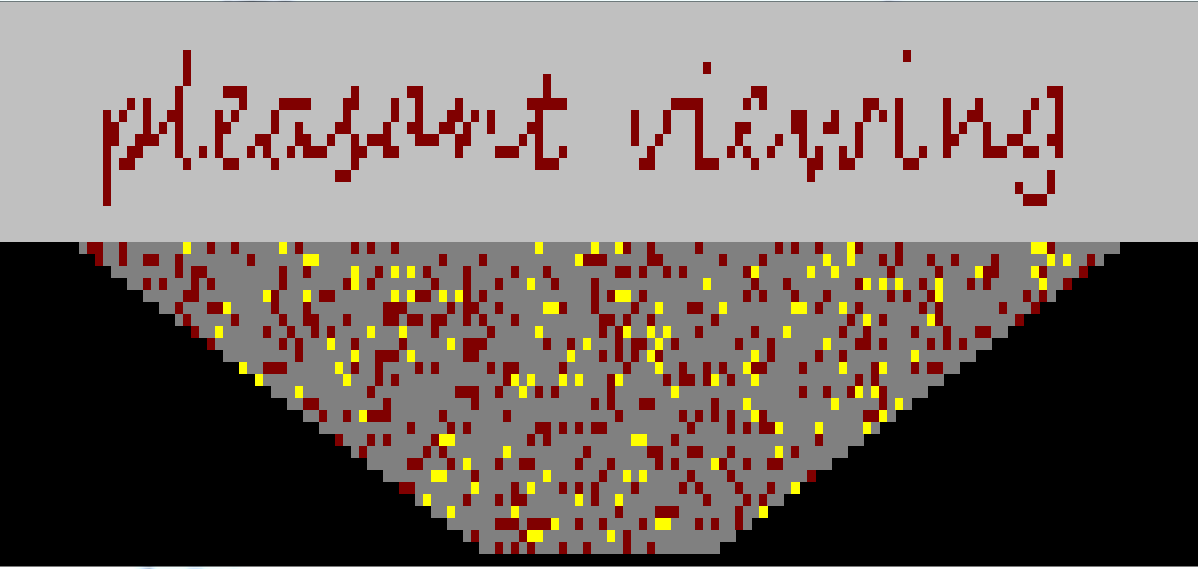
**Результати проведення тестів:**

****

**Приклад використання описаних модулів**

|  |
| --- |
| main.c |
| **#include <stdio.h>**  **#include <stdlib.h>**  **#include "errormodule.h"**  **#include "cinemahall.h"**  **#include "console.h"**  **#include "client.h"**  **void emulate\_cinema ()**  **{**  **srand (time (NULL));**  **CinemaHall cinema;**  **const int maxSeat = 2080;**  **CinemaHall\_initialize(&cinema, maxSeat);**  **FILE\* pLog = fopen ("log.txt", "w");**  **Console console;**  **Console\_initialize (&console, 150, 47);**  **Console\_showCinema(&console);**  **const int qClients = 600;**  **const int updateSpeed = 1;**  **Client clients[qClients];**  **for (int i = 0; i < qClients; i++)**  **Client\_initialize (&clients[i], maxSeat);**  **for (int i = 0; i < qClients; i++)**  **{**  **int seatNumber = -1, seatState = -1;**  **Client\_getAction(&clients[i], &seatNumber, &seatState);**  **if ( CinemaHall\_getSeatState(&cinema, seatNumber) == SEAT\_AVAILABLE )**  **{**  **if (seatState == SEAT\_BOUGHT)**  **{**  **fprintf (pLog, "Client %d has bought seat number: %d\n", i, seatNumber);**  **}**  **if (seatState == SEAT\_BOOKED)**  **{**  **fprintf (pLog, "Client %d has booked seat number: %d\n", i, seatNumber);**  **}**  **CinemaHall\_setSeatState(&cinema, seatNumber, seatState);**  **}**  **else**  **{**  **fprintf (pLog, "Client %d access to seat number %d denied\n", i, seatNumber);**  **}**  **//Sleep (100);**  **if (i % (qClients/20\*updateSpeed) == 0)**  **Console\_showCinemaHall(&console, &cinema);**  **}**  **for (int i = 0; i < qClients; i++)**  **Client\_deinitialize(&clients[i]);**  **fclose (pLog);**  **Console\_deinitialize(&console);**  **CinemaHall\_deinitialize(&cinema);**  **Sleep (INFINITE);**  **}**  **void test\_all ()**  **{**  **totalTests ();**  **}**  **int main()**  **{**  **//emulate\_cinema();**  **test\_all ();**  **return 0;**  **}** |

**Результат:**



**Висновки**

Виконавши дану лабораторну роботу, я освоїв основні принципи модульного програмування в мові С, навчився розділяти програму на модулі, виділяти їх інтерфейси та приховувати реалізацію. Навчився генерувати всередині функцій та модулів помилкові ситуації та обробляти їх у клієнтському коді, забезпечувати якість розробленого програмного коду за допомогою покриття модульними тестами.

Я отримав непогану практику розбиття функціоналу програми на модулі і зрозумів як використовувати такі принципи об’єктно-орієтнованого програмування, як абстракція та інкапсуляція реалізації, в мові С.