МІНІСТЕРСТВО ОСВІТИ УКРАЇНИ

Бердичівський політехнічний коледж

КОНТРОЛЬНА РОБОТА

з дисципліни “Технологія розробки програмного забезпечення”

(варіант №15)

Виконав: студент групи ПЗС-504

Томашов О.В.

Перевірив: викладач

Тростянський Б.Г.

м. Бердичів

2007 р.

Зміст

1. Методи розробки структури програми

2. Документація, що створюється і використовується в процесі розробки програмних засобів

3. Інструменти комп’ютерної підтримки розробки програмних засобів

4. Практичне завдання

Список використаної літератури

## 1. Методи розробки структури програми

У якості модульної структури програми прийнято використовувати деревоподібну структуру, яка відображує принцип спадаючого проектування. У вузлах такого дерева розміщаються програмні модулі, а спрямовані дуги (стрілки) показують статичну підпорядкованість модулів, тобто кожна дуга показує, що в тексті модуля, з якого вона виходить, мається посилання на модуль, у який вона входить. Іншими словами, кожен модуль може звертатися до підлеглих йому модулів, тобто виражається через ці модулі. При цьому модульна структура програми, у кінцевому рахунку, повинна включати і сукупність специфікацій модулів, що утворять цю програму. Специфікація програмного модуля містить:

синтаксичну специфікацію його входів, що дозволяє побудувати використовуваною мовою програмування синтаксично правильне звертання до нього (до будь-якого його входу),

функціональну специфікацію модуля (опис функцій, виконуваних цим модулем по кожному з його входів).

Функціональна специфікація модуля будується так само, як і функціональна специфікація ПЗ.

У процесі розробки програми її модульна структура може по-різному формуватися і використовуватися для визначення порядку програмування і налагодження модулів, зазначених у цій структурі. Найчастіше використовуються два методи: метод висхідної розробки і спадаючої розробки. Спочатку будується модульна структура програми у виді дерева. Потім по черзі програмуються модулі програми, починаючи з модулів самого нижнього рівня (листи дерева модульної структури програми), у такому порядку, щоб для кожного програмного модуля були вже запрограмовані всі модулі, до яких він може звертатися. Після того, як усі модулі програми запрограмовані, виконується їхнє почергове тестування і налагодження у такому ж (висхідному) порядку, у якому велося їхнє програмування. Такий порядок розробки програми на перший погляд здається цілком природним: кожен модуль при програмуванні виражається через уже запрограмовані безпосередньо підлеглі модулі, а при тестуванні використовує вже налагоджені модулі. Однак, сучасна технологія не рекомендує такий порядок розробки програми. По-перше, для програмування якого-небудь модуля зовсім не потрібно наявності текстів використовуваних їм модулів (для цього досить, щоб кожен використовуваний модуль був лише описаний в мірі, яка дозволяє побудувати правильне звертання до нього), а для тестування його можливо використовувані модулі заміняти їх імітаторами (заглушками). По-друге, кожна програма в якомусь ступені підкоряється деяким внутрішнім для неї, але глобальним для її модулів принципам реалізації, припущенням, структурам даних і т.п., що визначає її концептуальну цілісність і формується в процесі її розробки. При висхідній розробці ця глобальна інформація для модулів нижніх рівнів ще не ясна в повному обсязі, тому дуже часто приходиться їх перепрограмувати, коли при програмуванні інших модулів виробляється істотне уточнення цієї глобальної інформації (наприклад, змінюється глобальна структура даних). По-третє, при висхідному тестуванні для кожного модуля (крім головного) приходиться створювати ведучу програму, яка повинна підготувати для модуля, який проходить тестування, необхідний стан інформаційного середовища і зробити необхідне звертання до нього. Це приводить до великого обсягу "відладочного" програмування й у той же час не дає ніякої гарантії, що тестування модулів виконується саме в тих умовах, у яких воно буде виконуватися в робочій програмі.

## 2. Документація, що створюється і використовується в процесі розробки програмних засобів

При розробці ПЗ створюється і використовується великий обсяг різноманітної документації. Вона необхідна як засіб передачі інформації між розроблювачами ПЗ, як засіб керування розробкою ПЗ і як засіб передачі користувачам інформації, необхідної для застосування і супроводу ПЗ. На створення цієї документації приходиться велика частка вартості ПЗ:

Цю документацію можна розбити на дві групи:

Документи управління розробкою ПЗ.

Документи, що входять до складу ПЗ.

Документи управління розробкою ПЗ керують і протоколюють процеси розробки і супроводу ПЗ, забезпечуючи зв'язок усередині колективу розроблювачів ПЗ і між колективом розроблювачів і менеджерами ПЗ - особами, що керують розробкою ПЗ. Ці документи можуть бути наступних типів:

Плани, оцінки, розклади. Ці документи створюються менеджерами для прогнозування і керування процесами розробки і супроводу ПЗ.

Звіти про використання ресурсів у процесі розробки. Створюються менеджерами.

Стандарти. Ці документи вказують розроблювачам, якими принципами, правилами, угодами вони повинні користуватися в процесі розробки ПЗ. Ці стандарти можуть бути як міжнародними чи національними, так і спеціально створеними для організації, у якій ведеться розробка ПЗ.

Робочі документи. Це основні технічні документи, що забезпечують зв'язок між розроблювачами. Вони містять фіксацію ідей і проблем, що виникають у процесі розробки, опис використовуваних стратегій і підходів, а також робочі (тимчасові) версії документів, що повинні увійти в ПЗ.

Замітки і переписування. Ці документи фіксують різні деталі взаємодії між менеджерами і розроблювачами. Документи, що входять до складу ПЗ, описують програми ПЗ як з погляду їхнього застосування користувачами, так і з погляду їхніх розроблювачів і супровідників. Тут слід зазначити, що ці документи будуть використовуватися не тільки на стадії експлуатації ПЗ але і на стадії розробки для керування процесом розробки (разом з робочими документами). Ці документи утворять два комплекти з різним призначенням:

Користувальницька документація ПЗ (К-документация).

Документація по супроводу ПЗ (С-документація)

## 3. Інструменти комп’ютерної підтримки розробки програмних засобів

Специфікація якості визначає основні орієнтири (цілі), які на всіх етапах розробки ПЗ так чи інакше впливають на прийняття різних рішень, на вибір придатного варіанту розробки ПЗ. Однак кожен примітив якості має свої особливості такого впливу, тим самим, забезпечення його наявності в ПЗ може зажадати своїх підходів і методів розробки ПЗ чи окремих його частин. Суперечливість критеріїв якості ПЗ, а також і їхніх примітивів якості, що виражають: задовільне забезпечення одного якого-небудь примітива якості ПЗ може істотно ускладнити забезпечення деяких інших з цих примітивів. Тому істотна частина процесу забезпечення якості ПЗ складається з пошуку прийнятних компромісів. Ці компроміси частково повинні бути визначені вже в специфікації якості ПЗ: модель якості ПЗ повинна конкретизувати необхідний ступінь присутності в ПЗ кожного його примітива якості і визначати пріоритети досягнення цих ступенів. Забезпечення якості здійснюється в кожнім технологічному процесі: прийняті в ньому рішення в тім чи іншому ступені впливають на якість ПЗ у цілому. Зокрема і тому, що значна частина примітивів якості зв'язана не стільки з властивостями програм, що входять у ПЗ, скільки із властивостями документації. У силу відзначеної суперечливості примітивів якості дуже важливо дотримуватися обраних пріоритетів у їхньому забезпеченні. При цьому варто дотримуватися двох загальних принципів:

спочатку необхідно забезпечити необхідну функціональність і надійність ПЗ, а потім уже доводити інші критерії якості до прийнятного рівня їхньої присутності в ПЗ;

немає ніякої необхідності і, може бути, навіть шкідливо домагатися більш високого рівня присутності в ПЗ якого-небудь примітива якості, чим той, котрий визначений у специфікації якості ПЗ. З користувальницькою документацією зв'язані такі примітиви якості ПЗ, як документованість та інформативність. У зв'язку з цим варто звернути увагу на широко використовуваний у даний час підхід інформування користувача в інтерактивному режимі (у процесі застосування програм ПЗ). Таке інформування в багатьох випадках виявляється більш зручним для користувача, чим за допомогою автономної документації, тому що дозволяє користувачу без якого-небудь пошуку викликати необхідну інформацію за рахунок використання контексту її виклику. Такий підхід до інформування користувача є дуже перспективним.

Програмним шляхом реалізуються такі примітиви якості ПЗ як комунікабельность. Комунікабельность забезпечується відповідною реалізацією обробки виняткових ситуацій і створенням придатного користувальницького інтерфейсу. Виникнення виняткової ситуації в багатьох випадках означає, що виникла необхідність інформувати користувача про хід виконання програми. При цьому видавана користувачу інформація повинна бути простою для розуміння. Користувальницький інтерфейс представляє засіб взаємодії користувача з ПЗ. При розробці користувальницького інтерфейсу варто враховувати потреби, досвід і здібності користувача. Тому потенційні користувачі повинні бути задіяні у процесі розробки такого інтерфейсу. Великий ефект тут дає його прототипування. При цьому користувачі повинні одержати доступ до прототипів користувальницького інтерфейсу, а їхня оцінка різних можливостей використовуваного прототипу повинна істотно враховуватися при створенні остаточного варіанта користувальницького інтерфейсу. У силу великої різноманітності користувачів і видів ПЗ існує безліч різних стилів користувальницьких інтерфейсів, при розробці яких можуть використовуватися різні принципи і підходи. Однак наступні найважливіші принципи варто дотримувати завжди:

користувальницький інтерфейс повинний базуватися на термінології і поняттях, знайомих користувачу;

користувальницький інтерфейс повинний бути однаковим;

користувальницький інтерфейс повинний дозволяти користувачу виправляти власні помилки;

користувальницький інтерфейс повинний дозволяти одержання користувачем довідкової інформації: як по його запитам, так і генеруємим ПЗ.

В даний час широко поширені командні і графічні користувальницькі інтерфейси.

Командний користувальницький інтерфейс надає користувачу можливість звертатися до ПЗ із деяким завданням (запитом), що представляється деяким текстом (командою) на спеціальній командній мові (мові завдань). Достоїнствами такого інтерфейсу є можливість його реалізації на дешевих алфавітно-цифрових терміналах і можливість мінімізації необхідного від користувача введення з клавіатури. Недоліками такого інтерфейсу є необхідність вивчення командної мови і досить велика імовірність помилки користувача при завданні команди. У зв'язку з цим командний користувальницький інтерфейс звичайно вибирають тільки досвідчені користувачі. Такий інтерфейс дозволяє їм здійснювати швидку взаємодію з комп'ютером і надає можливість поєднувати команди з процедурами і програмами (наприклад, мова Shell операційної системи Unix).

Графічний користувальницький інтерфейс надає користувачу можливості:

звертатися до ПЗ шляхом вибору на екрані придатного графічного чи текстового об'єкта,

одержувати від ПЗ інформацію на екрані у виді графічних і текстових об'єктів,

здійснювати прямі маніпуляції з графічними і текстовими об'єктами, представленими на екрані.

Графічний користувальницький інтерфейс дозволяє

розміщати на екрані безліч різних вікон, у які можна виводити інформацію незалежно;

використовувати графічні об'єкти (піктограмами чи іконки), для позначення різних інформаційних об'єктів чи процесів;

використовувати курсор для вибору об'єктів (чи їхніх елементів), розміщених на екрані, який переміщається за допомогою клавіатури чи миші.

Достоїнством графічного користувальницького інтерфейсу є можливість створення зручної і зрозумілої користувачу моделі взаємодії з ПЗ (панель керування, робочий стіл і т.п.) без необхідності вивчення якої-небудь спеціальної мови. Однак його розробка вимагає великих витрат часу, порівнянних із витратами по створення самого ПЗ. Крім того, виникає серйозна проблема по перенесенню ПЗ на інші операційні системи, тому що графічний інтерфейс істотно залежить від можливостей (графічної користувальницької платформи), наданих операційною системою для його створення.

## 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 ('Томашов О.В. ' + #13 + ' студент групи ПЗС-504');

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. Б. Іванов “Дискретная математика. Алгоритмы и программы".
4. Конспект лекцій з предмету.
5. Інтернет.