

МIНIСТЕРСТВО ОСВIТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**Лабораторна робота №** 2

з дисципліни “ Основи програмування ”

тема “Статичні та динамічні бібліотеки мовою 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 створювати та використовувати у різних проектах статичні та динамічні бібліотеки написані мовою С. Реалізувати динамічний вибір підвантаження частини функціоналу програми за допомогою динамічних бібліотек.

**Постановка завдання**

Виконати планування реалізації завдання 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.

Система автофіксації порушень правил дорожнього руху (ПДР). До системи підключено декілька постів, кожен з яких складається із радара, що визначає швидкість авто та камери, що розпізнає номер відповідного авто. Інформація (номер + швидкість) поступає від кожного поста у ядро системи, яке пересилає інформацію про порушника, що перевищили швидкість до терміналу оператора (консоль).

**Тексти коду програми**

|  |
| --- |
| main.c |
| #include <stdio.h>  #include <stdlib.h>  #include <time.h>  #include "queue.h"  #include "dynamicLibs.h"  const char\* chooseLib (void);  int main()  {  queue\_t queue = queue\_new();  dynamic\_t \* dll = dynamic\_init(chooseLib ());  if (NULL == dll) {  printf("Can't load dynamic!\n");  return 1;  }  if (NULL == dll->chk) {  printf("Can't get check function!\n");  return 1;  }  if (NULL == dll->react) {  printf("Can't get reaction function!\n");  return 1;  }  printf("Dynamic loaded!\n");  srand(time(NULL));  int i = 0;  for (i = 0; i < 7; i++){  queue\_add(queue, rand() % 25);  }  queue\_print(queue);  if(dll->chk(queue) == 1) {  dll->react();  }  queue\_free(queue);  return 0;  }  const char\* chooseLib (void) {  puts ("Enter number of library you want to use");  puts ("DynLib1 >> 1");  puts ("DynLib2 >> 2");  printf("enter your choise >> ");  int num;  scanf("%i", &num);  if (num == 1)  return "DynLib1.dll";  if (num == 2)  return "DynLib2.dll";  } |

|  |
| --- |
| dynamicLibs.c |
| #include "dynamicLibs.h"  dynamic\_t \* dynamic\_init(const char \* dllName) {  dynamic\_t \* dyn = malloc(sizeof(struct dynamic\_s));  dyn->hLib = LoadLibrary(dllName);  dyn->chk = NULL;  dyn->react = NULL;  if (NULL != dyn->hLib) {  dyn->chk = (chk\_f)GetProcAddress(dyn->hLib, "Check");  dyn->react = (react\_f)GetProcAddress(dyn->hLib, "Reaction");  return dyn;  } else {  return NULL;  }  }  void dynamic\_clean(dynamic\_t \* dyn) {  FreeLibrary(dyn->hLib);  free(dyn);  } |

|  |
| --- |
| queue.h |
| #ifndef QUEUE\_H\_INCLUDED  #define QUEUE\_H\_INCLUDED  typedef struct queue\_s \* queue\_t;  queue\_t queue\_new (void);  void queue\_add (queue\_t q, int value);  int queue\_remove (queue\_t q);  int queue\_getsize (queue\_t q);  void queue\_print(queue\_t q);  void queue\_free (queue\_t q);  #endif // QUEUE\_H\_INCLUDED |

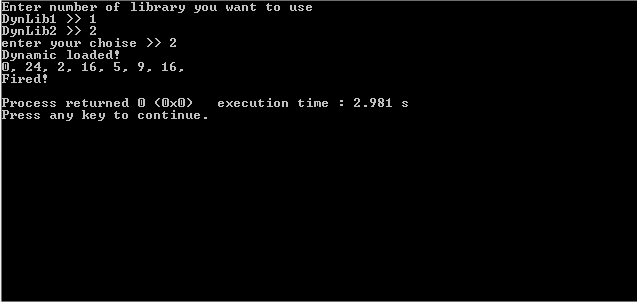
|  |
| --- |
| dyn.h |
| #ifndef \_\_MAIN\_H\_\_  #define \_\_MAIN\_H\_\_  #include <windows.h>  /\* To use this exported function of dll, include this header  \* in your project.  \*/  #ifdef BUILD\_DLL  #define DLL\_EXPORT \_\_declspec(dllexport)  #else  #define DLL\_EXPORT \_\_declspec(dllimport)  #endif  #ifdef \_\_cplusplus  extern "C"  {  #endif  #include "queue.h"  int DLL\_EXPORT Check (queue\_t q);  void DLL\_EXPORT Reaction (void);  #ifdef \_\_cplusplus  }  #endif  #endif // \_\_MAIN\_H\_\_ |

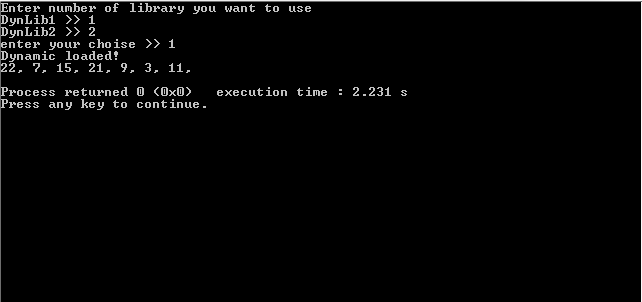
|  |
| --- |
| dyn1.c |
| #include "main.h"  #include <stdio.h>  // a sample exported function  int DLL\_EXPORT Check(queue\_t q)  {  int sum = 0, i;  for (i = 0; queue\_getsize(q) != 0; i++){  sum += queue\_remove(q);  }  if (sum < 40)  return 1;  else return 0;  }  void DLL\_EXPORT Reaction (void) {  printf("Warning\n");  }  extern "C" DLL\_EXPORT BOOL APIENTRY DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)  {  switch (fdwReason)  {  case DLL\_PROCESS\_ATTACH:  // attach to process  // return FALSE to fail DLL load  break;  case DLL\_PROCESS\_DETACH:  // detach from process  break;  case DLL\_THREAD\_ATTACH:  // attach to thread  break;  case DLL\_THREAD\_DETACH:  // detach from thread  break;  }  return TRUE; // succesful  } |

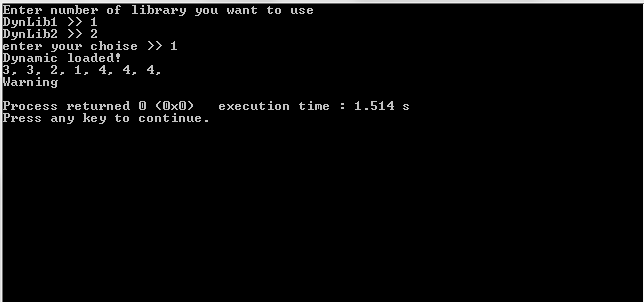
|  |
| --- |
| dyn2.c |
| #include "main.h"  #include <stdio.h>  // a sample exported function  int DLL\_EXPORT Check (queue\_t q)  {  int i, status = 0;  for (i = 0; queue\_getsize(q) != 0; i++){  if (queue\_remove(q) < 3) {  status++;  }  //if (status >= 2)  // break;  }  if (status >= 2)  return 1;  else return 0;  }  void DLL\_EXPORT Reaction (void) {  printf("Fired!\n");  }  extern "C" DLL\_EXPORT BOOL APIENTRY DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)  {  switch (fdwReason)  {  case DLL\_PROCESS\_ATTACH:  // attach to process  // return FALSE to fail DLL load  break;  case DLL\_PROCESS\_DETACH:  // detach from process  break;  case DLL\_THREAD\_ATTACH:  // attach to thread  break;  case DLL\_THREAD\_DETACH:  // detach from thread  break;  }  return TRUE; // succesful  } |

|  |
| --- |
| statlib.c |
| #include "queue.h"  struct queue\_s {  int\* elem;  int size;  };  queue\_t queue\_new (void) {  queue\_t q;  q = malloc(sizeof(struct queue\_s));  q->elem = malloc (sizeof(int)\*10);  q->size = 0;  return q;  }  void queue\_add (queue\_t q, int value) {  if (q->size == 10)  return;  q->elem[q->size] = value;  q->size++;  }  int queue\_remove (queue\_t q) {  if (q->size == 0)  return 1;  int i, val;  val = q->elem[0];  for (i = 0; i < q->size; i++){  q->elem[i] = q->elem[i+1];  }  q->size--;  return val;  }  int queue\_getsize (queue\_t q) {  return q->size;  }  void queue\_print(queue\_t q) {  int i;  for (i = 0; i < q->size; i++) {  printf("%i, ", q->elem[i]);  }  printf("\n");  }  void queue\_free (queue\_t q) {  free(q->elem);  free(q);  } |

**Приклади результатів**







**Висновки**

Навчився створювати та використовувати у різних проектах статичні та динамічні бібліотеки написані мовою С. Реалізував динамічний вибір підвантаження частини функціоналу програми за допомогою динамічних бібліотек.