Модуль №1 "Програмне забезпечення та апаратні засоби. Системи програмування"

Лабораторна робота 1.1

Ознайомлення з програмним забезпеченням для контролю за апаратурою комп'ютера та автоматизованого керування драйверами

Мета роботи: Ознайомлення з програмним забезпеченням для контролю за апаратурою комп'ютера та автоматизованого керування драйверами.

Завдання роботи: Здобути практичні навички у використанні программ для контролю за апаратурою комп'ютера та автоматизованого керування драйверами.

Теоретичні відомості

1. Засоби Windows для контролю за апаратним забезпеченням.

Операційна система – сукупність програмних засобів, що забезпечує управління апаратною частиною комп'ютера і прикладними програмами, а також їх взаємодією між собою і користувачем. У більшості обчислювальних систем операційні системи є основною частиною системного програмного забезпечення.

Кожна з сучасних розповсюджених операційних систем надає користувачу утиліти, що забезпечують ті або інші функції контролю стану аппаратури комп'ютера. Розглянемо два приклади таких утиліт.

1. Дефрагментація диска. Для дефрагментації диска необхідно зайти у пункт меню Пуск / Програми / Стандартні / Службові і вибрати Дефрагментація диска. У діалоговому вікні, що відкриється, необхідно обрати виконання дефрагментації і клацнути на піктограмі Аналіз. За результатами аналізу стану диска операційна система видасть оцінку використання диска до дефрагментації і висновок про доцільність проведення дефрагментації.

2. Управління живленням комп’ютера. Схема управління живленням – це набір апаратних і системних параметрів, які керують споживанням комп'ютером енергії та її економією. Схеми управління живленням дозволяють заощадити енергію, максимально збільшити швидкодію системи, або забезпечити оптимальне співвідношення між цими показниками.

Програма управління живленням доступна в Панелі керування, або ж її можна запустити за допомогою команди «Виконати», набравши в командному рядку powercfg.cpl.

2. Програма AIDA64.

AIDA64 (раніше – Everest, AIDA32) – популярна програма для перегляду інформації щодо апаратної и програмної конфигурації комп’ютера, розроблена компанією Lavalys. Програма має такі основні можливості:

1. Інформація про апаратне забезпечення. Повна інформація про складові комп'ютерів (материнські плати, процесори, відео- і мережеві адаптери, дисководи, пристрої введення, а також порти, зовнішні підключені пристрої і управління живленням.)

2. Інформація про програмне забезпечення. Детальна інформація про мережеві підключення і ресурси, користувачів, групи, поштові облікові записи, установки мережі, програмне забезпечення і операційну систему в цілому.

3. Функції настроювання параметрів безпеки. Запобігання виконання даних (наприклад, шкідливого коду) за допомогою функції DEP (Data Execution Prevention), спеціальні програми для створення файервола, антишпигунів, антітроянів і антивірусів.

4. Діагностика. Інтерфейс програми оснащений вбудованою панеллю AIDA64 CPUID, на якій компактно відображається інформація про процесор, материнську плату, оперативну пам'ять і архітектуру комп'ютера. Програма дозволяє відстежувати температуру процесорів, стан вентилятора і відносний знос диска, відстежувати проблеми сумісності, тощо. Також надаються дані про можливості «розгону» системи, динамічні оновлення та інше.

3. Керування драйверами. Програма Driver Genius.

Драйвер – комп'ютерна програма, за допомогою якої операційна система отримує доступ до керування апаратним забезпеченням. У

загальному випадку для використання кожного пристрою, підключеного до комп'ютера, необхідний спеціальний драйвер. Зазвичай операційна система вже містить драйвери для ключових компонентів апаратного забезпечення, без яких система не зможе працювати. Однак для більш специфічних пристроїв (графічна плата або принтер) можуть знадобитися спеціальні драйвери.

За допомогою диспетчера пристроїв Windows можна дізнатися про належність кожного драйвера до певного пристрою. Також можна подивитися список драйверів, які не відповідають вимогам PnP (Plug and Play – технологія, призначена для швидкого визначення і конфігурування пристроїв в комп'ютері і інших технічних пристроях). Для цього в меню «Вигляд» слід обрати пункт «Показати приховані пристрої», і в дереві, що відображає список основних приладів, з'явиться відповідний розділ з приладами, які не відповідають специфікації PnP.

Утиліта msinfo32 має розгалужену функціональність. Одна з її можливостей полягає в тому, щоб дізнатися, які драйвери встановлені в системі, які завантажені в даний момент і які можуть бути завантажені. З допомогою цієї утиліти можна побачити встановлені в системі драйвери і такі важливі їх параметри, як відповідне драйверу ім'я файлу, тип запуску, поточний стан і багато іншого. На жаль, цей метод можна застосувати тільки на локальній системі; при підключенні до комп'ютера по мережі подивитися, які драйвери на ньому встановлені, не вдасться.

Утиліта командного рядка sc.exe – консоль управління службами (Service Control Manager, SCM), яка управляє і драйверами. З її допомогою можна отримати список драйверів як локальної системи, так і віддалених систем (за умови наявності відповідних прав, звичайно). Простий виклик sc query type = driver дозволить вам швидко отримати шукане. Ця утиліта дуже зручна і проста в застосуванні і, крім того, дозволяє проводити опитування не лише локально, але і по корпоративній мережі. Варто відзначити, що в Windows NT 4.0 для управління драйверами у SCM був графічний інтерфейс, а в наступних версіях Windows його не стало.

Driver Genius – це програма для управління драйверами, яка виконує створення резервних копій драйверів, відновлення пошкоджених драйверів, оновлення застарілих і видалення

непотрібних драйверів, а також має також деякі можливості для діагностики апаратного забезпечення. Driver Genius дозволяє в автоматичному режимі знайти необхідні драйвери для всіх пристроїв, використовуючи дані про апаратне забезпечення системи. База даних Driver Genius налічує більше ніж 30 000 драйверів, серед яких є драйвери до материнських плат, відеокарт, звукових карт, мережевих карт, мишей, модемів, сканерів, клавіатур, принтерів і інших пристроїв. На кожен драйвер в програмі доступна офіційна версія тестів Windows Hardware Quality Lab (WHQL), через що у користувача відпадають будь-які турботи, пов'язані із сумісністю.

Порядок виконання роботи

1. Ознайомтесь з теоретичними відомостями.
2. Розгляньте Панель керування вашої версії Windows. Занесіть до звіту перелік утиліт з контролю аппаратного забезпечення, доступних у Вашій панелі керування, та короткі відомості про роботу з ними.
3. Встановіть на комп’ютер програму AIDA64, яка забезпечує контроль за апаратурою комп’ютера. Випробуйте основні можливості програми AIDA64.
4. Навчіться використовувати утиліти msinfo32, sc.exe.
5. Встановіть на комп’ютер програму Driver Genius, яка слугує для керування драйверами. Випробуйте основні можливості програми Driver Genius.
6. Зробіть висновки щодо роботи з програмами AIDA64 та Driver Genius.

Контрольні запитання

1. Що називається драйвером?
2. Дайте визначення поняття «операційна система».
3. Які утиліти операційної системи для контролю за апаратурою комп’ютера Вам відомі?
4. Назвіть основні можливості утиліти AIDA64.
5. Які функції виконує програма Driver Genius?
6. Охарактеризуйте основні можливості програми Driver Genius.

Література: [1], [3].

Лабораторна робота 1.2
Розробка у середовищі Embarcadero RAD Studio

Мета роботи: ознайомлення з середовищем Embarcadero RAD Studio, набуття навичок роботи з ним.

Завдання роботи: навчитися створювати прості програми у середовищі Embarcadero RAD Studio.

Теоретичні відомості

У візуальному програмуванні програма являє собою не просто послідовно виконуваний набір інструкцій, а сукупність підпрограм, що реагують на зовнішні події, такі як натискання клавіші, кнопки миші і т.д.

RAD (Rapid Application Development) – поняття, що описує принцип швидкої розробки програмного забезпечення з використання комплексного набору готових елементів візуального інтерфейсу та автоматичного генерування заготовок відповідних процедур. До систем RAD можна віднести Microsoft Visual Studio, Embarcadero RAD Studio, Delphi, C++ Builder, Prism, RadPHP.

Сучасні системи RAD в більшості випадків базуються на концепції об’єктно-орієнтованого програмування (ООП), що дозволяють значно збільшити складність програм і скоротити час їх розробки за рахунок більш ефективного повторного використання коду.

Всі популярні середовища швидкої візуальної розробки мають подібні елементи інтерфейсу, а саме: Інспектор об’єктів, редактор форм, редактор коду, палітру інструментів та ін.

За допомогою інспектора об’єктів ви можете змінювати властивості компонентів форми і визначати події, на які повинна реагувати та чи інша форма чи її компоненти. Вікно інспектора об’єктів має дві закладки: властивості та події.

Редактор форм відображає вигляд форми та дозволяє розробнику редагувати його як візуальний об’єкт. Тут ви визначаєте, як буде виглядати ваш додаток з погляду користувача. Ви вибираєте компоненти з палітри компонентів і перетягуєте їх на форму, використовуючи мишку для точного розташування і визначення розмірів компонента. Ви можете керувати зовнішнім виглядом і поведінкою компонента за допомогою інспектора об’єктів та редактора коду. Це і є основою візуального програмування.

Редактор коду дозволяє вводити розроблений вами код (програму) чи редагувати згенерований системою код для компонентів розробленої форми. Редактор коду використовує технологію вкладок, кожна вкладка відповідає своєму модулю чи файлу.

Палітра компонентів (Component Palette) містить готові елементи інтерфейсу для програми (кнопки, перемикачі, поля введення, діалоги тощо).

Порядок виконання роботи

1. Ознайомтесь з теоретичними відомостями.

2. Запустіть середовище розробки (Embarcadero RAD Studio) та виберіть новий проект C++ Builder. Після запуску перед вами з'явитися вікно вашого майбутнього додатка. Код, який вже згенерований, можна переглянути у вікні коду, для переходу до нього натисніть F12.

3. Тепер переходимо до редагування програми. Першим ділом поміняємо розмір форми, для цього слід клікнути в будь-якому місці форми. Після цього у вікні під назвою «Object inspector», яке знаходиться зліва, відредагуйте властивості Height і Width. (Перша властивість визначає висоту вікна, а друге його ширину.) Введіть у ці поля значення 150 і 180 відповідно. Також можна підписати вікно, для цього відредагуємо властивість Caption, ввівши, наприклад, значення «Програма».

3. Виберіть у розділі Standard палітри компонентів елемент Button (кнопка) і помістіть його на формі, після цього відредагуйте властивість Caption кнопки, поставивши значення «ОК». (Ця властивість визначає напис на кнопці.)

4. Виберіть на палітрі Standard елемент Label і помістіть його на формі, після цього відредагуйте властивість Caption, поставивши значення «Тут буде напис».

5. Натиснувши лівою клавішею миші на кнопці, виділіть її і перейдіть на вкладку Events вікна Object Inspector. Двічі клацніть лівою кнопкою миші в рядку процедури обробки події OnClick. (Дана подія відбувається, коли хтось натисне кнопку під час роботи програми.) Після цього відбудеться автоматичний перехід до вікна коду, у якому з'явиться згенерований код події, наведений нижче.

```
void __fastcall TForm1::Button1Click(TObject *Sender) //Назва процедури
{ //початок
    //місце для коду
} //кінець
```

Тепер допишемо потрібний код для роботи нашої програми.

А саме код Label1-> Caption = "Моя перша програма";

де:

Label1 – Назва об'єкта, в нашому випадку це напис.

Caption – Властивість об'єкта.

= – оператор присвоювання; за допомогою цього оператора ми присвоюємо нове значення властивості об'єкта.

" – лапки позначають початок і кінець рядкового значення.

; – позначає кінець оператора.

Після внесення даного рядка код повинен виглядати так.

```
void __ fastcall TForm1 :: Button1Click (TObject * Sender)
{
Label1-> Caption = "Моя перша програма";
}
```

Тепер запустіть програму і перевірте її роботу шляхом натискання кнопки «ОК».

Контрольні запитання

1. Що мається на увазі під поняттям RAD?
2. Яку функцію виконує інспектор об'єктів?

3. Для чого призначений редактор форм?
4. Для чого призначений редактор коду?
5. Яку функцію виконує палітра компонентів?

Література: [3], [5].

Лабораторна робота 1.3
Розробка у середовищі Microsoft Visual Studio .NET

Мета роботи: ознайомлення з середовищем Microsoft Visual Studio .NET, набуття навичок роботи з ним.

Завдання роботи: навчитися створювати прості програми у середовищі розробки Microsoft Visual Studio .NET.

Теоретичні відомості

Microsoft Visual Studio – лінійка продуктів компанії Майкрософт, що включають інтегроване середовище розробки програмного забезпечення і ряд інших інструментальних засобів.

Інтегроване середовище розробки – це система програмних засобів, пов'язаних спільним інтерфейсом, використовувана програмістами для розробки програмного забезпечення (ПЗ).

Інтерфейс користувача Microsoft Visual Studio .Net включає в себе ряд елементів: Toolbox, Class View, Solution Explorer, Task List, Command Window, текстове поле Find та інші.

Toolbox

Інструмент Toolbox є репозиторієм, що «містить» всі основні візуальні компоненти. Додавання компонента до проекту відбувається шляхом перенесення компонента з Toolbox на одну з форм проекту.

Вельми цікавою вкладкою на Toolbox є вкладка Clipboard, яка містить елементи, скопійовані в буфер обміну, і дозволяє здійснювати перегляд вмісту буфера і швидку вставку.

Class View

ClassView дозволяє за допомогою перетягування (drag & drop) «витягувати» імена методів, властивостей і класів безпосередньо у вікно редагування тексту, а також здійснювати пошук визначення функції. У Visual Studio .NET, Class View також містить список методів і властивостей базових класів, що дозволяє легко переходити до опису цих методів і властивостей в Object Browser (для цього необхідно двічі клацнути по обраному елементу списку).

Solution Explorer

Solution Explorer відображає список всіх файлів поточного проекту. За допомогою цього інструменту можна легко створювати нові директорії для різних файлів (за типом, за логікою або на розсуд користувача), викликати вікно властивостей для обраного файлу. Також, за допомогою цього вікна можна легко додавати нові елементи або посилання до проекту: достатньо клацнути правою кнопкою миші по References і вибрати у випадаючому меню тип посилання, після чого відобразитися відповідне вікно.

Task List

Task List – список задач; зручний інструмент, що допомагає у роботі над вихідними файлами проекту, особливо якщо над одним і тим же кодом працюють кілька людей. Залиште коментар, наприклад: "'TODO: Перевір цей код!", і він тут же з'явиться у списку завдань вікна Task List, разом з описом та посиланням на рядок у коді, в якому залишений коментар. Подвійне клацання по повідомленню автоматично переведе виділення на цей рядок. Крім цього, ви можете додати будь-яку задачу в список, встановити їй пріоритет, ставити або знімати галочки, що символізують виконання задачі.

Command Window

Це вікно, у кому можна вводити команди у текстовій формі. За допомогою цього інструменту можна, зокрема, отримувати і встановлювати значення змінних і властивостей елементів управління.

Текстове поле Find

Для швидкого пошуку текстового рядка у файлах проекту зручно користуватися розташованим на панелі інструментів Standard текстовим полем Find. Цікавою властивістю цього поля є те, що в ньому можна виконувати всі ті ж команди, що й у вікні Command Window; для цього достатньо на початку команди ввести символ ">".

Порядок виконання роботи

1. Ознайомтесь з теоретичними відомостями.
2. Запустіть середовище Microsoft Visual Studio та натисніть File – New Project або іконку New Project на панелі швидкого доступу. Далі виберіть тип додатка – Windows Forms Application. Зрештою, натисніть Finish. Перед нами з'являється вікно нової програми.
3. Розгляньте інтерфейс середовища розробки. Зверніть увагу на вкладку Toolbox з лівого боку, елементи якої розкриваються при наведенні на них курсора миші. Розкрийте вкладку Toolbox і огляньте докладно блок Common Controls. Наприклад, першим в списку у цьому блоці стоїть компонент Button – це звичайна кнопка. Такі ж кнопки ви бачите у всіх програмах, що мають візуальний інтерфейс.
4. Щоб додати елемент Button на форму, зробіть подвійний клік по цьому елементу у списку. Тепер Ви можете за допомогою курсору миші розмістити його на формі додатка. Після розміщення можна пересувати елемент по площині форми і змінювати його розміри.
5. Поверніться на вкладку Toolbox. У блоці Common Control знайдіть компонент TextBox. Це текстове поле редагування; саме в нього ми вводимо адреси посилань, логіни, паролі та іншу подібну інформацію.
6. Щоб додати елемент TextBox на форму, повторіть ті ж дії, які ви виконували з кнопкою. Нам буде потрібно два елементи TextBox на формі, так що розмістіть два екземпляри. Суть роботи програми не залежить від взаємного розташування візуальних елементів, тому розмістіть їх на формі так, як вам зручно.
7. Знайдіть у списку елемент Label. Він слугуватиме, так би мовити, «екраном», на який програма буде виводити відповідь. Цей

елемент також доступний на вкладці Toolbox в блоці Common Controls. Додайте елемент Label на форму у будь-яке місце. Зовнішній вигляд нашого додатку сформовано.

8. Тепер приступимо до створення коду програми. Зробіть подвійний клік по кнопці Button1. Відкриється вікно дизайнера коду. В цьому вікні ви побачите заготовку коду процедури, що оброблятиме подію натиснення на кнопку.

9. Відредагуйте код у вікні дизайнера так, щоб він виглядав наступним чином:

```
namespace WindowsFormsApplication1
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }

      private void button1_Click(object sender, EventArgs e)
      {
         int result = Convert.ToInt32(this.textBox1.Text)
            + Convert.ToInt32(this.textBox2.Text);
         this.label1.Text = result.ToString();
      }
   }
}
```

Як бачимо, дана програма отримує з текстових рядків два числа та визначає їх суму. Візуальні елементи, які ми розмістили на формі, у програмі мають назви label, textBox1 і textBox2. Функція Convert.ToInt32() конвертує рядкові значення (string) у числові (int); метод .ToString() виконує зворотне перетворення.

10. Спробуйте роботу програми. Для запуску програми перейдіть в меню Debug - Start Debugging, або натисніть F5.

Контрольні запитання

1. Охарактеризуйте загальний принцип створення візуального додатку (Windows Forms Application) у середовищі Microsoft Visual Studio .Net.
2. Які компоненти інтерфейсу Microsoft Visual Studio .Net Ви знаєте?
3. Яку функцію виконує елемент Toolbox?
4. Для чого призначений Solution Explorer?
5. Для чого призначений Class View?
6. Яку функцію виконує Task List?
7. Для чого слугує текстове поле Find?

Література: [3], [5].

Лабораторна робота 1.4
Створення та використання програмних бібліотек

Мета роботи: ознайомлення з середовищем програмування Visual C++; ознайомлення з поняттям програмних бібліотек, способами їх створення та використання.

Завдання роботи: створити програмну бібліотеку та програму, яка використовує функції бібліотеки.

Теоретичні відомості

Програмна бібліотека – це зібрання об'єктів чи підпрограм для вирішення близьких за тематикою задач. Бібліотеки містять початковий код та дані, необхідні для інтеграції нових можливостей в програмні рішення.

Поняття «бібліотека» може означати те саме, що «модуль», або «декілька модулів».

Бібліотеки діляться на *статичні* та *динамічні*.

Статичні бібліотеки

Статичні бібліотеки можуть мати вигляд початкового тексту, що підключається програмістом до своєї програми на етапі написання. (Наприклад, для мови Fortran існує величезна кількість бібліотек для вирішення різних задач, які розповсюджуються саме в

початкових текстах). Статичні бібліотеки також можуть надаватися у вигляді об'єктних файлів, що приєднуються (лінкуються) до виконуваної програми на етапі компіляції. У Microsoft Windows такі файли мають розширення .lib, у UNIX-подібніх ОС – зазвичай розширення .a .

В результаті використання статичної бібліотеки код програми містить код всіх необхідних функцій. Це робить програму автономною, але збільшує її розмір.

Динамічні бібліотеки

Також називаються розділюваними бібліотеками (англ. – shared library), бібліотеками часу виконання (англ. – runtime library), або бібліотеками, що динамічно підключаються (англ. – dynamic link library, DLL). Це окремі файли, що надають програмі набір використовуваних функцій для завантажування на етапі виконання при зверненні програми до ОС із заявкою на виконання функції з бібліотеки. Якщо необхідна бібліотека вже завантажена в оперативну пам'ять, програма використовуватиме завантажену копію бібліотеки. Такий підхід дозволяє зекономити час і пам'ять, оскільки декілька програм можуть використовувати одну копію бібліотеки, вже завантажену в пам'ять.

Динамічні бібліотеки зберігаються зазвичай у визначеному місці й мають стандартне розширення. Наприклад, файли .library у логічному томі Libs: у AmigaOS; у Microsoft Windows і OS/2 файли бібліотек загального користування мають розширення .dll; у UNIX-подібних ОС – зазвичай .so; у MacOS X – .dylib.

При написанні програми програміст вказує транслятору мови програмування (компілятору або інтерпретатору), що слід підключити певну бібліотеку і використовувати певну функцію зі вказаної бібліотеки. У випадку використання динамічних бібліотек ні початковий текст, ні виконуваний код функції до складу програми не входить.

Порядок виконання роботи

1. Ознайомтесь з теоретичними відомостями.

2. Запустіть середовище Visual C++ і створіть у ньому проект статичної бібліотеки:

• У меню Файл виберіть пункт Створити і потім пункт Проект ....

• У вузлі C++ області Типи проектів виберіть Win32.

• В області Шаблони виберіть Консольний додаток Win32.

• Виберіть ім'я проекту, наприклад MathFuncsLib, і введіть його в поле Ім'я. Виберіть ім'я рішення, наприклад StaticLibrary, і введіть його в полі Ім'я рішення.

• Для запуску майстра додатків Win32 натисніть кнопку ОК. На сторінці Загальні відомості діалогового вікна Майстер додатків Win32 натисніть кнопку Далі.

• На сторінці Параметри додатка діалогового вікна Майстер додатків Win32 в полі Тип програми виберіть пункт Статична бібліотека.

• На сторінці Параметри додатка діалогового вікна Майстер додатків Win32 в поле Додаткові параметри зніміть прапорець Передкомпільований заголовок.

• Щоб створити проект, натисніть кнопку Готово.

3. Додайте клас в статичну бібліотеку

Клас є представленням типу об'єкта; його можна представити як креслення, що описує об'єкт. Подібно до того, як одне креслення може бути використаний для побудови декількох будівель, окремий клас може бути використаний для створення необхідної кількості об'єктів.

• Щоб створити файл заголовка для нового класу, в меню Проект виберіть команду Додати новий елемент .... Відкриється діалогове вікно Додавання нового елемента. У вузлі Visual C++ області Категорії виберіть пункт Код. В області Шаблони виберіть пункт Заголовний файл (. H). Виберіть ім'я заголовного файлу, наприклад MathFuncsLib.h, і натисніть Додати. Відобразиться порожній файл.

• Додайте простий клас з ім'ям MyMathFuncs, що здійснює звичайні арифметичні операції, такі як додавання, віднімання, множення і ділення. Код повинен виглядати приблизно таким чином:

```
// MathFuncsLib.h
namespace MathFuncs
```

```
{
   class MyMathFuncs
   {
   public:
      // Returns a + b
      static double Add(double a, double b);
      // Returns a - b
      static double Subtract(double a, double b);
      // Returns a * b
      static double Multiply(double a, double b);
      // Returns a / b
      // Throws DivideByZeroException if b is 0
      static double Divide(double a, double b);
   };
}
```

• Щоб створити вихідний файл для нового класу, в меню Проект виберіть команду Додати новий елемент .... Відкриється діалогове вікно Додавання нового елемента. У вузлі Visual C + + області Категорії виберіть пункт Код. В області Шаблони виберіть пункт Файл C + + (. Cpp). Виберіть ім'я вихідного файлу, наприклад MathFuncsLib.cpp, і натисніть Додати. Відобразиться порожній файл.

• Реалізуйте функціональність класу MyMathFuncs у вихідному файлі. Код повинен виглядати приблизно таким чином:

```
// MathFuncsLib.cpp
// compile with: /c /EHsc
// post-build command: lib MathFuncsLib.obj
#include "MathFuncsLib.h"
#include <stdexcept>
using namespace std;
namespace MathFuncs
{
   double MyMathFuncs::Add(double a, double b)
   {
      return a + b;
   }
   double MyMathFuncs::Subtract(double a, double b)
```

```
    {
        return a - b;
    }
    double MyMathFuncs::Multiply(double a, double b)
    {
        return a * b;
    }
    double MyMathFuncs::Divide(double a, double b)
    {
        if (b == 0)
        {
            throw new invalid_argument("b cannot be zero!");
        }
        return a / b;
    }
}
```

• Щоб побудувати статичну бібліотеку проекту, в меню Проект виберіть СвойстваMathFuncsLib. У лівій області в полі Властивості конфігурації виберіть Загальні. У правій області в полі Тип конфігурації виберіть Статична бібліотека (. Lib). Натисніть кнопку ОК для збереження змін.

• Скомпілюйте статичну бібліотеку, вибравши команду Побудувати рішення в меню Побудова. В результаті буде створена статична бібліотека, яка може використовуватися іншими програмами.

4. Створіть додаток, що посилається на статичну бібліотеку

• Щоб створити додаток, що буде посилатися і використовувати створену раніше статичну бібліотеку, в меню Файл виберіть пункт Створити і потім пункт Проект ....

• У вузлі C + + області Типи проектів виберіть Win32.

• В області Шаблони виберіть Консольний додаток Win32.

• Виберіть ім'я проекту, наприклад MyExecRefsLib, і введіть його в поле Ім'я. У списку поруч із полем Рішення виберіть пункт Додати в рішення. Після цього новий проект буде додано до того ж рішення, що і статична бібліотека.

• Для запуску майстра додатків Win32 натисніть кнопку ОК. На сторінці Загальні відомості діалогового вікна Майстер додатків Win32 натисніть кнопку Далі.

• На сторінці Параметри додатка діалогового вікна Майстер додатків Win32, в полі Тип програми, виберіть пункт Консольний додаток.

• На сторінці Параметри додатка діалогового вікна Майстер додатків Win32, в полі Додаткові параметри, зніміть прапорець Передкомпільований заголовок.

• Щоб створити проект, натисніть кнопку Готово.

5. Випробуйте можливість використання функціональних можливостей статичної бібліотеки в консольному додатку.

• По завершенні процесу створення консольного додатку майстер створить порожню програму. Ім'я вихідного файлу буде збігатися з іменем, вибраним раніше для проекту. У цьому прикладі він має ім'я MyExecRefsLib.cpp.

• Для використання математичних процедур з статичної бібліотеки необхідно послатися на цю бібліотеку. Для цього в меню Проект виберіть пункт Посилання ... . У діалоговому вікні Вікна властивостей розгорніть вузол Загальні властивості і виберіть пункт Посилання. Потім натисніть кнопку Додати нове посилання ....

• З'явиться діалогове вікно Додати посилання. У цьому діалоговому вікні відображається список всіх бібліотек, на які можна посилатися. На вкладці Проект перераховуються всі проекти поточного рішення і включені в них бібліотеки. На вкладці Проекти виберіть MathFuncsLib. Натисніть кнопку ОК.

• Для створення посилання на заголовні файли статичної бібліотеки необхідно змінити шлях до каталогів включення. Для цього в діалоговому вікні Вікна властивостей послідовно розгорніть вузли Властивості конфігурації, C/C++, а потім виберіть Загальні. Поряд з полем Додаткові каталоги включення введіть шлях до місця розміщення заголовного файлу MathFuncsLib.h.

• Тепер клас MyMathFuncs можна використовувати в додатку. Замініть код у файлі MyExecRefsLib.cpp наступним кодом:

```
// MyExecRefsLib.cpp
// compile with: /EHsc /link MathFuncsLib.lib
#include <iostream>
```

```
#include "MathFuncsLib.h"
using namespace std;
int main()
{
   double a = 7.4;
   int b = 99;
   cout << "a + b = " <<
      MathFuncs::MyMathFuncs::Add(a, b) << endl;
   cout << "a - b = " <<
      MathFuncs::MyMathFuncs::Subtract(a, b) << endl;
   cout << "a * b = " <<
      MathFuncs::MyMathFuncs::Multiply(a, b) << endl;
   cout << "a / b = " <<
      MathFuncs::MyMathFuncs::Divide(a, b) << endl;
   return 0;
}
```

• Побудуйте виконуваний файл, використовуючи команду Побудувати рішення в меню Побудова.

Контрольні запитання

1. Поясніть поняття «програмна бібліотека».
2. Для чого призначені статичні бібліотеки?
3. Як можна створити статичну бібліотеку?
4. Для чого призначені розділювані бібліотеки?
5. Як знайти файли бібліотек у файловій системі Вашого комп'ютера? Який формат мають ці файли?

Література: [3], [5].

Лабораторна робота 1.5
Ознайомлення з середовищами розробки драйверів

Мета роботи: ознайомлення з середовищами розробки драйверів, основними відомостями про програмну структуру драйверів та засобами їх створення.

Завдання роботи: створити простий драйвер та використати його у комп’ютерній системі.

Теоретичні відомості

Драйвер – це комп'ютерна програма, що дозволяє іншим, більш високорівневим комп'ютерним програмам, взаємодіяти з апаратним пристроєм, віртуальним пристроєм, інформаційним об’єктом, протоколом, або іншим драйвером. Найбільш відомим різновидом драйверів є драйвери фізичних пристроїв. Драйвери використовуються як прикладними, так і системними програмами, зокрема компонентами операційних систем.

Звичайно стандартні функції драйвера обробляють 7 основних подій, пов’язаних з функціонуванням драйвера в системі:

1. Завантаження драйвера. Під час свого завантаження драйвер реєструється в системі, проводить первинну ініціалізацію і т. д.
2. Вивантаження драйвера. Під час свого вивантаження драйвер звільняє раніше захоплені ним ресурси – пам'ять, файли, пристрої та ін.
3. Відкриття драйвера (початок основної роботи). Зазвичай драйвер відкривається програмою як файл, для цього використовуються функції createfile() у Win32 або fopen() у UNIX-подібних системах;
4. Читання інформації з регістрів або адресного простору пристрою, що обслуговується драйвером.
5. Запис інформації у регістри або адресний простір пристрою.
6. Закриття драйвера – операція, зворотна відкриттю, звільняє зайняті при відкритті ресурси і знищує дескриптор файлу;
7. Спеціалізоване управління введенням-виведенням (I/O control, ioctl). Часто драйвер підтримує специфічний для даного пристрою інтерфейс введення-виведення. За допомогою цього інтерфейсу програма може надіслати одну з спеціальних команд, які підтримує даний пристрій. Наприклад, для SCSI пристроїв можна надіслати команду GET_INQUIRY, щоб отримати опис пристрою. У Win32 системах управління здійснюється через API-функцію DeviceIoControl(); у UNIX-подібних – через стандартну POSIX-функцію ioctl().

На відміну від прикладної програми, драйвер не є процесом і не має власного потоку виконання. Натомість будь-яка функція

драйвера виконується в контексті того потоку і процесу, в якому вона була викликана, і сам процес її виклику відрізняється від процесу звичайного виклику функції; як правило, при зверненні до коду драйвера відбувається передача так званого запиту вводу/виводу (I/O Request, I/O Request Package).

Порядок виконання роботи

1. Ознайомтесь з теоретичними відомостями.
2. Встановіть одне з середовищ розробки драйверів (Driver Development Kit, Windows Driver Kit), що відповідає Вашій операційній системі.
3. Розгляньте структуру директорій, створених при встановленні програмного забезпечення. Вивчіть їх вміст.
4. Ознайомтесь з системою допомоги середовища розробки.
5. Спробуйте запустити окремі утиліти, що входять до складу середовища. Вивчіть їх функціональні можливості.
6. Розгляньте зразки коду драйверів, що поставляються разом з середовищем розробки. Знайдіть у цьому коді основні функції, що визначають роботу драйвера.
7. Використовуючи зразки початкового коду, скомпілюйте та встановіть у системі прототип драйвера.
8. Підготуйте звіт щодо виконаної роботи та дайте відповіді на контрольні запитання.

Контрольні запитання

1. Що називається драйвером?
2. Які функції може виконувати драйвер у комп’ютерній системі? Наведіть приклади.
3. Які моделі розробки драйверів Ви знаєте? В чому полягає різниця між ними?
4. Охарактеризуйте утиліти, що входять до складу середовища розробки драйверів.
5. Які основні складові частини можуть бути присутні у коді драйвера?
6. Які дії необхідно виконати, щоб створити драйвер для конкретного пристрою?

Література: [3], [5].