

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ФАХОВИЙ КОЛЕДЖ РАКЕТНО-КОСМІЧНОГО МАШИНОБУДУВАННЯ
ДНІПРОВСЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ ім. О.Гончара

МЕТОДИЧНА РОЗРОБКА
для забезпечення самостійної роботи студентів

З НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

«Об’єктно-орієнтоване програмування»

для студентів денної форми навчання за спеціальністю

121 «Інженерія програмного забезпечення»

(код і назва спеціальності)

Розроблено викладачем

_____/ Гапоненко Н.В./

Розглянуто та схвалено

на засіданні ЦК ПІ

Протокол № _____ від ____ . ____ .20

Голова ЦК ПІ

_____/Ланська С.С./

План самостійного вивчення теми № 1

з навчальної дисципліни "Об'єктно-орієнтоване програмування"

Тема	Інструментарій середовища візуального програмування.
	Створення консольного додатку. Робота з проєктом
Мета	Ознайомитись з інструментарієм середовища візуального програмування, навчитись створювати консольний додаток, працювати з проєктом
	<i>знати:інструментарій середовища візуального програмування</i>
	<i>вміти:використовувати інструментарій середовища візуального програмування</i>

Час – 3 години

Навчальні питання:

- 1 Інтегроване середовище розробки
- 2 Створення проєкту консольного додатку
- 3 Робота з проєктом консольного додатку

Методичні рекомендації

Самостійно вивчаючи запропоновану тему, студент повинен засвоїти такі основні положення:

<i>По-перше</i>	Інтегроване середовище розробки — це програмний продукт, який об'єднує текстовий редактор, компілятор, відладчик і довідкову систему
<i>По-друге</i>	Проект — це набір взаємопов'язаних вихідних файлів, компіляція і компонування яких дозволяє створити виконувану програму
<i>По-третє</i>	Пакет включає інструментарій, необхідний для роботи в середовищі програмування

Література

Довбуш, Г. Ф. Visual C++ на прикладах [Текст] / Г. Ф. Довбуш, А. Д. Хомоненко; под ред. А. Д. Хомоненко. – СПб. : БХВ-Петербург, 2007. -528 с.

Навчальні матеріали до самостійного вивчення теми

ВСТУП

Integrated Development Environment (інтегроване середовище розробки), або, скорочено, IDE — це програмний продукт, об'єднуючий текстовий редактор, компілятор, відладчик і довідкову систему. До складу Microsoft Visual Studio входить IDE Microsoft Visual C++.

1 Інтегроване середовище розробки

Виклик Visual C++ здійснюється або через меню Пуск>Програми>Microsoft Visual C++, або клацанням мишею по піктограмі з відповідним ім'ям, розміщеної на робочому столі комп'ютера. Після запуску Visual C++ з'являється головне вікно програми.

Робочий стіл Visual C++ включає в себе три вікна, наведені нижче.

Вікно Project Workspace (вікно робочої області) призначено для надання допомоги при написанні і супроводі великих багатофайлових програм. Спочатку воно закрите, але після створення нового проекту (або завантаження раніше створеного проекту) одна з вкладок цього вікна буде містити список файлів проекту.

Вікно Editor (вікно редактора) використовується для введення та перевірки вихідного коду.

Вікно Output (вікно виводу) служить для виведення повідомлень про хід компіляції, складання та виконання програми. Зокрема, повідомлення про помилки з'являються саме в цьому вікні.

Під заголовком головного вікна, як і у всіх Windows-додатках, знаходиться рядок меню. Для кнопок панелей інструментів передбачена зручна контекстна допомога: при наведенні курсору миші на кнопку затримці на секунду-другу спливе підказка з призначенням кнопки.

Середовище розробки дозволяє будувати проекти різних типів, орієнтовані на різні сфери застосування. Так як середовище спроектована на Windows-платформі, то майже всі типи проектів є віконними Windows-додатками з відповідним графічним інтерфейсом.

Розробники середовища передбачили роботу і з так званими консольними додатками. При запуску консольного застосування операційна система створює так зване консольне вікно, через яке йде весь ввід-вивід програми. Цей тип

додатків найбільше підходить для мети вивчення мови C / C + +, так як компільовані програми не «покриваються» товстим шаром проміжного Windows-коду.

2 Створення проекту консольного додатку

Будь-яка програма, що створюється в середовищі Visual C + +, завжди оформляється як окремий проект (project). Проект - це набір взаємопов'язаних вихідних файлів і, при необхідності, підключаємих (заголовних) файлів , компіляція і компонування яких дозволяє створити виконувану програму.

Є поняття робочої області проекту (project workspace). Робоча область може містити будь-яку кількість різних проектів, згрупованих разом для узгодженої розробки: від окремого додатка до бібліотеки функцій або цілого програмного пакета. Очевидно, що для вирішення наших навчальних завдань кожна програма буде втілюватися у вигляді одного проекту, тому робоча область проекту у нас завжди буде містити один проект.

Для створення нового проекту типу «консольний додаток» слід виконати дії, подібні наведеним далі.

1. Виберіть у рядку меню головного вікна команду File> New ...

2. У діалоговому вікні New виберіть вкладку Projects:

- виберіть тип Win32 Console Application;
- введіть ім'я проекту в текстовому полі Project Name, наприклад First;
- введіть ім'я каталогу розміщення файлів проекту в текстовому полі Location1 (якщо зазначений каталог відсутній, то його буде створено автоматично);
- клацніть лівою кнопкою миші на кнопці OK.

3. Після цього запускається так званий майстер додатків Application Wizard, який відкриває діалогове вікно Win32 Console Application - Step1 of 1 з пропозицією визначитися, який підтип консольного застосування вам потрібно створити. Виберіть тип An empty project; клацніть на кнопці Finish.

4. Після цього з'явиться вікно New Project Information зі специфікаціями проекту та інформацією про каталог, в якому буде розміщено створюваний проект. Клацніть на кнопці OK.

Рекомендується для розміщення проектів виділити спеціальну папку, наприклад D: Student MyProjects.

Перш ніж продовжувати роботу, заглянемо в папку First, створену майстром програм для проекту, а точніше — для робочої області. Там є файли First.dsw, First.dsp, First.opt, First.ncb, а також папка Debug (або Release — залежно від конфігурації проекту). Дамо короткий опис кожного з файлів.

First.dsw — файл робочої області проекту, який використовується всередині інтегрованого середовища розробки. Він об'єднує всю інформацію про проекти, які входять у цю робочу область.

First.dsp — проектний файл, який використовується для побудови (building) окремого проекту або підпроекту.

First.opt — файл, що містить опції робочої області проекту. Завдяки цьому файлу при кожному відкритті робочої області проекту всі параметри, вибрані під час останнього сеансу роботи з даної робочої областю, будуть відновлені.

First.ncb — службовий файл. Він створюється компілятором і містить інформацію, яка використовується в інструменті інтегрованого середовища під названням ClassView. Панель ClassView знаходиться у вікні Project Workspace і показує всі класи C++, для яких доступні визначення в рамках даного проекту, а також всі елементи цих класів.

Debug — папка, в яку будуть міститися файли, що формуються компілятором і компоновальником. Виконуваний файл має розширення exe.

У вікні Project Workspace з'явилися дві сторінки: Class View і File View. File View призначена для перегляду списку файлів проекту. Відкриємо список First files, клацнувши мишею на значку «+». З'явиться дерево списку файлів, що містить піктограми трьох папок: Source Files, Header Files, Resource Files. Так як в консольних додатках файли ресурсів не використовуються, папка Resource Files порожня. Інші папки порожні, тому що підтип консольного застосування An empty project.

3 Робота з проектом консольного додатку

Зберегти проект можна за допомогою команди File, Save All.

Виконати програму за допомогою команди Build, Execute First.exe.

Закрити проект за допомогою команди File, Close Workspace.

Компіляція, компоновка і виконання проекту можуть бути виконані або через меню Build головного вікна, або за допомогою кнопок панелі інструментів. Опишемо коротко основні команди меню Build:

- ° Compile — компіляція обраного файлу. Результати компіляції виводяться у вікно Output.

- ° Build — компоновка проекту. Компілюються всі файли, в яких відбулися зміни з моменту останньої компонування. Після компіляції відбувається збірка (link) всіх об'єктних модулів, включаючи бібліотечні, в результуючий виконуваний файл. Повідомлення про помилки компоновки виводяться у вікно Output. Якщо обидві фази компонування завершилися без помилок, то створений виконуваний файл з розширенням. Exe може бути запущений на виконання.

- ° Rebuild All — те ж, що і Build, але компілюються всі файли проекту незалежно від того, чи були в них зроблені зміни.

- ° Execute — виконання файлу, створеного в результаті компоновки проекту.

Visual C ++ дозволяє будувати проект або в відлагодчій конфігурації (Win32 Debug), або у випускній конфігурації (Win32 Release). Виберіть в меню Project пункт Settings ... Відкриється діалогове вікно Project Settings. Перевірте, яке значення встановлено у вікні комбінованого списку Settings For: .. Якщо це не Win32 Debug, то перейдіть на потрібне значення через команду меню Build, Set Active Configuration ...

Щоб закінчити роботу над проектом можна вибрати меню File, пункт Close Workspace або закрити програму Visual C ++.

Відкрити проект, над яким ви працювали раніше можна способами:

1. В Visual C ++ виберіть меню File, пункт Open Workspace ..., знайдіть папку з вашим проектом, а в ній файл ProjectName.dsw, відкрийте файл, клацнувши мишею.

2. В Visual C ++ виберіть меню File, пункт Recent Workspaces, в списку останніх файлів, з якими йшла робота, знайдіть файл ProjectName.dsw, клацніть мишею.

3. В провіднику знайдіть папку з вашим проектом, а в ній файл ProjectName.dsw, клацніть мишею.

Висновки

IDE Microsoft Visual C++ — це програмний продукт, який об'єднує текстовий редактор, компілятор, відладчик і довідкову систему.

Перевірена і налагоджена виконується програма може виконуватися на комп'ютері до тих пір, поки вона існує. Після запуску виконуваної програми для вирішення завдання завантажувач розміщує програму в оперативній пам'яті. Центральний процесор вибирає і виконує кожну інструкцій програми, зберігаючи нові значення даних по мірі виконання програми.

Пакет MVC + + включає інструментарій, необхідний для роботи в середовищі програмування.

Питання для самоконтролю знань студентів

- 1 Дати огляд інструментарію пакету IDE Microsoft Visual C++.
- 2 Як створити консольний проект у Microsoft Visual C++, яке призначення файлів проекту?
- 3 Які є команди для роботи з проектом?

План самостійного вивчення теми № 2

з навчальної дисципліни "Об'єктно-орієнтоване програмування"

Тема	Робота з текстовим редактором. Компіляція, компоновка.
	Вихідний код
Мета	Ознайомитись з інструментарієм середовища візуального програмування, навчитись створювати консольний додаток, працювати з проектом
	<i>знати:інструментарій середовища візуального програмування</i>
	<i>вміти:використовувати інструментарій середовища візуального програмування</i>

Час – 3 години

Навчальні питання:

- 1 Інтегроване середовище розробки
- 2 Створення проекту консольного додатку
- 3 Робота з проектом консольного додатку

Методичні рекомендації

Самостійно вивчаючи запропоновану тему, студент повинен засвоїти такі основні положення:

<i>По-перше</i>	Інтегроване середовище розробки — це програмний продукт, який об'єднує текстовий редактор, компілятор, відладчик і довідкову систему
<i>По-друге</i>	Проект — це набір взаємопов'язаних вихідних файлів, компіляція і компонування яких дозволяє створити виконувану програму
<i>По-третє</i>	Пакет включає інструментарій, необхідний для роботи в середовищі програмування

Література

Довбуш, Г. Ф. Visual C++ на прикладах [Текст] / Г. Ф. Довбуш, А. Д. Хомоненко; под ред. А. Д. Хомоненко. – СПб. : БХВ-Петербург, 2007. -528 с.

Навчальні матеріали до самостійного вивчення теми

ВСТУП

Integrated Development Environment (інтегроване середовище розробки), або, скорочено, IDE — це програмний продукт, об'єднуючий текстовий редактор, компілятор, відладчик і довідкову систему. До складу Microsoft Visual Studio входить IDE Microsoft Visual C++.

1 Створення проекту консольного додатку

Будь-яка програма, що створюється в середовищі Visual C + +, завжди оформляється як окремий проект (project). Проект - це набір взаємопов'язаних вихідних файлів і, при необхідності, підключаємих (заголовних) файлів , компіляція і компонування яких дозволяє створити виконувану програму.

Є поняття робочої області проекту (project workspace). Робоча область може містити будь-яку кількість різних проектів, згрупованих разом для узгодженої розробки: від окремого додатка до бібліотеки функцій або цілого програмного пакета. Очевидно, що для вирішення наших навчальних завдань кожна програма буде втілюватися у вигляді одного проекту, тому робоча область проекту у нас завжди буде містити один проект.

Для створення нового проекту типу «консольний додаток» слід виконати дії, подібні наведеним далі.

1. Виберіть у рядку меню головного вікна команду File> New ...

2. У діалоговому вікні New виберіть вкладку Projects:

- виберіть тип Win32 Console Application;
- введіть ім'я проекту в текстовому полі Project Name, наприклад First;
- введіть ім'я каталогу розміщення файлів проекту в текстовому полі Location1 (якщо зазначений каталог відсутній, то його буде створено автоматично);
- клацніть лівою кнопкою миші на кнопці OK.

3. Після цього запускається так званий майстер додатків Application Wizard, який відкриває діалогове вікно Win32 Console Application - Step1 of 1 з пропозицією визначитися, який підтип консольного застосування вам потрібно створити. Виберіть тип An empty project; клацніть на кнопці Finish.

4. Після цього з'явиться вікно New Project Information зі специфікаціями проекту та інформацією про каталог, в якому буде розміщено створюваний проект. Клацніть на кнопці ОК.

Рекомендується для розміщення проектів виділити спеціальну папку, наприклад D: Student MyProjects.

3 Робота з проектом консольного додатку

Зберегти проект можна за допомогою команди File, Save All.

Виконати програму за допомогою команди Build, Execute First.exe.

Закрити проект за допомогою команди File, Close Workspace.

Компіляція, компоновка і виконання проекту можуть бути виконані або через меню Build головного вікна, або за допомогою кнопок панелі інструментів. Опишемо коротко основні команди меню Build:

- ° Compile — компіляція обраного файлу. Результати компіляції виводяться у вікно Output.

- ° Build — компоновка проекту. Компілюються всі файли, в яких відбулися зміни з моменту останньої компонування. Після компіляції відбувається збірка (link) всіх об'єктних модулів, включаючи бібліотечні, в результуючий виконуваний файл. Повідомлення про помилки компоновки виводяться у вікно Output. Якщо обидві фази компонування завершилися без помилок, то створений виконуваний файл з розширенням. Exe може бути запущений на виконання.

- ° Rebuild All — те ж, що і Build, але компілюються всі файли проекту незалежно від того, чи були в них зроблені зміни.

- ° Execute — виконання файлу, створеного в результаті компоновки проекту.

Visual C ++ дозволяє будувати проект або в відлагодчій конфігурації (Win32 Debug), або у випускній конфігурації (Win32 Release). Виберіть в меню Project пункт Settings ... Відкриється діалогове вікно Project Settings. Перевірте, яке значення встановлено у вікні комбінованого списку Settings For: .. Якщо це не Win32 Debug, то перейдіть на потрібне значення через команду меню Build, Set Active Configuration ...

Щоб закінчити роботу над проектом можна вибрати меню File, пункт Close Workspace або закрити програму Visual C ++.

Відкрити проект, над яким ви працювали раніше можна способами:

1. В Visual C++ виберіть меню File, пункт Open Workspace ..., знайдіть папку з вашим проектом, а в ній файл ProjectName.dsw, відкрийте файл, клацнувши мишею.

2. В Visual C++ виберіть меню File, пункт Recent Workspaces, в списку останніх файлів, з якими йшла робота, знайдіть файл ProjectName.dsw, клацніть мишею.

3. В провіднику знайдіть папку з вашим проектом, а в ній файл ProjectName.dsw, клацніть мишею.

Висновки

IDE Microsoft Visual C++ — це програмний продукт, який об'єднує текстовий редактор, компілятор, відладчик і довідкову систему.

Перевірена і налагоджена виконується програма може виконуватися на комп'ютері до тих пір, поки вона існує. Після запуску виконуваної програми для вирішення завдання завантажувач розміщує програму в оперативній пам'яті. Центральний процесор вибирає і виконує кожну інструкцій програми, зберігаючи нові значення даних по мірі виконання програми.

Пакет MVC++ включає інструментарій, необхідний для роботи в середовищі програмування.

Питання для самоконтролю знань студентів

1 Дати огляд інструментарію пакету IDE Microsoft Visual C++.

2 Як перебудувати проєкт і для чого?

3 Як скопіювати файл?

План самостійного вивчення теми № 3

з навчальної дисципліни "Об'єктно-орієнтоване програмування"

Тема	Обробка виключень
Мета	Ознайомитись з принципами обробки виключень
<i>знати: правила обробки виключень</i>	
<i>вміти: використовувати правила обробки виключень</i>	

Час – 2 години

Виключення, або виняткові ситуації (exception handling) представляють собою події, що виникають всередині програми і призводять до ненормальної її роботи або помилок часу виконання.

Винятки вбудовані в мову C++ для обробки помилкових ситуацій, які іншим чином оброблені бути не можуть. Це дозволяє автоматизувати велику частину коду для обробки помилок, для чого раніше було потрібно ручне кодування.

Обробка виключень

Основи обробки виключень полягають в тому, що коли програма зустрічає ненормальну ситуацію, на яку вона не розрахована, можна передати управління іншій частині програми, здатній впоратися з цією проблемою і продовжити виконання програми або завершити роботу.

Як правило, виняток можна викидати в наступних випадках:

- немає іншого способу повідомити про помилку (наприклад, в конструкторах, перевантажених операціях і т. д.);
- помилка невинна (наприклад, не вистачає пам'яті);
- помилка незрозуміла або несподівана, в зв'язку з чим її тестування вельми складно.

У C++ передбачено три ключових слова для обробки виняткових ситуацій:

- try (контролювати);
- catch (ловити);

– throw (формування).

Якщо блок try виявляє виключення всередині блоку програмного коду, відбувається програмне переривання і виконується така послідовність дій:

1 Програма шукає відповідний оброблювач виключення.

2 Якщо обробник знайдений, управління передається обробнику виключення.

3 Якщо обробник не знайдений, викликається функція terminate для завершення програми.

Виняткова ситуація, яка виникла, обробляється всередині блоку catch, який повинен слідувати безпосередньо за блоком try. Для кожного виключення необхідно передбачити свій обробник, інакше може виникнути ненормальне завершення програми.

Обробники винятків проглядаються по порядку і управління передається тому обробнику, тип аргументу якого збігається з типом, який викликав цей обробник виключення. Якщо в тілі обробника виключення не міститься операторів переходу, то виконання програми триває з точки, яка йде безпосередньо за останнім блоком catch.

Блоки try і catch мають формати, наведені в лістингах 1 та 2.

Лістинг 1 – Формат блоку обробки виключень

```
try {
    // Код, який генерує виключення
}
catch { Type X ) {
    // Оброблювач виключення X типу Type, яке могло бути раніше згенеровано
    // всередині відповідного блоку try
}
// Обробники інших винятків попереднього блоку try
catch (...) { // Обробник будь-якого виключення відповідного блока try
}
```

```

int _tmain() {
    cout << "Programme start\n";
    try { // start of try block
        cout << "Inside of try block\n";
        throw 13; // error generation
        cout << "This line will not receive the control ";
    }
    catch (int exception)
        cout << "Caught error with value; cout << exception << "\nn";
    }
    cout << "End of programme"; return 0;
}

```

Лістинг 2 – Формат блоку обробки виключень

```

int main () {
    try {
        // запис даних в файл
        ...
        if(!fp)
            throw "Can't open file\n";
        if(b==0) throw "Zero devide!";
        c = a/b;
        if(sprintf("Result of division:%f",c) {
            throw "Output error";
        }
        // обробник виключень
    }
    catch(char * s) {
        puts(s);
    }
    return 0;
}

```

Послідовність дій програми при виникненні помилки.

1. Код нормально виконується поза блоком try.
2. Управління переходить до блоку try.
3. Який-небудь вираз в цьому блоці призводить до виникнення помилки в функції.
4. Функція генерує виняток.
5. Управління переходить до блоку обробника помилок, який знаходиться відразу за блоком try.

У будь-якому вираженні блоку try може статися помилка, але тепер не потрібно турбуватися про отримання повертаємого значення для кожного з них, тому що це робиться автоматично.

Важливою проблемою, яку вирішує механізм винятків, є виявлення помилок в бібліотеках класів. У бібліотечної функції легко може виникнути помилка, але зазвичай зовсім неясно, де саме і що з нею робити. Крім того, треба враховувати, що алгоритми для бібліотеки були написані якимись іншими людьми, в інший час, тобто досить сильно можуть відрізнятися за концепцією від програми, яку ви збираєтеся писати. Що зобов'язана в будь-якому випадку бібліотечна функція зробити, так це повідомити про виниклу помилку в викликавшу її програму.

Механізм винятків включає в себе таку можливість, оскільки виключення передаються наверх по ієрархічному ланцюжку вкладених функцій до тих пір, поки не дійдуть до блоку, який перехопить помилку і обробить її. Вираз видачі виключення може входити в бібліотеку, але пастка може бути тільки в програмі, яка викликала і яка одна знає, як краще обробити помилку.

Якщо необхідно написати бібліотеку класів, слід подбати про те, щоб виключення видавалися в викликавшу програму з усіх функцій, в яких може статися щось неординарне. Якщо ж ви пишете програму, яка використовує бібліотеку, то потрібно передбачити блок повторних спроб і блок-пастку для будь-яких винятків, які може видавати бібліотека.

Виключення — це механізм обробки помилок, що виникають при некоректній роботі програм, написаних на C++, з використанням систематичного, об'єктно-орієнтованого підходу. Виняткові ситуації зазвичай виникають внаслідок помилкових виразів в блоках повторних спроб (try-блоках), які працюють з об'єктами деякого класу. Метод класу виявляє помилку і «генерує виняток», який ловиться програмою за допомогою класу, написаного в кодї обробника винятків відразу за блоком повторних спроб.

Питання та завдання до контролю знань студентів

Для узагальнення та перевірки засвоєного матеріалу на лекції

1. Для чого використовується обробка виключень?
2. Які ключові слова використовуються в блоці обробки виключень?

Література

Лафоре, Р. Объектно-ориентированное программирование в C++ [Текст] / Р. Лафоре. —
М. - СПб. : ПИТЕР, 2004.- 923 с.

Тема Інструментарі середовища візуального програмування. Створення візуального додатку. Робота з проєктом

Мета	Розглянути інструментарій середовища візуального програмування, кроки зі створення візуального додатку і роботу з проєктом на прикладі додатків типу Windows Forms
-------------	--

Загальне уявлення про Windows Forms. Windows Forms (Форми Windows) - це набір засобів для створення Windows-додатків, що виконуються в середовищі CLR (Common Language Runtime - загальномовне виконуюча середовище).

Форма - це вікно, що служить основою вікна програми або діалогового вікна, в яке можна додавати інші елементи управління, призначені для взаємодії з користувачем.

Стандартний пакет Visual C ++ 2005 поставляється зі стандартним набором більше 60 елементів управління, які можна застосовувати в формах. Багато стандартних елементів управління, такі як Button, що представляють кнопки, призначені для обробки клацань миші на них, або TextBox, які дозволяють вводити текст, реалізують прості функції взаємодії з користувачем. Інші є контейнерами - тобто можуть містити інші елементи управління.

Наприклад, елемент управління GroupBox може містити інші елементи управління на кшталт Button або TextBox, і його функція полягає просто в групуванні елементів управління з будь-якою метою і, можливо, в постачанні групи написом, яка буде відображатися всередині графічного інтерфейсу користувача.

Додатки Windows Forms

Програми, що використовують засоби Windows Forms.

Розглянемо створення графічного інтерфейсу користувача для додатків Windows Forms, як забезпечити підтримку взаємодії користувача з елементами управління інтерфейсу, а саме:

- як використовуючи можливості конструктора форм Form Design, інтерактивно будувати графічний інтерфейс користувача додатки;

- додавати в форму елементи управління;

- додавати до елементів управління обробники подій.

Форма і використовувані з нею елементи управління представлені класом C++ / CLI. Кожен клас має набір властивостей, які визначають поведінку і зовнішній вигляд елемента керування або форми. Наприклад, видимість елемента управління в додатку і можливість взаємодії з ним користувача визначаються установкою відповідних значень властивостей.

Визначення властивостей елемента управління може виконуватися інтерактивно при побудові графічного інтерфейсу користувача за допомогою засобів IDE (Integrated Development Environment - інтегроване середовище розробки). Установка значень властивостей може проводитися також під час виконання з допомогою додаються в програму функцій, або за допомогою коду, який додається в існуючі функції. Класи визначають також функції, що викликаються для виконання операцій з елементами управління.

Загальне уявлення про додатки Windows Forms. Одержимо уявлення про роботу вихідного коду програми Windows Forms. Використовуючи шаблон Windows Forms Application, створимо новий проєкт CLR. Панель Editor відобразить графічне представлення вікна програми, оскільки побудова графічного інтерфейсу користувача за допомогою програми Windows Forms виконується графічно. Навіть подвійне клацання на імені файлу Form1.h в панелі Solution Explorer (Провідник рішень) не приводить до відображення коду. Але його можна відкрити, клацаючи правою кнопкою миші у вікні Editor (Редактор) і вибираючи пункт View Code (Показати код) з контекстного меню.

Код визначає клас Form1, що представляє вікно програми. Він визначений у власному просторі імен, наведеному в лістингу 1.

Лістинг 1 – Приклад простору імен.

```
namespace Ex21_01 {
using namespace System;
```

```

using namespace System::ComponentModel;
using namespace System::Collections;
using namespace System::Windows::Forms;
using namespace System::Data;
using namespace System::Drawing;
// решта коду

```

Компіляція проекту призводить до створення нової збірки, код якої відноситься до простору імен Ex21_01, що збігається з ім'ям проекту. Простір імен дозволяє розрізняти типи з однаковими іменами в різних збірках, оскільки кожне ім'я типу уточнюється ім'ям конкретного простору імен.

Існують шість рекомендованих до використання просторів імен бібліотеки .NET, що охоплюють функціональні можливості, які, швидше за все, будуть потрібні в додатках, що наведено в таблиці 1.

Таблиця 1 – Простір імен бібліотеки .NET

Простір імен	Вміст
System	Цей простір імен містить класи, які визначають типи даних, що використовуються в усіх додатках CLR. Він містить також класи подій і обробників подій, винятки та класи, які підтримують функції загального застосування
System :: ComponentModel	Цей простір імен містить класи, які підтримують роботу компонентів графічного інтерфейсу користувача в додатках CLR
System :: Collections	Цей простір імен містить класи колекцій, призначені для всілякої організації даних, в тому числі класи визначення списків, черг і стеків
System :: Windows :: Forms	Це простір імен містить класи, які підтримують використання в додатку засобів Windows Forms
System :: Data	Це простір імен містить класи, які підтримують набір компонентів ADO.NET, який використовується для доступу та оновлення джерел даних

System :: Drawing	Це простір імен визначає класи, які підтримують основні графічні операції, подібні малювання на формі або компоненті
-------------------	--

Клас Form1 - похідний від класу Form, який визначений в просторі імен System :: Windows :: Forms. Клас Form представляє вікно програми або діалогового вікна, а клас Form1, який визначає вікно для простору імен Ex21_01, успадковує всі члени класу Form.

Розділ в кінці класу Form1 містить визначення функції InitializeComponent (). Ця функція викликається конструктором для визначення вікна програми і будь-яких компонентів, що додаються в форму. Згідно з наведеними коментарів, цей розділ коду не слід модифікувати за допомогою редактора коду, оскільки він оновлюється автоматично при інтерактивному зміні вікна програми.

При використанні коштів Form Design важливо дотримуватися рекомендацію цього коментаря і не змінювати вручну автоматично генерується код. В іншому випадку рано чи пізно це неминуче призведе до виникнення помилок. Звичайно, весь код для додатка Windows Forms можна створити з нуля, але застосування можливостей Form Design для інтерактивного створення графічного інтерфейсу користувача додатки значно прискорює роботу і знижує ймовірність помилок. Але потрібно знати, як працює цей код.

Спочатку код функції InitializeComponent () виглядає наступним чином:

```
void InitializeComponent (void) {
this-> components = gcnew System :: ComponentModel :: Container ();
this-> Size * System :: Drawing :: Size (300,300);
this-> Text * L "Form1";
this-> Padding = System :: Windows :: Forms :: Padding (0);
this-> AutoScaleMode = System :: Windows :: Forms :: AutoScaleMode
:: Font; }
```

Член components класу Form1 успадкований від базового класу, і його завдання-відстеження компонентів, поступово додаються в форму. Перший оператор зберігає в члені components дескриптор об'єкта Container, що представляє колекцію, яка зберігає компоненти графічного інтерфейсу користувача в списку.

Кожен новий компонент, який додається до форму за допомогою засобів Form Design, додається в цей об'єкт Container.

Зміна властивостей форми. Решта оператори функції InitializeComponent () визначають властивості об'єкта Form1. Жодне з цих властивостей не слід змінювати безпосередньо в коді, але їх значення можна вибирати за допомогою вікна Properties (Властивості) форми. Тому знову зверніться до вкладки Form1.h [Design] форми у вікні редактора і клацніть на ній правою кнопкою миші, щоб відкрити вікно Properties.

Щоб отримати уявлення про доступні можливості, має сенс переглянути список властивостей форми. Клацання на будь-якому з них призводить до відображення опису в нижній частині вікна.

Щоб змінити значення властивості, можна вибрати комірку в правій колонці. Властивості, зліва від яких відображений символ +, володіють кількома значеннями, і клацання на цьому символі відображає значення, щоб їх можна було змінювати окремо. Клацаючи на кнопці, можна також упорядкувати властивості за алфавітом, це полегшує пошук властивості з відомим ім'ям.

Висновки

При створенні проекту додатка створюється як вікно програми Windows Forms, побудоване на основі класу Form, так і весь код, що забезпечує відображення цього вікна програми.

Після створення проекту Windows Forms розробка програми зводиться до виконання чотирьох окремих операцій:

- інтерактивне створення графічного інтерфейсу користувача на вкладці Form Design (Конструктор форми), що відображається в панелі Editor (Редактор), шляхом вибору елементів управління у вікні Toolbox (Панель інструментів) і їх приміщення в форму. На цьому етапі можна також створювати додаткові вікна форм;

- зміна властивостей елементів управління і форм у вікні Properties (Властивості) відповідно до потреб додатка;

- обробники подій клацань для елементів управління можна створювати, двічі клацаючи на елементі управління на вкладці Form Design. У вікні Properties елемента управління в якості його обробника події можна також визначати існуючу функцію;

- для задоволення потреб програми можна змінювати і розширювати класи, автоматично створювані в результаті взаємодії з вкладкою Form Design.

Питання та завдання до контролю знань студентів

1 Для актуалізації опорних знань та перевірки попереднього матеріалу

1 Що собою представляють візуальні компоненти?

2 Як виглядає ієрархічна побудова класів компонентів?

2 Для узагальнення та перевірки засвоєного матеріалу на лекції

1 Що собою представляють засоби Windows Forms?

2 Що собою представляють простори імен бібліотеки .NET?

Література

1 Хортон, А. Visual C++ 2005: базовый курс. [Текст] / А.Хортон. – М. : ООО "И.Д. Вильямс", 2007. - 1152 с.

Тема Ієрархічна побудова класів компонентів

Мета Ознайомитись з ієрархічною побудовою класів компонентів

заняття

Візуальні середовища розробки спираються на бібліотеки компонентів. Компоненти, в свою чергу, являються класами і утворюють ієрархії класів на чолі з абстрактними класами, призначеними для угруповання найбільш загальних рис інтерфейсів.

Ієрархічна побудова класів компонентів

При створенні додатків на основі об'єктно-орієнтованого підходу використовуються компоненти бібліотеки MFC (Microsoft Foundation Classes). Відповідні додатки називають додатками MFC. Розглянемо їх для прикладу.

При створенні додатків MFC можна використовувати всілякі елементи управління графічного інтерфейсу користувача Windows: кнопки, перемикачі, меню та ін. Дамо коротку характеристику названої бібліотеці.

Фрагмент ієрархії класів бібліотеки MFC наведений на рисунку 1.

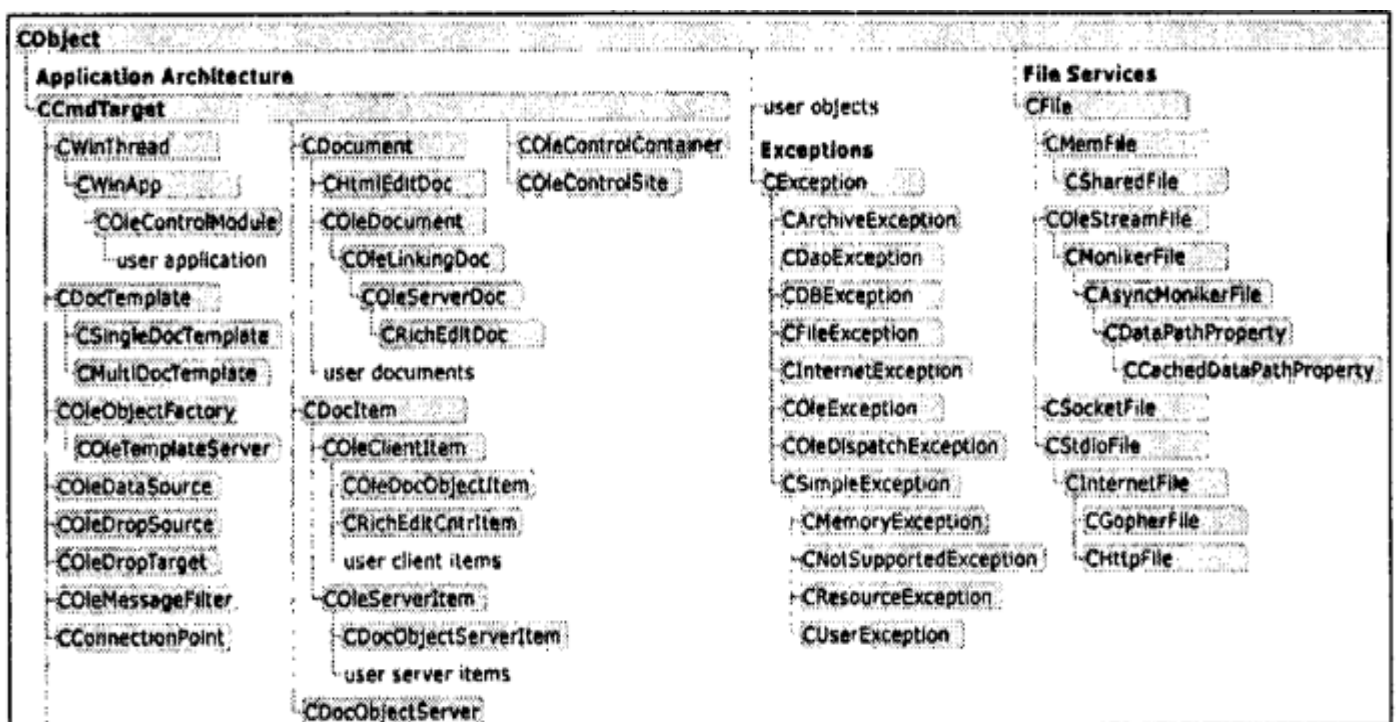


Рис. 1 – Фрагмент ієрархії класів бібліотеки MFC

Бібліотека MFC містить набір класів, створених для спрощення програмування додатків. При розробці бібліотеки ставилося завдання забезпечити мінімальний обсяг додаткового коду і швидкість виконання додатків, що використовують MFC, яку можна порівняти зі швидкістю виконання додатків, що використовують API Win32. Крім того, є можливість зручного виклику будь-яких функцій з API Win32.

Основним класом є клас CObject, від нього успадковуються всі інші класи. Всі вони розбиті на групи, зокрема в групі класів обслуговування файлів File Services основним є клас CFile.

Додатки MFC можуть створюватися з однодокументним (SDI, Single Document Interface) і багатодокументним (MDI, Multiple Document Interface) інтерфейсом. Вони можуть забезпечувати роботу з базами даних і роботу зі складними документами. Крім того, додатки можуть використовувати колективні бібліотеки DLL або статичні бібліотеки. Розглянемо склад найпростішого додатка MFC з однодокументним інтерфейсом.

Додаток SDI має меню, за допомогою якого користувач може відкрити один документ і з ним працювати. Майстром додатків MFC при настройках за умовчанням створюються п'ять класів.

Для прикладу MFCSDI ці класи мають такі імена:

- CAboutDlg - клас діалогового вікна About;
- CMFCSDIApp - клас для додатка, похідний від класу CWinApp;
- CMFCSDIDoc - клас документа, похідний від класу CDocument;
- CMFCSDIView - клас уявлення, похідний від класу CView;
- CMainFrame - клас фрейму.

У файлі заголовка важливу роль відіграє оголошення класу CMFCSDIApp, успадкованого від класу CWinApp. Останній клас включає в себе більшість функціональних можливостей програми.

Майстер додатків MFC згенерував кілька функцій для класу-спадкоємця, які перевантажують відповідні функції базового класу.

Майстер додатків MFC генерує текст методів класу CMFCSDIApp: конструктора, `initinstance` про та `OnAppAbout ()` - і розміщує його в файлі `MFCSDI.cpp`. Наведемо текст конструктора, який ініціалізує об'єкт класу, в лістингу 46.1.

Лістинг 46.1 – Текст конструктора, який ініціалізує об'єкт класу

```
CMFCSDIApp:: CMFCSDIApp ()
// TODO: додайте текст конструктора і
// виконайте ініціалізацію
// об'єкта в методі Initinstance.
```

Цей конструктор нічого не повертає і, по суті, не ініціалізує. Щоб повідомити програмі про виниклі при ініціалізації об'єкта проблеми, застосовується так звана двоетапна ініціалізація. Створюється функція ініціалізації зі стандартним ім'ям `InitInstance()`.

Фрагмент програмного коду наведений в лістингу 46.2.

Лістинг 46.2 – Фрагмент створеного програмного коду

```
class CMFCSDIApp : public CWinApp
{
public:
CMFCSDIApp () ;
// Перегрузка public:
virtual BOOL initInstance();
// Реализация
afx_msg void OnAppAbout(); DECLARE_MESSAGE_MAP()
};
extern CMFCSDIApp theApp;
```

Тут директива препроцесора `#pragma` служить для захисту заголовка. Вона гарантує, що файл заголовка не буде відкомпільований в процесі складання проекту двічі. Подвійне включення файлу заголовка досить характерно для програм на мові C++.

Вікно Windows належить одному з існуючих в системі класів, який повинен бути визначений до відображення вікна на екрані. Зазвичай клас конкретного діалогового вікна успадковується від базового класу CDialog бібліотеки MFC.

Після створення ресурсу діалогового вікна можна виконати створення класу вікна і зв'язати його з нашим діалоговим вікном.

Це дозволяє створити карту повідомлень для діалогового вікна і забезпечити потрібну обробку повідомлень, що генеруються за допомогою елементів управління, розташованих на цьому вікні.

Висновки

Кожен об'єкт-вікно використовує клас вікна і віконну процедуру обробки повідомлень з ім'ям AfxWndProc(). В додатках API Windows повідомлення призначаються віконним процедур обробки повідомлень за допомогою дескрипторів вікон hwnd.

У бібліотеці MFC-додатки є так звана карта дескрипторів, що складається з таблиці дескрипторів вікон і покажчиків об'єктів. Таким чином, головна програма додатку за допомогою названої інформації знаходить покажчик на об'єкт-вікно і викликає відповідну їй віртуальну процедуру цього об'єкта.

Потрібна реалізація названої процедури для різних елементів управління або вікон забезпечується на основі поліморфізму.

Питання та завдання до контролю знань студентів

1 Для актуалізації опорних знань та перевірки попереднього матеріалу

1 Які є види візуальних додатків в MVC++?

2 Що забезпечує API Windows?

2 Для узагальнення та перевірки засвоєного матеріалу на лекції

1 Що собою представляють візуальні компоненти?

2 Як виглядає ієрархічна побудова класів компонентів?

Література

1 Довбуш, Г. Ф. Visual C++ на примерах [Текст] / Г. Ф. Довбуш, А. Д. Хомоненко;
под

ред. А. Д. Хомоненко. – СПб. : БХВ-Петербург, 2007. -528 с.

2 Хортон, А. Visual C++ 2005: базовый курс. [Текст] / А.Хортон. – М. : ООО "И.Д.
Вильямс", 2007. - 1152 с.
