

МIНIСТЕРСТВО ОСВIТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**Лабораторна робота №** **3**

з дисципліни “ Основи програмування ”

тема “Функції зворотнього виклику в мові С”

|  |  |  |
| --- | --- | --- |
| Виконав  студент I курсу  групи КП-52  Сорочинський Владислав Володимирович  (*прізвище, ім’я, по батькові*)  варіант №16 |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 2016 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**

Навчитись використовувати функції зворотнього виклику у мові С для структурування коду. Реалізувати підтримку модулями подій на основі функцій зворотнього виклику для гнучкого налаштування поведінки модулів.

**Постановка завдання**

Відповідно до варіанту завдання реалізувати модуль та додати можливість встановлювати функції зворотнього виклику у конструкторах модуля (або у спеціальних методах) для реакції на події, що відбуваються всередині модуля.

Реалізувати як мінімум одну подію, яка буде виконувати набір callback функцій, що підписані на неї.

У якості користувачів можна описувати функції, які просто виводять певне повідомлення у відповідь на згенеровану подію.

Покрити один модуль модульними тестами.

Сховище даних SupaDrive дає можливість завантажувати файли (назва + випадковий масив байт), зберігати та видаляти їх декільком користувачам одночасно.

Якщо перевищено обмеження по сумарній пам’яті файлів, сповіщати про це всіх користувачів сховища.

Сповіщати всіх користувачів про те, що хтось залив у сховище новий файл із інформацією про цей файл.

**Тексти коду програми**

|  |
| --- |
| main.c |
| #include "supaDrive.h"  #include "test.h"  #include <windows.h>  int main()  {  test\_SupaDrive();  puts("");  srand(time(NULL));  Callback Success = &SuccessMessage;  Callback Failure = &FailureMessage;  user\_t users[4];  users[0] = user\_new("Vlad");  users[1] = user\_new("Ruslan");  users[2] = user\_new("Elena");  users[3] = user\_new("Viktor");  file\_t files[9];  files[0] = file\_new ("File1", users[rand()%4], rand()%101+100);  files[1] = file\_new ("File2", users[rand()%4], rand()%101+100);  files[2] = file\_new ("File3", users[rand()%4], rand()%101+100);  files[3] = file\_new ("File4", users[rand()%4], rand()%101+100);  files[4] = file\_new ("File5", users[rand()%4], rand()%101+100);  files[5] = file\_new ("File6", users[rand()%4], rand()%101+100);  files[6] = file\_new ("File7", users[rand()%4], rand()%101+100);  files[7] = file\_new ("File8", users[rand()%4], rand()%101+100);  files[8] = file\_new ("File9", users[rand()%4], rand()%101+100);  SupaDrive\_t drive = SupaDrive\_new();  for (int i = 0; i < 9; i++){  Sleep(1000);  SupaDrive\_add(files[i], drive, Success, Failure);  }  SupaDrive\_free(drive);  for(int i = 0; i < 9; i++) {  file\_free(files[i]);  }  for (int i = 0; i < 4; i++) {  user\_free(users[i]);  }  return 0;  } |

|  |
| --- |
| supaDrive.c |
| #include "supaDrive.h"  struct SupaDrive\_s {  file\_t files[MAX\_FILES];  int amount;  int bytes;  };  SupaDrive\_t SupaDrive\_new () {  SupaDrive\_t self = malloc(sizeof(struct SupaDrive\_s));  self->amount = 0;  self->bytes = 0;  return self;  }  void SupaDrive\_free (SupaDrive\_t self) {  free(self);  }  int SupaDrive\_getBytes (SupaDrive\_t self) {  return self->bytes;  }  void SupaDrive\_add (file\_t file, SupaDrive\_t drive, Callback func1, Callback func2) {  if (file\_getBytes(file) + SupaDrive\_getBytes(drive) <= MAX\_BYTES){  drive->files[drive->amount] = file;  drive->amount++;  drive->bytes += file\_getBytes(file);  func1(file, drive);  }  else  func2(file, drive);  }  file\_t SupaDrive\_delete (SupaDrive\_t drive, file\_t file) {  int index = 0;  for (int i = 0; i < drive->amount; i++) {  if (drive->files[i] == file) {  index = i;  i = drive->amount;  }  }  file\_t deleted = drive->files[index];  for (int i = 0; i < drive->amount - index; i++) {  drive->files[index + i] = drive->files[index + i + 1];  }  drive->amount--;  drive->bytes -= file\_getBytes(deleted);  return deleted;  }  void SuccessMessage (file\_t file, SupaDrive\_t drive) {  char\* user = file\_getAuthor(file);  char\* fileName = file\_getName(file);  int fileBytes = file\_getBytes(file);  int driveBytes = SupaDrive\_getBytes(drive);  int freeSpace = MAX\_BYTES - driveBytes;  printf("%s added file '%s'(%i bytes) succesfully\nRemaining space on SupaDrive: %i bytes\n\n",  user, fileName, fileBytes, freeSpace);  }  void FailureMessage(file\_t file, SupaDrive\_t drive) {  char\* user = file\_getAuthor(file);  char\* fileName = file\_getName(file);  int fileBytes = file\_getBytes(file);  int driveBytes = SupaDrive\_getBytes(drive);  int requiredSpace = driveBytes + fileBytes - MAX\_BYTES;  printf("%s failed to add file '%s'(%i bytes) \nNot enough space on SupaDrive: additional %i bytes required\n\n",  user, fileName, fileBytes, requiredSpace);  } |

|  |
| --- |
| file.c |
| #include "file.h"  struct file\_s {  char name[MAX\_NAME];  user\_t author;  int bytes;  };  file\_t file\_new (const char\* name, user\_t user, int bytes) {  file\_t self = malloc(sizeof(struct file\_s));;  strcpy(self->name, name);  self->author = user;  self->bytes = bytes;  return self;  }  void file\_free (file\_t self) {  free(self);  }  char\* file\_getAuthor (file\_t self) {  return user\_getName(self->author);  }  char\* file\_getName (file\_t self) {  char\* name = self->name;  return name;  }  int file\_getBytes (file\_t self) {  return self->bytes;  }  void file\_print (file\_t file) {  printf("File : %s\nAdded by: %s\nSize: %i bytes\n\n", file->name,  user\_getName(file->author),  file->bytes);  } |

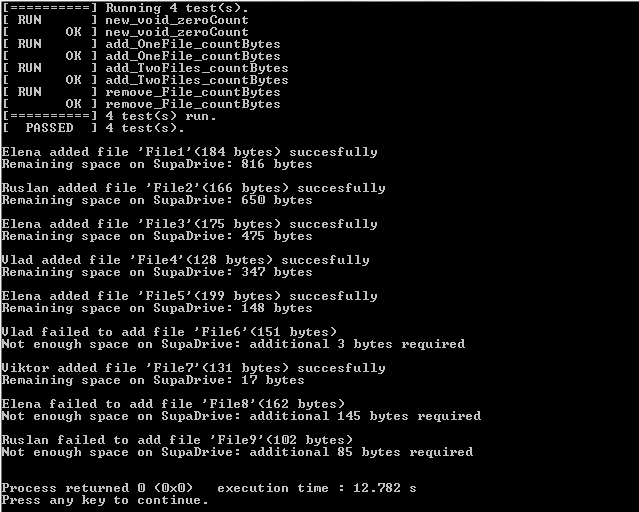
|  |
| --- |
| user.c |
| #include "user.h"  struct user\_s {  char name[MAX\_NAME];  };  user\_t user\_new (const char\* name) {  user\_t self = malloc(sizeof(struct user\_s));  strcpy(self->name, name);  return self;  }  void user\_free (user\_t self) {  free(self);  }  char\* user\_getName (user\_t self) {  char\* name = self->name;  return name;  } |

|  |
| --- |
| supaDrive.h |
| #ifndef SUPADRIVE\_H\_INCLUDED  #define SUPADRIVE\_H\_INCLUDED  #define MAX\_FILES 50  #define MAX\_BYTES 1000  #include "file.h"  #include <time.h>  typedef struct SupaDrive\_s \* SupaDrive\_t;  typedef void (\*Callback) (file\_t file, SupaDrive\_t drive);  SupaDrive\_t SupaDrive\_new ();  void SupaDrive\_free (SupaDrive\_t self);  int SupaDrive\_getBytes (SupaDrive\_t self);  void SupaDrive\_add (file\_t file, SupaDrive\_t drive, Callback func1, Callback func2);  file\_t SupaDrive\_delete (SupaDrive\_t drive, file\_t file);  void SuccessMessage (file\_t file, SupaDrive\_t drive);  void FailureMessage(file\_t file, SupaDrive\_t drive);  #endif // SUPADRIVE\_H\_INCLUDED |

|  |
| --- |
| file.h |
| #ifndef FILE\_H\_INCLUDED  #define FILE\_H\_INCLUDED  #include "user.h"  typedef struct file\_s \* file\_t;  file\_t file\_new (const char\* name, user\_t user, int bytes);  void file\_free (file\_t self);  void file\_print (file\_t file);  char\* file\_getAuthor (file\_t self);  char\* file\_getName (file\_t self);  int file\_getBytes (file\_t self);  #endif // FILE\_H\_INCLUDED |

|  |
| --- |
| user.h |
| #ifndef USER\_H\_INCLUDED  #define USER\_H\_INCLUDED  #define MAX\_NAME 100  #include <string.h>  #include <stdio.h>  #include <stdlib.h>  typedef struct user\_s \* user\_t;  user\_t user\_new (const char\* name);  void user\_free (user\_t self);  char\* user\_getName (user\_t self);  #endif // USER\_H\_INCLUDED |

**Приклади результатів**



**Висновки**

Навчився використовувати функції зворотнього виклику у мові С для структурування коду. Реалізував підтримку модулями подій на основі функцій зворотнього виклику для гнучкого налаштування поведінки модулів.

Створив набір модульних тестів для модуля SupaDrive. Його наведено у додатку А.

**Додаток А**

|  |
| --- |
| test.c |
| #include "test.h"  #include "supaDrive.h"  void TestMessage (file\_t file, SupaDrive\_t drive) {  }  static void new\_void\_zeroCount(void \*\*state)  {  SupaDrive\_t drive = SupaDrive\_new();  assert\_int\_equal(SupaDrive\_getBytes(drive), 0);  SupaDrive\_free(drive);  }  static void add\_OneFile\_countBytes(void \*\*state)  {  SupaDrive\_t drive = SupaDrive\_new();  user\_t user = "testUser";  Callback Cb1 = &TestMessage;  Callback Cb2 = &TestMessage;  int fileBytes = 100;  file\_t file = file\_new("test", user, fileBytes);  SupaDrive\_add(file, drive, Cb1, Cb2);  assert\_int\_equal(SupaDrive\_getBytes(drive), fileBytes);  SupaDrive\_free(drive);  file\_free(file);  user\_free(user);  }  static void add\_TwoFiles\_countBytes(void \*\*state)  {  SupaDrive\_t drive = SupaDrive\_new();  user\_t user = "testUser";  Callback Cb1 = &TestMessage;  Callback Cb2 = &TestMessage;  int fileBytes1 = 100;  int fileBytes2 = 150;  file\_t file1 = file\_new("test1", user, fileBytes1);  file\_t file2 = file\_new("test1", user, fileBytes2);  int sumBytes = fileBytes1 + fileBytes2;  SupaDrive\_add(file1, drive, Cb1, Cb2);  SupaDrive\_add(file2, drive, Cb1, Cb2);  assert\_int\_equal(SupaDrive\_getBytes(drive), sumBytes);  SupaDrive\_free(drive);  file\_free(file1);  file\_free(file2);  user\_free(user);  }  static void remove\_File\_countBytes(void \*\*state) {  SupaDrive\_t drive = SupaDrive\_new();  user\_t user = "testUser";  file\_t file = file\_new("test1", user, 150);  Callback Cb1 = &TestMessage;  Callback Cb2 = &TestMessage;  SupaDrive\_add(file, drive, Cb1, Cb2);  SupaDrive\_delete(drive, file);  assert\_int\_equal(SupaDrive\_getBytes(drive), 0);  SupaDrive\_free(drive);  user\_free(user);  }  int test\_SupaDrive(void)  {  const struct CMUnitTest tests[] =  {  cmocka\_unit\_test(new\_void\_zeroCount),  cmocka\_unit\_test(add\_OneFile\_countBytes),  cmocka\_unit\_test(add\_TwoFiles\_countBytes),  cmocka\_unit\_test(remove\_File\_countBytes),  };  return cmocka\_run\_group\_tests(tests, NULL, NULL);  } |

|  |
| --- |
| test.h |
| #ifndef TEST\_H\_INCLUDED  #define TEST\_H\_INCLUDED  #include "supaDrive.h"  #include <stdarg.h>  #include <stddef.h>  #include <setjmp.h>  #include <cmocka.h>  int test\_SupaDrive();  #endif // TEST\_H\_INCLUDED |