МІНІСТЕРСТВО ОСВІТИ УКРАЇНИ

Бердичівський політехнічний коледж

Контрольна робота

з дисципліни “Технологія розробки програмного забезпечення”

(варіант №20)

Виконав:

студент групи Пзс-503

Михайлус Михайло Геннадійович

Перевірив:

викладач

Тростянський Борис Геннадійович

м. Бердичів 2009 р.

Зміст

1. Контроль структури програми

2. Планування і складання розкладу по розробці програмних засобів

3. Інструментальні системи технології програмування

4. Практичне завдання

Список використаної літератури

Рецензія

## 1. Контроль структури програми

Для контролю архітектури програмного забезпечення використовується суміжний контроль і ручна імітація.

Суміжний контроль архітектури програмного забезпечення зверху - це її контроль розроблювачами зовнішнього опису, розроблювачами специфікації якості і розроблювачами функціональної специфікації. Суміжний контроль архітектури програмного забезпечення знизу - це її контроль потенційними розроблювачами програмних підсистем, що входять до складу програмного забезпечення відповідно до розробленої архітектури.

Ручна імітація архітектури програмного забезпечення виконується аналогічно ручної імітації функціональної специфікації, тільки метою цього контролю є перевірка взаємодії між програмними підсистемами. Так само як і у випадку ручної імітації функціональної специфікації програмного забезпечення повинні бути спочатку підготовлені тести. Потім група розроблювачів повинна для кожного такого тесту імітувати роботу кожної програмної підсистеми, що входить до складу програмного забезпечення. При цьому роботу кожної підсистеми імітує один який-небудь розроблювач (не автор архітектури), ретельно виконуючи усі взаємодії цієї підсистеми з іншими підсистемами (точніше, з розроблювачами, що їх імітують) відповідно до розробленої архітектури програмного забезпечення. Тим самим забезпечується імітаційне функціонування програмного забезпечення у цілому в рамках архітектури, що перевіряється.

## 2. Планування і складання розкладу по розробці програмних засобів

Управління розробкою програмного засобів - це діяльність, спрямована на забезпечення необхідних умов для роботи колективу розроблювачів програмних засобів, на планування і контроль діяльності цього колективу з метою забезпечення необхідної якості програмних засобів, виконання строків и бюджету розробки програмних засобів.

Хоча види діяльності по керуванню розробкою програмних засобів можуть бути дуже різноманітними в залежності від специфіки розроблювального програмних засобів і організації робіт з його створення, можна виділити деякі загальні процеси (види діяльності) по керуванню розробкою програмних засобів:

складання загального плану по розробці програмних засобів;

планування і складання розкладів по розробці програмних засобів;

керування витратами по розробці програмних засобів;

поточний контроль і документування діяльності колективу по розробці програмних засобів;

підбор і оцінка персоналу колективу розроблювачів програмних засобів.

Планування і складання розкладів по розробці програмних засобів - це діяльність, зв'язана з розподілом робіт між виконавцями і за часом їхнього виконання в рамках намічених термінів і наявних ресурсів

Планування і складання розкладів по розробці програмних засобів являє собою ітеративний процес, що закінчується тільки після припинення робіт із самого програмного проекту.

На початку цього опису оцінюються загальний термін розробки програмних засобів, використовувані штати виконавців, граничний бюджет і інші обмеження (умови) розробки. З урахуванням цього фіксуються початкові параметри проекту (його структура і розподіл функцій). Повинні бути також визначені "етапи розвитку проекту" і їхні терміни.

Далі починається ітераційний процес, основу якого складає повторюванні складання розкладів. Складання розкладу полягає у поділі всієї роботи, необхідної для виконання проекту, на окремі самостійно виконувані завдання;

у складанні сіткового графіка виконання завдань;

у складанні гістограми виконання завдань;

у розміщенні виконавців завдань.

При виділенні самостійних завдань для кожного з них оцінюється час його виконання і його залежність від інших завдань з погляду порядку виконання. Сітковий графік являє собою схему (мережа) шляхів виконання завдань із указівкою часу виконання кожного завдання і з розміщенням етапів розвитку проекту. У сітковому графіку повинний бути визначений критичний шлях, що представляє собою такий шлях завдань, сумарний час виконання яких є найбільшим. Гістограма виконання завдань містить для кожного завдання свою тимчасову смугу від моменту, коли виконання цього завдання може бути почато, і до моменту, коли виконання цього завдання повинне бути закінчено. У такій смузі фіксується як тривалість виконання самого завдання, так і можливий запас часу для завершення його виконання. Це дає можливість модифікувати план розвитку проекту у визначених рамках без зміни загальних термінів виконання проекту. При розміщенні виконавців оцінюється для кожного виконавця відповідність його кваліфікації і досвіду характеру пропонованої роботи. Особлива увага приділяється розміщенню виконавців завдань, що знаходяться на критичному шляху. Через деякий час (звичайно 2-3 тижня) після активізації процесів, зазначених у розкладі, виконується огляд (перегляд) ходу розвитку проекту і відзначаються виниклі протиріччя. З урахуванням цього виробляється перегляд (уточнення) параметрів проекту й оцінюється вплив змінених параметрів на розклад проекту. Якщо виявиться, що ці зміни збільшують час розробки програмних засобів, необхідно обговорити з замовником можливість зміни обмежень проекту і терміну його завершення. У тому випадку, коли замовник не може піти на придатні зміни, виробляється технічний перегляд проекту з метою пошуку альтернативних підходів до розробки програмних засобів.

## 3. Інструментальні системи технології програмування

Швидка розробка додатків (RAD - Rapid Application Development) ґрунтується на візуалізації процесу створення програмного коду. Розглянута технологія є інструментальним програмним забезпеченням, що надає програмістам засоби, які прискорюють розробку необхідного прикладного процесу, скорочують роботу з модифікації вже готової прикладної програми і внесенню в неї необхідних доповнень чи змін. З метою максимального спрощення вказаних дій використовуються графічні інструментальні засоби.

Не слід зводити RAD тільки до візуальної генерації користувальницького інтерфейсу. Можливості цієї технології набагато ширше простого набору процедур, що включають вставку керуючих елементів на форми, з наступною установкою їхніх властивостей. Засоби швидкої розробки додатків ґрунтуються на компонентній архітектурі. При цьому компоненти є об'єктами, що поєднують дані і методи, а також властивості. Властивості, з одного боку, дозволяють працювати з даними так само, як із членами класів, а з іншого боку, ховають за операціями читання/запису виклики методів, переводячи операції над об'єктами на більш високий рівень абстракції.

Компоненти можуть бути як візуальними, так і не візуальними; атомарними і контейнерними (такими, що містять інші компоненти); низькорівневими (системними) і високорівневими.

Візуальне проектування користувальницького інтерфейсу надає можливість вибору окремих компонентів з палітри з наступним розміщенням їх у потрібному місці. Процедура розробки інтерфейсу засобами RAD зводиться до набору послідовних операцій, що включають:

розміщення компонентів інтерфейсу в потрібному місці;

завдання часу їхньої появи на екрані;

настроювання зв'язаних з ними атрибутів і подій.

В ідеалі середовище візуальної розробки повинне дозволяти швидко перетаскувати компоненти за допомогою миші і задавати значення змінюваних параметрів. Ефективність візуального програмування визначається не стільки наявністю візуальних компонентів, скільки їхнім взаємозв'язком і взаємодією з традиційними засобами.

Інтегроване середовище розробки є засобом, за допомогою якого виконується проектування, налагодження, тестування і подальше поширення прикладних програм.

В даний час існує велика кількість засобів візуального програмування, заснованих на різних алгоритмічних мовах. Лідерами в розробці таких засобів є фірми Microsoft і Inprise (колишня Borland). Кожна з них надає кілька середовищ візуального програмування:

Microsoft - Visual Basic, Visual C++ і Visual J++;

Inprise - C++ Builder, Delphi і JBuilder.

Зробимо короткий огляд достоїнств і недоліків кожної з перерахованих середовищ проектування.

Visual C++ і C++ Builder

Системи Visual C++ і C++ Builder базуються на використанні мови програмування C++. Вони орієнтовані в основному на професійних розроблювачів програмного забезпечення.

Основними перевагами мови C++ є широка поширеність, присутність практично на всіх платформах і висока ефективність створених на її основі програм. З іншого боку, використання мови C++ вимагає досить високої професійної підготовки, що знижує швидкість розробки додатків. За деякими оцінками, швидкість розробки додатків мовою С в кілька разів менше, ніж при використанні більш простих мов Object Pascal і Visual Basic.

Delphi

Основу середовища проектування Delphi складає мова програмування Object Pascal. Останні версії цієї мови по своїх можливостях наближаються до C++. У той же час Pascal має більш простий і ясний синтаксис і на відміну від мови С є строго типізованою мовою. Це дозволяє мінімізувати число помилок і підвищити швидкість розробки додатків. Object Pascal - повноцінна об’єктно - орієнтована мова, що підтримує всі основні концепції ООП: інкапсуляцію, спадкування і поліморфізм. На відміну від C++ у Object Pascal не підтримується множинне спадкування. Хоча мову Object Pascal можна вважати більш простою, чим C++, проте для її ефективного використання потрібно відносно висока кваліфікація програміста.

Delphi має могутні засоби для розробки додатків, що керують базами даних як на основі власного процесора баз даних BDE, так і на основі технології ADO, підтримуваної Microsoft, BDE взаємодіє з базами даних через драйвери. Для більшості розповсюджених баз даних розроблені стандартні драйвери. Крім того, забезпечується можливість взаємодії з базою даних через драйвер ODBC, Доступ до SQL-серверів забезпечує система драйверів SQL Links, що дозволяє розробляти додатки для серверів Oracle, Informix, Sybase, DB2 і InterBase.

До недоліків Delphi можна віднести малу поширеність Object Pascal, який у даний час ніхто, крім Inprise, не підтримує.

Visual Basic

Система Visual Basic заснована однойменною мовою програмування, що мало чим відрізняється від мови Quick Basic. Visual Basic простіше, ніж Pascal, і, імовірно, тому він одержав досить широке поширення серед непрофесіоналів і починаючих програмістів. Visual Basic не можна вважати повністю об’єктно - орієнтованою мовою, тому що в ньому не підтримується принцип поліморфізму. Однак, незважаючи на усі свої обмеження, Visual Basic цілком придатний для створення досить складних додатків.

Інтерфейс середовища розробки Visual Basic не має принципових відмінностей від Delphi, а мова програмування синтаксично близька до Pascal. Тому досить легко перейти від однієї системи до іншої.

Створення додатків для обробки інформації в середовищі Visual Basic базується на використанні наданих їм спеціальних засобів, що включають:

об'єкти для доступу до даних - Data Access Object (DAO);

32-розрядний процесор даних - JET 3.0, що підтримує усі стандартні операції по створенню, зміні і видаленню таблиць, індексів і запитів. Форматування створюваної їм БД відповідає формату Access. JET 3.0 також забезпечує підтримку цілісності і перевірку змінюваних даних на рівні полів і записів. Для зміни даних JET дозволяє використовувати мову SQL:

Великою перевагою Visual Basic є те, що він підтримується фірмою Microsoft - безперечним лідером і фактичним монополістом в області розробки програмного забезпечення.

Visual J++ і JBuilder

Системи Visual J++ і JBuilder призначені для розробки додатків мовою Jawa. Дана мова з'явилася недавно (на початку 90-х років) і поки не дуже розповсюджена. Основне достоїнство Jawa - незалежність від платформи. Однак досягається це за рахунок того, що Jawa - інтерпретована мова. Тому швидкість виконання Java-додатку в 10-20 разів повільніше в порівнянні з відкомпільованими програмами.

## 4. Практичне завдання

З використанням засобів візуального програмування розробити програму для автоматичного розрахунку значень складної функції:



Приклад файлу форми Delphi6 для табулювання функції:

unit Func\_tab;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls, Buttons, Grids, Menus;

type

TForm1 = class (TForm)

Label1: TLabel;

Edit1: TEdit;

Label2: TLabel;

Edit2: TEdit;

Label3: TLabel;

Edit3: TEdit;

StringGrid1: TStringGrid;

BitBtn1: TBitBtn;

Label4: TLabel;

ListBox1: TListBox;

Memo1: TMemo;

BitBtn2: TBitBtn;

Label5: TLabel;

Label6: TLabel;

Label7: TLabel;

MainMenu1: TMainMenu;

N1: TMenuItem;

N2: TMenuItem;

N4: TMenuItem;

N5: TMenuItem;

N3: TMenuItem;

N7: TMenuItem;

N8: TMenuItem;

N9: TMenuItem;

BitBtn3: TBitBtn;

procedure BitBtn1Click (Sender: TObject);

procedure Edit1KeyPress (Sender: TObject; var Key: Char);

procedure Edit2KeyPress (Sender: TObject; var Key: Char);

procedure Edit3KeyPress (Sender: TObject; var Key: Char);

procedure Edit1Exit (Sender: TObject);

procedure Edit2Exit (Sender: TObject);

procedure Edit3Exit (Sender: TObject);

procedure BitBtn2Click (Sender: TObject);

procedure N4Click (Sender: TObject);

procedure N3Click (Sender: TObject);

procedure FormActivate (Sender: TObject);

procedure BitBtn3Click (Sender: TObject);

procedure N5Click (Sender: TObject);

procedure N7Click (Sender: TObject);

procedure N8Click (Sender: TObject);

procedure N9Click (Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form1: TForm1;

X,Xn,Xk,H: real; // Параметри табулювання

fname: String [25] ; //

f: textfile;

implementation

{$R \*. dfm}

// Повідомлення про помилку у завданні інтервалів табулювання

procedure P1;

begin

MessageDlg ('"Xп" не може бути більшим ніж "Хк". ' +#13

+'Введіть значення правильно. ', mtWarning, [mbCancel], 0);

Form1. Edit1. Text: ='';

Form1. Edit2. Text: ='';

end;

// Повідомлення про помилку у значенні кроку табулювання по відношенню до

// меж табулювання

procedure P2;

begin

MessageDlg ('Крок табулювання "H" не може приймати таких значень. ' +#13

+'Введіть значення правильно. ', mtWarning, [mbCancel], 0);

Form1. Edit3. Text: ='';

end;

// Повідомлення про помилку перевищення допустимої розмірності даних

procedure P3;

begin

MessageDlg ('Введене значення знаходться за межами допустимого. ' +#13

+'Введіть значення правильно. ', mtWarning, [mbCancel], 0);

end;

procedure P4;

begin

MessageDlg ('Треба ввести всі дані. ', mtWarning, [mbCancel], 0);

end;

// Процедура очищення даних у формі

procedure P5;

begin

Form1. Edit1. Text: ='';

Form1. Edit2. Text: ='';

Form1. Edit3. Text: ='';

Form1. Edit1. SetFocus;

Form1. Height: =167;

Form1. Position: =poScreenCenter;

Form1. Label4. Visible: =False;

Form1. Label5. Visible: =False;

Form1. Label6. Visible: =False;

Form1. Label7. Visible: =False;

Form1. StringGrid1. Visible: =False;

Form1. ListBox1. Items. Clear;

Form1. Memo1. Lines. Clear;

Form1. ListBox1. Visible: =False;

Form1. Memo1. Visible: =False;

end;

// Контроль введення даних у поля фории

procedure TForm1. Edit1KeyPress (Sender: TObject; var Key: Char);

begin

case Key of

'0'. '9',Chr (8):;

'-': if (pos ('-',Edit1. Text) = 0) and (length (Edit1. Text) = 0)

Then Key: = '-'

else Key: = Chr (0);

',': if pos (',',Edit1. Text) <>0

THen Key: = Chr (0);

else Key: = Chr (0);

end;

end;

procedure TForm1. Edit2KeyPress (Sender: TObject; var Key: Char);

begin

case Key of

'0'. '9',Chr (8):;

'-': if (pos ('-',Edit2. Text) = 0) and (length (Edit2. Text) = 0)

Then Key: = '-'

else Key: = Chr (0);

',': if pos (',',Edit2. Text) <>0

THen Key: = Chr (0);

else Key: = Chr (0);

end;

end;

procedure TForm1. Edit3KeyPress (Sender: TObject; var Key: Char);

begin

case Key of

'0'. '9',Chr (8):;

',': if pos (',',Edit3. Text) <>0

THen Key: = Chr (0);

else Key: = Chr (0);

end;

end;

procedure TForm1. Edit1Exit (Sender: TObject);

begin

If Edit1. Text='' Then Exit;

If (Abs (StrToFloat (Edit1. Text)) >100000) Then

begin

P3;

Edit1. Text: ='';

Edit1. SetFocus;

end;

end;

procedure TForm1. Edit2Exit (Sender: TObject);

begin

If Edit2. Text='' Then Exit;

If (Abs (StrToFloat (Edit2. Text)) >100000) Then

begin

P3;

Edit2. Text: ='';

Edit2. SetFocus;

end;

end;

procedure TForm1. Edit3Exit (Sender: TObject);

begin

If Edit3. Text='' Then Exit;

If (StrToFloat (Edit3. Text) >10000) Then

begin

P3;

Edit3. Text: ='';

Edit3. SetFocus;

end;

end;

// Основна процедура програми

Procedure TForm1. BitBtn1Click (Sender: TObject);

var

I,K: integer;

Y: array [0. .1000] of Real;

label L1;

begin

// Перевірка наявності правильних значень в полях введення і їх взаємної

// коректності, з виведенням відповдних повідомлень і формуванням переходів

IF (Edit1. Text = '') or (Edit2. Text = '') or (Edit3. Text = '') then

begin

P4;

Exit;

end;

IF Edit3. Text = '0' then

begin

MessageDlg ('Треба задати крок табулювання,'

+ #13 +' який має ненульове значення', mtWarning, [mbCancel], 0);

Edit3. Text: = '';

Edit3. SetFocus;

goto l1;

end;

Xn: =StrToFloat (Edit1. Text);

Xk: =StrToFloat (Edit2. Text);

H: =StrToFloat (Edit3. Text);

If Xk<Xn Then

begin

P1;

goto L1;

end;

If (H<=0) Or (H>=Abs (Xk-Xn)) Then

begin

P2;

goto L1;

end;

X: =Xn-H;

K: = Round ( (Abs ( (Xk-Xn)) /H));

If K>1000 Then

begin

MessageDlg ('Переповнення масиву даних. '

+#13 +'Треба зменшити інтервал або'

+#13 +' збільшити крок табулювання', mtWarning, [mbCancel], 0);

Edit1. Text: = '';

Edit2. Text: = '';

Edit3. Text: = '';

goto l1;

end;

// Фомування компонентів для виведення результатів

StringGrid1. RowCount: = K+2;

Form1. Height: =430;

Form1. Position: =poScreenCenter;

Label4. Visible: =True;

Label5. Visible: =True;

Label6. Visible: =True;

Label7. Visible: =True;

StringGrid1. Visible: =True;

Label7. Caption: ='у полі memo';

ListBox1. Items. Clear;

Memo1. Lines. Clear;

ListBox1. Visible: =True;

Memo1. Visible: =True;

StringGrid1. Cells [0,0]: ='X';

StringGrid1. Cells [1,0]: ='Y';

// Розрахунок і виведення результатів

For I: =0 to K do

begin

Y [I]: = (1+ln (2-Xn+H\*I)) / (1-Xn+H\*I+0.1);

// Наступний рядок забезпечує виведення результату

// з точністю до тисячних

Y [I]: = Round (Y [I] \*1000) /1000;

StringGrid1. Cells [0, I+1]: =FloatToStr (Xn+H\*I); // Виведення у таблицю

StringGrid1. Cells [1, I+1]: =FloatToStr (Y [I]);

ListBox1. Items. Add (FloatToStr (Xn+H\*I) +' '+FloatToStr (Y [i])); // Виведення у список

Memo1. Lines. Add (FloatToStr (Xn+H\*I) +' '+FloatToStr (Y [i])); // Виведення у поле Мемо

end;

l1:;

end;

// Запис результатів у файл

procedure TForm1. BitBtn2Click (Sender: TObject);

begin

ListBox1. Items. SaveToFile ('result. txt');

end;

// Збереження у файлі

procedure TForm1. N4Click (Sender: TObject);

begin

ListBox1. Items. SaveToFile (fname);

end;

// Зчитати з файла і вивести у поле Мемо із скриттям зайвих компонентів

procedure TForm1. N3Click (Sender: TObject);

begin

If FileExists ('result. txt') = False Then

Begin

MessageDlg ('Файла не існує', mtWarning, [mbCancel], 0);

Exit;

end;

Label7. Visible: =True;

Label7. Caption: = 'Результати зчитування з файлу';

Memo1. Lines. LoadFromFile ('result. txt');

Memo1. Visible: =True;

Label4. Visible: =False;

Label5. Visible: =False;

Label6. Visible: =False;

ListBox1. Visible: =False;

StringGrid1. Visible: =False;

Form1. Height: =430;

Memo1. SetFocus;

Form1. Position: =poScreenCenter;

end;

// Створення файлу з перевіркою його існування

procedure TForm1. FormActivate (Sender: TObject);

begin

fname: ='result. txt';

AssignFile (f, fname);

If FileExists ('result. txt') = False Then

begin

rewrite (f);

CloseFile (f);

end;

end;

// Очищення полів введення

procedure TForm1. BitBtn3Click (Sender: TObject);

begin

P5;

end;

procedure TForm1. N5Click (Sender: TObject);

begin

P5;

end;

// Вихід з програми

procedure TForm1. N7Click (Sender: TObject);

begin

Close;

end;

// Виведення довідки

procedure TForm1. N8Click (Sender: TObject);

begin

ShowMessage ('Михайлус Михайло Геннадійович' + #13 + ' студент групи Пзс-503');

end;

procedure TForm1. N9Click (Sender: TObject);

begin

ShowMessage ('Навчальна програма табулювання функції. ' + #13 +

' Версія 1.0');

end;

end.

## Список використаної літератури

1. "Требования и спецификации в разработке программ." М. Мир, 1984.
2. В. Турский. "Методология программирования".
3. Конспект лекцій з предмету.
4. Інтернет.

## Рецензія

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |