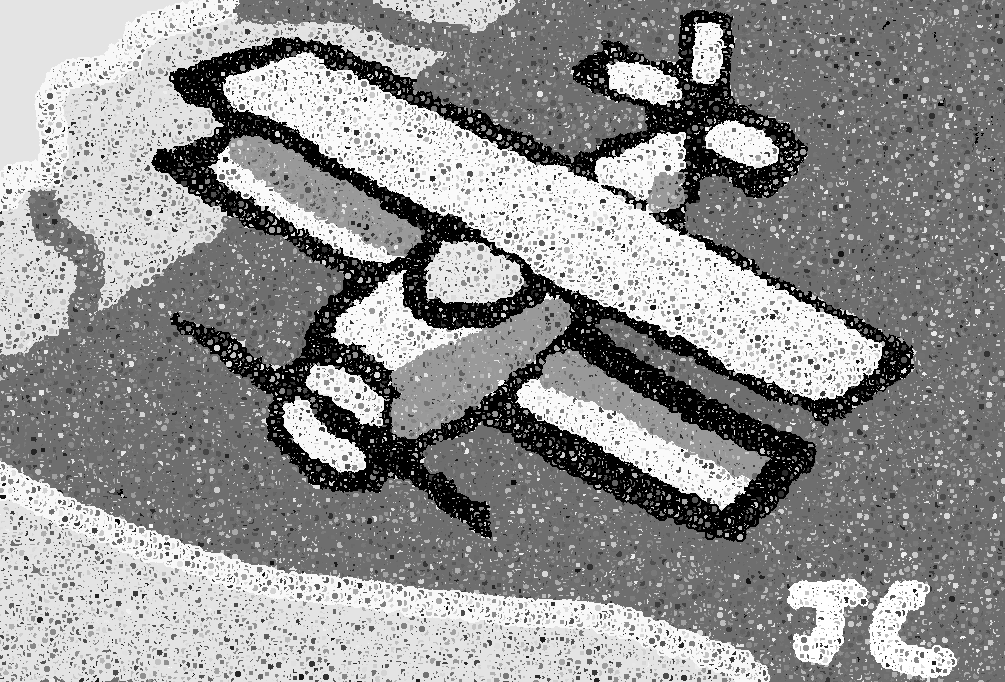
**Section 2: Tutorials**



**Week 1: Tutorial B**

**About the Paper**

**Objectives**

To introduce you to the organisation of COMPX101. You will also be introduced to some C# programming language.

**Reading**

Before coming to this tutorial, you should have read Chapters 1 and 2 of Gaddis.

**Summary**

1. Organisation of the paper.

2. Assignment statements.

3. Pop Quiz

**Exercises**

1. Discuss the paper organisation.

2. List five things you should do each week in/for COMPX101:

•

•

•

•

•

3. How do I get help if I am stuck?

4. Describe what happens in the following click event method:

private void buttonTest\_Click(object sender, EventArgs e)

{

buttonTest.Width = 100;

buttonTest.Height = buttonTest.Width;

buttonTest.BackColor = Color.Yellow;

}

5. Describe what the following lines of code will do?

buttonTest.Enabled = false;

buttonTest.Visible = false;

6. What are the values of big and small after the following code is executed:

int big = 10;

int small = 20;

big = small;

small = big;

7. What values do a, b and c hold after this code is executed:

int a = 100, b = 200, c = 300;

c = a;

a = b;

b = c;

8. **(Tutorial Handin Exercise)**   
Answer the surprise pop quiz the tutor will hand out.

**Week 2: Tutorial A**

**Using Visual Studio .NET and Designing Applications**

**Objectives**

To introduce you to the Visual Studio (VS) .NET environment and to the concepts in an object oriented / event driven environment. You will also be introduced to the C# programming language.

**Reading**

Before coming to this tutorial, you should have read Chapters 1 and 2 of Gaddis. You should also have read Appendix D: Introduction to Algorithms and the pseudo-code guide at:

[**http://users.csc.calpoly.edu/~jdalbey/SWE/pdl\_std.html**](http://users.csc.calpoly.edu/~jdalbey/SWE/pdl_std.html)

**Summary**

1. Concepts in an object oriented / event driven language.

2. Creating objects and using methods of objects.

3. Drawing graphics on a form using C#.

4. VS .NET properties and controls.

**Exercises**

1. Define the following Object-Oriented programming terms:

• Objects

• Classes

• Properties/Attributes

• Methods/Behaviours

2. Comment each line of code in the following click event method:

private void buttonDrawLine\_Click(object sender, EventArgs e)

{

//

//

Graphics paper = pictureBoxDisplay.CreateGraphics();

//

//

Pen pen1 = new Pen(Color.Blue, 5);

//

//

paper.DrawLine(pen1, 10, 30, 100, 200);

}

3. **(Tutorial Handin Exercise)** In the previous click event method create another pen of the colour **Red** with a thickness of **2**. Then draw a line from the coordinates 150,100 to 250,200.

4. Modify the code from the previous exercise so that it uses the same pen to draw the two lines of code in the correct colours and thickness.

5. What will the following code draw?

private void buttonDrawLine\_Click(object sender, EventArgs e)

{

Graphics paper = pictureBoxDisplay.CreateGraphics();

Pen pen1 = new Pen(Color.Blue, 5);

paper.DrawLine(pen1, 0, 0, pictureBoxDisplay.Width,

pictureBoxDisplay.Height);

}

Hint: The width and height properties of the picturebox will get the width and height of the picturebox and use those values in   
the code.

6. Complete the code below which will draw a line from the top left hand corner of the picturebox to the centre of the picturebox.

private void buttonDrawLine\_Click(object sender, EventArgs e)

{

Graphics paper = pictureBoxDisplay.CreateGraphics();

Pen pen1 = new Pen(Color.Blue, 5);

paper.DrawLine(pen1, 0, 0,

}

7. What will the following code draw?

Graphics paper = pictureBoxDisplay.CreateGraphics();

Pen pen1 = new Pen(Color.Red, 2);

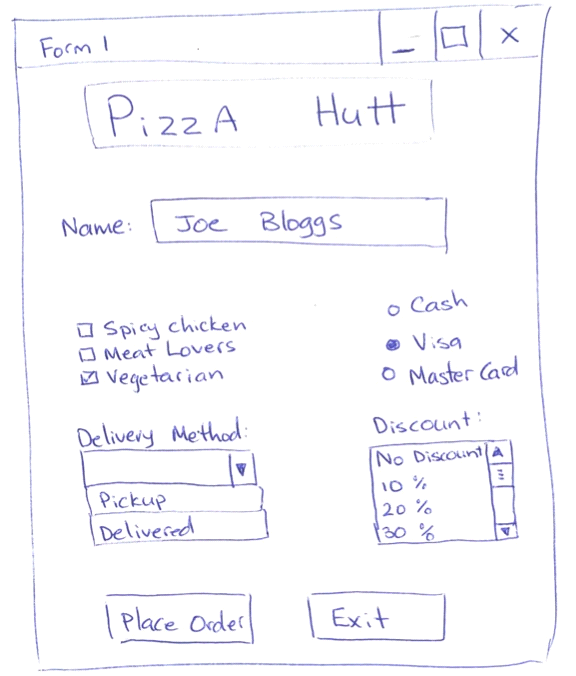
paper.DrawLine(pen1, 100, 50, 150, 100);

paper.DrawLine(pen1, 150, 100, 50, 100);

paper.DrawLine(pen1, 50, 100, 100, 50);

8. What do you need to do to move the shape in exercise 6 to a different location?

9. Complete the following paper prototype by adding a name to each control (except labels). The name of a control should start with what type of control it is, i.e. textBoxHours, checkBoxYes, buttonLogin, etc



10. For each control listed below, state what it is used for.

Label:

Textbox:

Checkbox:

Radiobutton:

Combobox:

Listbox:

Button:

**Week 2: Tutorial B**

**Using Visual Studio .NET and Designing Applications**

**Objectives**

To introduce you to user interface paper prototyping (storyboarding), documenting projects and changing the properties of controls within the C# code. To also introduce the concepts of algorithms and pseudo-code.

**Reading**

Before coming to this tutorial, you should have read Chapters 1 and 2 of Gaddis. You should also have read Appendix D: Introduction to Algorithms and the pseudo-code guide at:

[**http://users.csc.calpoly.edu/~jdalbey/SWE/pdl\_std.html**](http://users.csc.calpoly.edu/~jdalbey/SWE/pdl_std.html)**.**

**Summary**

1. What is a user interface paper prototype?

A sketch of the user interface drawn on paper.

2. Designing the interface.

username : textboxusername

password:textboxpassword [Hidden Characters]

LOGIN Button EXIT Button

3. Using Methods.

4. Writing pseudo-code to describe an algorithm.

**Exercises**

1. Draw a paper prototype for a login screen below. Think about what information you will need to get from the user, then what types of controls are to be used. Also state the names for the important controls.

2. The Karawhiua Kapahaka Club have decided to get their members to register themselves online. You have been asked to create a form which adds a person to the Clubs database system. The form should allow the user to enter the following data about themselves:

ID Number (automatically allocated non-editable)

Name of the person

Address

City

Birthdate (dd/mm/yyyy, restrict to 10 characters)

Phone Number

Email Address

Level (choose one of the options below):

Junior

Adult

Senior

Reasons for Joining (any combination of):

whanangatanga (social)

kaiwhakataetae (competitor)

kaiāwhina (supporter)

kaiako (teacher)

kaiwaiata (singer)

ringapuoro (musician)

The user should then be able to enter the appropriate data and then save the data to the system or cancel adding the new person.

Draw a paper prototype on the next page for the above specifications below. Think carefully about which controls you would use and the names of the important controls.

**Paper Prototype:**

3. What is an algorithm and what is pseudo-code?

Algorithm: A set of instructions performed in a particular order

Pseudo-code: The actual instructions written in english like text.

4. Which of the following are algorithms?

a) Assembly instructions-Algorithm

b) Dictionary

c) Safety Guidelines

d) Knitting Pattern - Algorithm

e) Recipe- Algorithm

f) Phone Book

g) Novel

h) 'Hokey Tokey' Song-Algorithm

5. **(Tutorial Handin Exercise)**   
Using pseudo-code, describe an algorithm to send a text message from your phone to a friend.

**Week 3: Tutorial A**

**Using Objects, Variables and Constants**

**Objectives**

To introduce you to using objects and variables in C# to store values while an application is running. You will also learn how to document a click event method.

**Reading**

Before coming to this tutorial, you should have read Chapter 3   
of Gaddis.

**Summary**

1. Data types in C#.

Specifies the type of information to store in the computers memory.

Int –whole numbers (45,-10)

Double-real numbers (45,-10.65)

Decimal- real numbers(very high precsion)

String-series of characters(“Mary”)

Bool-stores the value true or false

1. Declaring and naming your objects and variables.

Variable : a memory location that can be used to store a value and the value can be read or changed while the application is running.

Datatype name=initial\_value;

Int hoursworked=0;

3. Using Point and Rectangle objects.

4. Documenting your projects.

**Exercises**

1. What is a data type? List the commonly used datatypes in C#?

2. What is a variable? How do you declare a variable in C#?

3. Declare variables with default values to store the following information about a DVD for a retro movie rental shop: the title of the DVD, the year the DVD was released, the current rental price of the DVD and whether the DVD is currently on hire or available.

4. Using the variable names you declared in Ex 3, store the following information:

The DVD title is **Serenity**, it was released in **2005**, its current rental price is **$3.50** and it is currently hired so the variable should be set to **true**.

5. What does the following code do?

private void buttonDrawLine\_Click(object sender, EventArgs e)

{

Graphics paper = pictureBoxDisplay.CreateGraphics();

Pen pen1 = new Pen(Color.Blue, 5);

Point startPoint = new Point(10, 30);

Point endPoint = new Point(100, 200);

int x = 140;

int y = 50;

paper.DrawLine(pen1, startPoint, endPoint);

paper.DrawRectangle(pen1, x, y, 100, 50);

}

6. **(Tutorial Handin Exercise)**   
An application requires the user to enter the number of tickets required for a concert and a discount rate using textboxes. It will then calculate the cost of the tickets (1 ticket costs $20) and the discount amount and then display both values along with the final cost to the user using textboxes, formatted as currency. Below is the pseudo-code to do this but the steps are not in the correct order. Write the letters in the correct order of steps.

A Calculate the total cost of the tickets

B Display the cost of tickets without discount

C Declare variables

D Display the final cost

E Get the number of tickets

F Display the discount amount

G Calculate the final cost

H Get the discount rate

I Calculate the discount amount

7. Consider each line of code in the following click event method:

/// <summary>

/// Gets the length and width of a wall and calculates the area

/// of the wall. Then calculates the amount of paint required

/// in litres and then calculates the cost of the paint. All

/// the values are then displayed to the user.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void buttonCalculate\_Click(object sender, EventArgs e)

{

double wallLength = 0;

double wallHeight = 0;

double area = 0;

double paintRequired = 0;

decimal paintCost = 0;

//Get the wall length value from the textbox and convert

//it into a double value and store in a variable

wallLength = double.Parse(textBoxLength.Text);

//Get the wall height value from the textbox and convert

//it into a double value and store in a variable

wallHeight = double.Parse(textBoxHeight.Text);

//Calculate the area of the wall by multiplying the

//values in the two variables

area = wallLength \* wallHeight;

//Calculate the amount of paint required

paintRequired = area \* 1.5;

//Calculate the cost of the paint required as a

//decimal value

//Cast:treats anumeric value as another numeric datatype

//for this calculation only

paintCost = (decimal)paintRequired \* 3.5m;

//Display the area of the wall in a textbox unformatted

textBoxArea.Text = area.ToString();

//Display the paint required value in a textbox formatted

//to 3 decimal places

textBoxPaintRequired.Text = paintRequired.ToString("N3");

//Display the cost of the paint required in a textbox

//formatted as currency

textBoxPaintCost.Text = paintCost.ToString("c");

}

8. Using the example code in Ex 7 write the code based on the correct order for the algorithm given in Ex 6.

/// <summary>

/// This click event method will calculate the cost

/// of buying concert tickets.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void buttonCalculate\_Click(object sender, EventArgs e)

{

//Declare variables

Int numTickets=0;

Double discRate=0;

Decimal totalcost=;

Decimal discAmount=0;

Decimal finalcost=0;

//Get the number of tickets (textBoxNumTickets)

numTickets=int.Parse(textBoxNumTickets.Text);

discRate=double.parse(textBoxDiscRate.Text)

//Get the discount rate (textBoxDiscRate)

discRAte=double.Parse(textBoxDiscRAte.Text);

//Calculate the total cost of the tickets

Totalcost=numTickets\*20;

//Calculate the discount amount

discAmount=totalcost\*(decimal)discRate;

//Calculate the final cost

Finalcost=totalcost-discAmount;

//Display the cost of tickets without discount

textBoxTicketCost

//Display the discount amount

textBoxDiscAmount

//Display the final cost

textBoxFinalCost.Text=finalcost.ToString(“c”);

}

**Week 3: Tutorial B**

**Using Variables and Constants**

**Objectives**

To introduce the student to exception handling and why it is important. To also introduce the student to using constants to make the application easier to modify in the future. You will also learn about the scope of a variable or constant and how to clear the contents of a textbox.

**Reading**

Before coming to this tutorial, you should have read Chapter 3   
of Gaddis.

**Summary**

1. Exception handling.

Condition

1)if(condition){ action}

Else { }

If() {

}

Else{}

2)

Try{ } //exception occurs only then catch works

Catch{ --

Return;}

3) switch(condition)

{

Case:

Break;

Case

Break;

Default:

Break;

}

(|| -checks only if 1st one is incorrect

|-checks both condition

Precedence

++x or --y

2. Using constants.

3. Scope of a variable or constant.

4. Clearing the contents of a textbox and using the Focus method.

**Exercises**

1. Explain what the following code will do if the textbox is empty.

private void buttonTest\_Click(object sender, EventArgs e)

{

try

{

const decimal WAGE\_RATE = 25m;

int hours = 0;

decimal weeklyWage = 0;

hours = int.Parse(textBoxHours.Text);

weeklyWage = hours \* WAGE\_RATE;

MessageBox.Show("Wages to be paid is: " +

weeklyWage.ToString("c"));

}

catch (Exception ex)

{

Messagebox.Show(ex.Message);

}

}

2. Why is exception handling important? Is it ever acceptable to have the catch part of the Try…Catch empty?

Stop from crashing.

Do not leave it empty.

Always provide feedback to user.

3. Write down the result of evaluating each of the following expressions when the integer variable **length** has the value **5**. After each result, write down the C# type of the result.

a) 4 + length \* 3;->19,int

b) 4.6 + length \* 3; ->19.6, Double

c) 2 + length / 2; ->4,int

d) 2 + (double) length / 2; ->4.5,Double

e) "Size is " + length + 6; ->size is 56

"Size is " + (length + 6) ->size is 11

1. **(Tutorial Handin Exercise)** What is the value of x?

Modulus (Gives the remainder of a division0

a) int x = 10 % 2; ->

b) int x = 9 % 2;

c) int x = 27 % 6;

d) int x = 4 % 10;

e) int x = 0 % 4;

f) double x = 5/2;

5. Write the c# code to implement the following formulae:

a) Assume the variables named area and r have been declared.

area = **π**r2

area=Math.PI\*r\*r;

b) Assume variables a, b, c and d have already been declared as doubles:

a=(b+c)/2\*d;

c) Assume variables a and b have already been declared as doubles:

a=b\*b\*b\*b\*b;

a=Math.Pow(b,5);

6. What is a constant?

A variable whose initial values cannot be changed while the application is running. A read-only variable.

const datatype name = initial\_value;

cons tint DAYS\_IN\_WEEK=7;

7. Declare constants and change the code for the following:

a. the price of an item which is $125.99

const decimal price=125.99m;

totalCost = numItems \* price;

totalCost = numItems \* 125.99m;

b. the number of hours a person works per day which is   
 8 hours.

Const int hours\_per\_day=8;

dailyPay = payRate \* hours\_per\_day;

dailyPay = payRate \* 8;

c. the name of the retail store “JB Hifi”.

Const string store\_name=” JB Hifi”;

this.Text = "JB Hifi";

this.Text = store\_name

8. What does scope of a variable/constant mean? What is the difference between a variable with method scope (local variable) and variable with class scope (class variable)? What are advantages and disadvantages of both?

Scope:

What parts of our code can access a variable or constant

Method Scope:

When you declare a variable/constant in a method, only the code in the method can access it.

Advantages:

Restrict access to just the code that needs it.

Less error in your code.

Disadvantages:

Can’t share a value across methods.

Doesn’t remember its’s value when method finishes executing

Class Scope:

When a variable /constant is declared at the beginning of the form class , all code in the form classes can access it.

Advantages:

Can share a value across methods

Does remember its value when method finish because they get deleted when the application closes.

Disadvantages:

Because there is no restriction can lead to errors

Which scope level is best for variables?->method

Which scope level is best for constants?->class

9. What will the following code do?

//Clears the textbox

textBoxBoardsGST.Text = "";

textBoxPadsGST.Text = "";

// Clears the textbox

textBoxHelmsGST.Clear();

textBoxBoardsGST.Focus();

10. The variable **paintRequired** stores as a double the litres of paint required to paint a wall. Paint tins contain 5 litres of paint. Write the C# code to work out the number of tins of paint required for the wall, rounded up to a whole number.

*Hint: x = Math.Ceiling(y) gives the value of y rounded up to the nearest whole number. It returns a double, so you need to cast or convert it to an int.*

int numTins = (int).Math.Ceiling(paintRequired/5);

**Week 4: Tutorial A**

**Using Selection Structures**

**Objectives**

To introduce you to using the If – Then – Else selection structure to alter the flow of control in your code. You will also learn how to 'nest' selection structures and how to display information in a message window.

**Reading**

Before coming to this tutorial, you should have read Chapter 4   
of Gaddis. You should also read section 5.8 from Chapter 5 on generating random numbers.

**Summary**

1. The If – Then – Else statement.

2. Using relational operators.

3. Drawing filled rectangles on a form using SolidBrush objects.

4. Parsing strings safely.

**Exercises**

1. Draw arrows to show the order of execution when the condition is true and when it is false:

if (condition)

{

//Code

}

//Rest of code

2. Name the following relational operators:

< : lessthan

<= :lessthan or equalto

> : greater than

>= : greaterthan or equalto

== :equals

!= : Not equals

3. Write the first line of the following If statements:

a. If the variable **number** is greater than or equal to **10**

if(x>=10)

{

}

b. If the variable **x** is equal to the variable **y**

if(x==y)

{

}

c. If the control **textBoxName** has the string **"Grogu"** in it.

If(textboxname.text==”**Grogu”)**

**{**

**}**

d. If the variable **valid** is not equal to **true**

if(valid!=True) orif(valid==false)

{

}

e. If the string variable **level** is greater than the value **5**.

If(int.Parse(level)>5

{

}

4. **(Tutorial Handin Exercise)**   
Assume that the textboxes are empty. What will be displayed by the following two if statements?

int x = 3;

if (x > 5)

textBoxMessage1.Text = "Hello";

textBoxMessage2.Text = "Good Bye";

**Answer:**

Only print message Good Bye , as Hello is very next message after if statement

int x = 3;

if (x > 5)

{

textBoxMessage1.Text = "Hello";

textBoxMessage2.Text = "Good Bye";

}

**Answer:**

Will not print any message.

5. Assume that the textboxes are empty. What is displayed in the textbox when the following code is executed.

int x = 3;

if (x > 5);

{

textBoxMessage1.Text = "Hello";

textBoxMessage2.Text = "Good Bye";

}

Prints both message

6. What does the following click event method do:const int WIDTH = 50;

const int HEIGHT = 100;

private void buttonDraw\_Click(object sender, EventArgs e)

{

Graphics paper = pictureBoxDisplay.CreateGraphics();

**SolidBrush br = new SolidBrush(Color.Black);**

int x = 10;

int y = 10;

**paper.FillRectangle(br, x, y, WIDTH, HEIGHT);**

//Shift the x position to the right by the width

x += WIDTH;// x=x+width

//x += WIDTH+20;

**br.Color = Color.White;**

**paper.FillRectangle(br, x, y, WIDTH, HEIGHT);**

}

7. How would you modify the code so the bars are drawn starting at position 100,100?

X=100;

Y=100;

8. How would you modify the code to draw ellipses instead of rectangles?

FillEllispse()

9. Draw arrows to show the order of execution when the condition is true and when it is false:

if (condition)

{

//Code

}

else

{

//Code

}

//Rest of code

10. The **CalcCost** button for an application will calculate the cost of an order based on the number of units specified by the user. It will get the number of units from a textbox and if the number of units is greater than 0, then work out the cost of the order and then display the total cost to the user in another textbox, otherwise it will display an error message that the number of units is not valid and clear input and output textboxes and set the focus. The pseudo-code for this problem is given below:

TRY

GET number of units

IF number of units is > 0 THEN

Calculate cost of order

Display total cost of order

ELSE

Display error message

Clear number of units textbox

Clear total cost textbox

Set focus to number of units textbox

ENDIF

CATCH

Display error message

Clear number of units textbox

Clear total cost textbox

Set focus to number of units textbox

ENDTRY

Using the algorithm above, write the code for the **CalcCost** button.

const decimal ITEM\_PRICE = 20m;

private void buttonCalcCost\_Click(object sender, EventArgs e)

{

int numUnits = 0;

decimal orderCost = 0;

try

{

//Get the number of units from textBoxNumUnits

numUnits = int.Parse(numunit\_text.text);

//Check if the number of units is greater than 0

if(numUnits>0)

{

//Calculate the cost of the order

CostofOrder=numUnits\*ItemPrice;

//Display total cost of the order in textBoxTotalCost

textBoxTotalCost.text= CostofOrder.ToString(“c”);

}

else

{

//Display error message

MessageBox.Show(“ ---“);

//Clear textboxes

textBoxNumUnits.Text = “”;

textBoxTotalCost.Clear();

//Set focus to first textbox

textBoxNumUnits.Focus();

}

catch(

{

//Display error message

MessageBox.Show(

//Clear textboxes

textBoxNumUnits.Text = “”;

textBoxTotalCost.Clear();

//Set focus to first textbox

textBoxNumUnits.Focus();

}

}

**Week 4: Tutorial B**

**Using Selection Structures**

**Objectives**

To introduce you to the logical operators and the else if clause. You will also learn how to use the switch selection structure.

**Reading**

Before coming to this tutorial, you should have read Chapter 4   
of Gaddis. You should also read section 5.8 from Chapter 5 on generating random numbers.

**Summary**

1. Using logical Operators.

2. Using the else if clause.

**Exercises**

1. What are the logical operators and what is their order of precedence?

Logical Operators:

Not(!) –Highest precedence

And (&&) -2nd Highest precedence

Or(||) – Lowest precedence

2. Complete the following truth tables for the logical operators:

The truth table for NOT is a single variable table. It has two possible combinations. Either A is **false** or A is **true**.

|  |  |
| --- | --- |
| A | !(A) |
| false | true |
| true | false |

If(x>5) ->(

The truth table for AND and OR each have four possible combinations:

|  |  |  |
| --- | --- | --- |
| A | B | A && B |
| false | false | false |
| false | true | false |
| true | false | false |
| true | true | true |

|  |  |  |
| --- | --- | --- |
| A | B | A || B |
| false | false | false |
| false | true | true |
| true | false | true |
| true | true | true |

3. **(Tutorial Handin Exercise)** Use the following info to evaluate the expressions below:

int x = 5 , y = 3 , z = 1;

1. **if (x == 5 || y == 3) ->true**

b. **if (y == 2 || x >= 5) ->true**

c. **if (z == 1 && y > 3)->false**

d. **if (z == 1 && x >= 5)->true**

e. **if (x == 5 || y > 0 && z < 1)->true**

f. **if (!(x > 0) && y > 0)->false**

4.   
How can the following code be improved? Write the improved code.

a)

if (count1 == 0)

{

if (count2 == 0)

{

if (count3 == 0)

{

MessageBox.Show("You Win");

}

}

}

If(count1==0 && count2==0 && count3==0)

{

Print “Message” }

b)

if (i % 2 == 0)

{

sum += i;

}

//else

//{

// sum = sum;

//}

c) The code below will calculate a fine if the speed value is above 100k/h.

//fine=0;

if (speed > 100)

{

fine = (speed – 100) \* 10;

}

else //if (speed <= 100)

{

fine = 0;

}

1. What is wrong with the following code:

// create this at class scope level where the constants go

Random rand = new Random();

Graphics paper = pictureBoxDisplay.CreateGraphics();

Pen pen1 = new Pen(Color.Black, 2);

int colorNum = 0;

//colorNum = rand.Next(1,4);

//minimum 1 -> inclusive

//maximum 4 ->exclusive

//if no minimum the value is 0;

if (colorNum == 1)

{

pen1.Color = Color.Red;

}

else if (colorNum == 2)

{

pen1.Color = Color.Blue;

}

else

{

pen1.Color = Color.Green;

}

paper.DrawRectangle(pen1, 10, 10, 100, 100);

6. Write statements for the following:

a) Check if the value stored in the variable **numFlags** is between 1 and 4, inclusive.

If(numflags>=1 && numflags<=4)

b) Check if the value stored in the variable **numFlags** is *not* between 1 and 4.

If(!(numflags>=1 && numflags<=4))

If(numflags<1|| numflags<=4)

7. Write some C# code for the following: if **radioButton1** is selected then set the back colour of **labelInput1** to **Red** otherwise set it to **Green**.

If(radioButton1.Checked == true)

{

Label Input1.BackColor =Color.Red;

}

Else

{

LabelInput1.BackColor =Color.Green;

}

8. Write the code so that if the bool variable **on** is true then set it to false and if it is false then set it to true.

If(on==true)

{

On=false;

}

Else

{

On=true;

//On=!on;

}

**Week 5: Tutorial A**

**Using Repetition Structures**

**Objectives**

To introduce you to repetition structures to repeat (or loop) statements many times. You will learn about counter controlled repetition and two structures that can be used for repetition. .

**Reading**

Before coming to this tutorial, you should have read sections 5.1 up to an including section 5.5 of Chapter 5 of Gaddis.

**Summary**

**Repetition**

1. Counter controlled repetition.

2. The for loop.

3. The while loop.

**Exercises**

1. Use arrows to show the order of execution of the while loop below:

int i = 1;

int sum = 0;

while (i < 3)

{

sum += i;

i++;

}

2. Write a while loop for the following conditions:

a) repeat from 1 to 100, incrementing by 2

int i=1;

while(i<=100)

{

i+=2;

}

b) repeat from 10 to 1, decrementing by 0.5

double i=10;

while(i>=1)

{

i-=0.5;

}

3. Use arrows to show the order of execution of the for loop below:

int sum = 0;

for (int i = 1; i < 3; i++)

{

sum += i;

}

4. Work through the code below keeping a track of the variables to work out what will be displayed in the message box window:

string info = "a";

for (int i = 1; i < 4; i++)

{

info += info;

}

MessageBox.Show(info.ToString());

Inf0: “aaaaaaaa”

i=4;

5. **(Tutorial Handin Exercise)**  Write a for loop for the following conditions:

a) repeat from 1 to 100, incrementing by 2

for(int i=1; i<=100;i+=2)

{

// …

}

b) repeat from 10 to 1, decrementing by 0.5

for(double i=10; i>=1;i-=0.5)

{

//…

}

c) the variable **numStudents** contains the number of students in a class. Repeat from 1 up to and including the number of students, incrementing by 1 each time.

For(int i=1; i<= numstudents; i++)

{

}

6. Write the pseudo-code that will draw a row of bricks across the bottom of a picturebox. The user will type in the number of bricks into a textbox.

Declare Variables

TRY

Get the number of bricks

For each brick to draw

Draw bricks

Shift x to the right by the brick width

End for

CATCH

Display error message

Clear textbox

Set focus to textbox

ENDTRY

7. Write the code for your algorithm in Ex 6.

//The width of a brick

const int BRICK\_WIDTH = 40;

//The height of a brick

const int BRICK\_HEIGHT = 20;

//The number of bricks in a row

const int NUM\_BRICKS\_IN\_ROW = 10;

//Where to draw the graphics

Graphics paper = pictureBoxDisplay.CreateGraphics();

//Brush used to fill in the bricks

SolidBrush br = new SolidBrush(Color.SaddleBrown);

//Pen used to draw outline around the brick

Pen pen1 = new Pen(Color.Black, 2);

//x position of the current square

int x = 0;

//y position is the bottom of the picture box minus the

//height to set the y position of the top left hand corner

int y = pictureBoxDisplay.Height - BRICK\_HEIGHT;

//The number of Bricks/columns to draw

int numBricks = 0;

try

{

//Get the number of bricks from textBoxNumBricks

numBricks = int.Parse(textBoxNumBricks.Text);

//FOR each brick/column to draw

for(int col = 1; col<=numbricks;col++)

{

//Draw brick at current x and y position

paper.FillRectangle(br, x, y, BRICK\_WIDTH, BRICK\_HEIGHT);

paper.DrawRectangle(pen1, x, y, BRICK\_WIDTH, BRICK\_HEIGHT);

//Shift x to the right by brick width

X+=BRICK\_WIDTH;

//Pause the application

System.Threading.Thread.Sleep(500);

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

textBoxNumBricks.Clear();

textBoxNumBricks.Focus();

}

8. Complete the C# code below for the following pseudo-code.

Declare variables

TRY

Get max value from textbox

FOR each number from 1 to max inclusive

Calculate square of current number

Display square of number in console window

ENDFOR

CATCH

Display Error message

ENDTRY

For instance, if the user types in 5 in the textbox, the following output could be displayed in the console window.

Square of 1 is 1

Square of 2 is 4

Square of 3 is 9

Square of 4 is 16

Square of 5 is 25

int max = 0;

int square =0;

try

{

//Get the max value from textBoxMax

max = int.Parse(textBoxMax.Text);

//FOR each number from 1 to max inclusive

for(int i = 1; i<=max;i++)

{

//Calculate the square of the current number

square =i\*I;

//Display the number and the square of number

Console.WriteLine(“Square of “ + i+ “ is” + square);

}

}

catch (Exception ex)

{

//Display error message and clear textbox

MessageBox.Show(ex.Message);

textBoxMax.Clear();

textBoxMax.Focus();

}

**Week 5: Tutorial B**

**Using Repetition Structures**

**Objectives**

To practice using loop structures by solving small exercises and writing pseudo-code.

**Reading**

Before coming to this tutorial, you should have read sections 5.1 up to an including section 5.5 of Chapter 5 of Gaddis.

**Exercises**

1. Write the pseudo-code that gets the number of characters to draw from a textbox and then displays a pattern. The number represents the number of columns to draw. For instance, if the user enters 5 in the textbox the following pattern will be drawn in the console window:

=\*=\*=

Declare variables

TRY

Get the number of characters to draw

For each character to draw

IF character is an odd numbered one THEN

DISPLAY “=” IN CONSOLE WINDOW

Else

Display “\*” in console window

ENDIF

ENDFOR

Drop down to the next line in the console window

CATCH

Display error message

ENDTRY

2. Complete the C# code for ex 1 from your pseudo-code.

int size = 0;

try

{

//Get the size of the pattern from textBoxSize

size = 0;

try

{

Size =int.Parse(textBoxSize.Text);

//FOR each column/character to draw

for(int col = 1; col<=size;col++)

{

// Check if the character is an odd numbered one

If(col% 2 ==1)

{

Console.Write(“=”);

}

Else

{

Console.Write(“\*”);

}

//Move down to next line in the console window

Console.WriteLine();

}

catch (Exception ex)

{

//Display error message and clear textbox

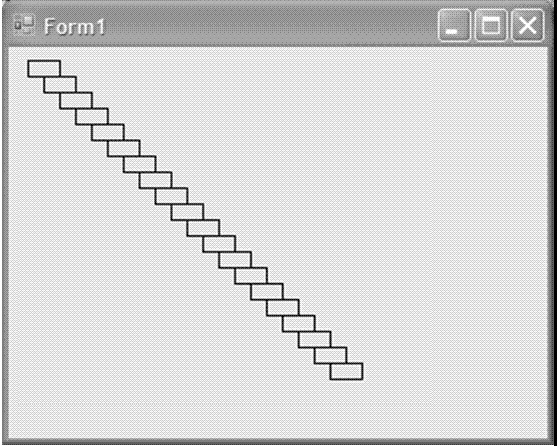
MessageBox.Show(ex.Message);

textBoxSize.Clear();

textBoxSize.Focus();

}

3. **(Tutorial Handin Exercise)**Write the pseudo-code that draws 20 steps of the staircase shown in the diagram.



4. Write the C# code for your algorithm in Ex 3.

Your code should draw exactly 20 steps. Each step is 20 pixels wide and 10 pixels high. The left-top corner of the top step is at position (10,10).

const int NUM\_STEPS = 20;

const int STEP\_WIDTH = 20;

const int STEP\_HEIGHT = 10;

Graphics paper = pictureBox1.CreateGraphics();

Pen pen1 = new Pen(Color.Black, 2);

int x = 10;

int y = 10;

For( int i=1;i<= NUM\_STEPS;i++)

{

//Draw step at current x and y position

Paper.DrawRectangle(pen1, x,y,STEP\_WIDTH,STEP\_HEIGHT);

//shift x to the right by half the width

X +=STEP\_WIDTH/2;

Y+=STEP\_HEIGHT;

}

5. Write the pseudo-code that will get the number of squares to draw and draw that number of boxes across the top of the picturebox, the first and last boxes should be LightBlue, the second and second to last boxes should be LightPink and all the rest should be in LightGreen.

Declare Variables

TRY

Get the number of squares

FOR each square to draw

IF square is first or last THEN

Set brush to light blue

ELSE IF square is second or second to last THEN

SET brush to light pink

ELSE

SET brush to light green

ENDIF

Draw square to current x and y position

Shift x to the right by the width

END FOR

CATCH

Display error message

ENDTRY

6. Write the code for your algorithm in the previous exercise.

const int SQUARE\_SIZE = 20;

const int NUM\_SQUARES = 10;

const int START\_LEFT = 10;

const int START\_TOP = 10;

int x = START\_LEFT;

int y = START\_TOP;

int numSquares = 0;

Graphics paper = pictureBox1.CreateGraphics();

Pen pen1 = new Pen(Color.Black, 2);

SolidBrush br = new SolidBrush(Color.Blue);

try

{

//Get the number of squares from textBoxNumSquares

numSquares = int.Parse(textBoxNumSquares.Text);

//FOR each square to draw

for(int col = 1; col<=numsquares;col++)

{

//Check if the current square is first or last one

If(col==1|| col ==numsquares)

{

Br.Color = Color.LightBlue;

}

//Check if the current square is 2nd or 2nd last

Else

If(col==2|| col ==numsquares-1)

{

Br.Color = Color.LightPink;

}

Else

Br.Color = Color.LightGreen;

Paper.FillRectangle(br,x,y,SQUARE\_SIZE,SQUARE\_SIZE);

Paper.DrawRectangle(pen,x,y, SQUARE\_SIZE,SQUARE\_SIZE);

X += SQUARE\_SIZE;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

**Week 6: Tutorial A**

**Problem Solving with loops**

**Objectives**

To practice problem solving by solving small exercises using nested loop structures and writing pseudo-code.

**Reading**

Before coming to this tutorial, you should have read sections 5.1 up to an including section 5.5 of Chapter 5 of Gaddis.

**Exercises**

1. Assume you are creating a proper console application. Write the code (on the next page) for the main method that will get a number from the console window which represents the number of rows and columns. Then it should draw the following pattern where every 3rd character is a dash, in the console window. If there are any errors then an error message is displayed and the application will end.

\*\*-\*\*-

\*\*-\*\*-

\*\*-\*\*-

\*\*-\*\*-

\*\*-\*\*-

\*\*-\*\*-

Complete the for and if statements:

static void Main(string[] args)

{

int size = 0;

try

{

Console.WriteLine(“Please enter size of pattern: “);

size = int.Parse(Console.ReadLine());

for (int row =1; row<= size; row++)

{

for (int col =1; col<=size; col++)

{

If(col%3==0)

(

{

Console.Write("-");

}

else

{

Console.Write("\*");

}

}

Console.WriteLine();

}

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

//Wait for a key press so user can see output

Console.ReadKey();

}

3. Extend the pseudo-code below to for a windows forms application that will get the size of a pattern which represents the number of rows and columns. It will check that the size is between the minimum and maximum number of rows and columns. If the size is valid it will then draw the correct number of rows of squares with every 4th column in LightBlue and all the rest in LightGreen.

Declare Variables

TRY

Get the size of the pattern

IF size of pattern is valid THEN

For each row to drwa

FOR each column to draw

IF current column is multiple of 4 THEN

Set brush colour to LightBlue

ELSE

Set brush colour to LightGreen

ENDIF

Draw square at current x and y position

Shift x to the right by width + gap

ENDFOR

Shift x to the right by height + gap

Shift x to the start of the row

ELSE

Display error message

ENDIF

CATCH

Display error message

ENDTRY

4. Complete the C# code from Ex 3 based on your pseudo-code.

//The width and height of a square

const int SIZE = 40;

//The gap between rows and columns

const int GAP = 5;

//Minimum and maximum size of pattern

const int MIN = 1;

const int MAX = 10;

//Where to draw the graphics

Graphics paper = pictureBox1.CreateGraphics();

//Brush used to fill in the bricks

SolidBrush br = new SolidBrush(Color.LightBlue);

//Pen used to draw outline around the brick

Pen pen1 = new Pen(Color.Black, 2);

//x position of the current square

int x = 0;

//y position of the current square

int y = 0;

//The size of the pattern

int size = 0;

try

{

//Get the size of the pattern textBoxSize

size = int.Parse(textBoxSize.Text);

//Check if the size is valid

if(

{

for(int col = 1; col <= size; col++)

{

if(col % 4 == 0)

{

br.Color = Color.LightBlue;

}

else

{

br.Color = Color.LightGreen;

}

paper.FillRectangle(br, x, y, SIZE, SIZE);

paper.DrawRectangle(pen1, x, y, SIZE, SIZE);

x += SIZE + GAP;

}

}

else

{

MessageBox.Show(“Invalid pattern size”);

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

5. **(Tutorial Handin Exercise)**   
The following code will draw even rows of squares in LightBlue and odd rows of squares in LightGreen. The code works correctly and produces the correct output but is inefficient. Discuss and write down (on the next page) why the code is inefficient and back up your answer with numeric data.

//The width and height of a square

const int SIZE = 40;

//The gap between rows and columns

const int GAP = 5;

//Size of the pattern

const int SIZE = 10;

//Where to draw the graphics

Graphics paper = pictureBox1.Creategraphics()

//Brush used to fill in the bricks

SolidBrush br = new SolidBrush(Color.LightBlue);

//Pen used to draw outline around the brick

Pen pen1 = new Pen(Color.Black, 2);

//x position of the current square

int x = 0;

//y position of the current square

int y = 0;

for(int row = 1; row <= SIZE; row++)

{

for(int col = 1; col <= SIZE; col++)

{

if(row % 2 == 0)

{

br.Color = Color.LightBlue;

}

else

{

br.Color = Color.LightGreen;

}

paper.FillRectangle(br, x, y, SIZE, SIZE)

paper.DrawRectangle(pen1, x, y, SIZE, SIZE)

x += SIZE + GAP;

}

Y += SIZE + GAP;

X = 0;

}

Answer:

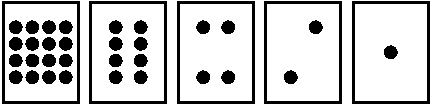
**Week 6: Tutorial B**

**Number Systems**

Did you know that computers use only zero or one? Everything that you see or hear on the computer — words, pictures, movies and even sound is stored just using these two numbers! This tutorial will cover the basics of binary numbers and how to convert between decimal and binary, and vice versa.

**Exercise 1:**

Take the 5 cards and arrange them in the following order:



* 1. Now flip the cards so exactly 5 dots show — keep your cards in the same order. Find out how to get 3, 12, 19. Is there more than one way to get any number?

No

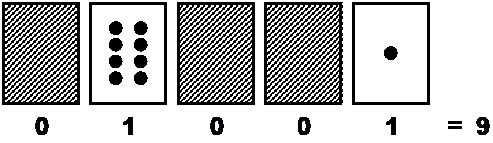
1.2 What is the biggest number you can make? 31 What is the smallest? 0 Is there any number that you can’t make between the smallest and biggest numbers?No

1.3 What do you notice about the number of dots on the cards? How many dots would the next card have if we carried on to the left? The next card after that?

Base 2 means we double the number of dots each time

**Exercise 2:**

The binary system uses zero and one to represent whether a card is face up or not. 0 shows that a card is hidden, and 1 means that you can see the dots. For example:



2.1 Can you work out what 10101 is? ->21

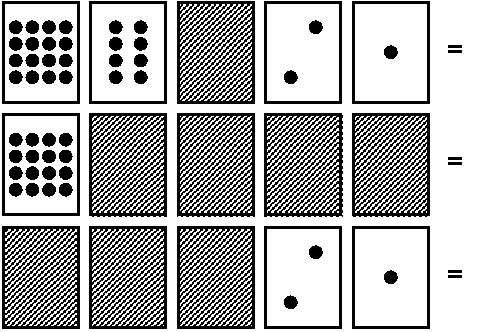
What about 11111-> 31

or 00000 ->0?

What day of the month were you born? Write it in binary.

01011

2.2 Write down the binary numbers the following cards represent:



00011011

10000

00011

2.3 Complete the following table, filling in the binary representation of each number.

|  |  |  |  |
| --- | --- | --- | --- |
| **decimal** | **binary** | **decimal** | **binary** |
| 0 |  | 16 |  |
| 1 |  | 17 |  |
| 2 |  | 18 |  |
| 3 |  | 19 |  |
| 4 |  | 20 |  |
| 5 |  | 21 |  |
| 6 |  | 22 |  |
| 7 |  | 23 |  |
| 8 |  | 24 |  |
| 9 |  | 25 |  |
| 10 |  | 26 |  |
| 11 |  | 27 |  |
| 12 |  | 28 |  |
| 13 |  | 29 |  |
| 14 |  | 30 |  |
| 15 |  | 31 |  |

**Exercise 3:**

Look at the binary cards again. If you look at the sequence carefully, you can find a very interesting relationship:

**1, 2, 4, 8, 16 ...**

3.1 Try adding 1 + 2 + 4 = ? What does it come to? Now try 1 + 2 + 4 + 8 = ? What happens if you add all the numbers up from all 5 cards?

When you add up all the dots from all the cards it is always 1 less than the next card

3.2 If you were going to make the next card in the sequence, how many dots would it have? What about the next card after that? What is the rule that you are following to make your new cards? As you can see, only a few cards will let you count up to very big numbers.

Num dots =2 ^ (n-1)

3.3 How high can you count using your fingers. If you answered 10, you would be wrong. If you use the binary system and let each finger on one hand represent one of the cards with dots you can count from 0 to 31. That’s 32 numbers. (Don’t forget that zero is a number too!)

Bit: an individual 1 or 0

Byte: 8 bit binary number

(256 different values, 0-255)

Kilobyte: 1024 byte

Megabyte: 1024 kilobyte

Gigabyte: 1024 Megabyte

Try counting in order using your fingers. If a finger is up, that is one, and if it is down, it is zero. You can actually get from 0-1023 if you use both hands! That’s 1024 numbers!

If you had really bendy toes, you could get even higher. If one hand can be used to count 32 numbers, and two hands can count to 32 x 32 = 1024 numbers, what is the biggest number that Miss Flexi-Toes can reach?

How many values can you represent with a byte (8 bits)?

How many values can you represent with 64 bits?

**Exercise 4:**

**(Tutorial Handin exercise)**   
Another interesting property of binary numbers is what happens when a zero is put on the right hand side of the number. If we are working in base 10, when you put a zero on the right hand side of the number, it is multiplied by 10. For example 9 becomes 90, 30 becomes 300.

But what happens when you put a 0 on the right of a binary number? Try this:



Make up some other examples to test your hypothesis. What is the rule? Why do you think this happens? What happens if you add a 0 to the left of a binary number?

Doubles the number

Do nothing

**Exercise 5:**

**Why Binary imprtant**

**The binary number system matches the stats of the disgital circuits inside the computer hardware**

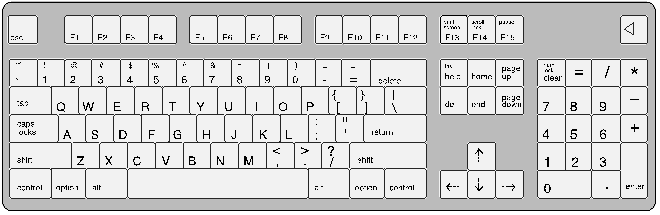
**The computer store only binary number**

**All characters are stored as ASCII numbers**

**ASCII has 127 characters -7 bits**

Each of the cards we have used so far represents a “bit” on the computer. If you start at a =1 and number each letter though to z = 26, our alphabet code can be represented using just five cards or “bits”. However, a computer has to know whether letters are capitals or not, and also recognize digits, punctuation and special symbols such as $ or ~.

5.1 Look at the keyboard on the next page and work out how many characters a computer has to represent. So how many bits does a computer need to store all the characters?



Most computers today use a representation called ASCII (American Standard Code for Information Interchange), which is based on using this number of bits per character, but some non-English speaking countries have to use longer codes.

**Explanation: Comparing the Decimal and Binary Number Systems**

(1) In the decimal, or base 10, number system, there are ten symbols or digits, 0, 1, 2, ... 9. The binary system, which is base 2, there are only two symbols or digits, 0 and 1.

(2) Both are positional number systems, that is the value of each digit is determined by its position in the number. For example, the number 256210 can be interpreted as:

= (2\*103) + (5\*102) + (6\*101) + (2\*100)

= 2000 + 500 + 60 + 2

= 256210

Note that the digit 2 has a different interpretation on its two occurrences in the number depending on its position. (Some number systems, such as the Roman number system, are not strictly positional — some symbols have values in their own right. i.e. X is always 10, L is always 50, etc. ).

Similarly the number 1011012 can be interpreted as:

= 1\*25 + 0\*24 + 1\*23 + 1\*22 + 0\*21 + 1\*20

= 1000002 + 02 + 10002 + 1002 + 02 + 12 = 1011012

= 3210 + 010 + 810 + 410 + 010 + 110 = 4510

(3) The base of the decimal number system is 10. In binary, the base is 2.

(4) Just as when we are counting in base 10 and reach the last single digit (9) we next go to 0 with a carry of 1 to the next highest position, e.g.

910 + 1 = 1010,

1910 + 1 = 2010,

99910 + 1 = 100010

then in the binary system, when we reach the last single digit (1) we next go to the 0 with a carry of 1 to the next highest position, e.g.

12 + 1 = 102

112 + 1 = 1002

1112 + 1 = 10002

(5) Fractional parts in both decimal and binary can be represented using a decimal point:

e.g. 5.62510 is

= 5\*100 + 6\*10-1 + 2\*10-2 + 5\*10-3

= 5 + .6 + .02 + .005

e.g. 101.1012 is

= 1\*22 + 0\*21 + 1\*20 + 1\*2-1 + 0\*2-2 + 1\*2-3

= 1\*4 + 0\*2 + 1\*1 + 1\*0.5 + 0\*0.25 + 1\*0.125

= 4 + 0 + 1 + 0.5 + 0 +0.125

= 5.62510

**The Hexadecimal number system (base 16)**

(1) There are 16 symbols or digits, 0, 1, ... 9, A, B, C, D, E, F

(2) It is a positional number system, e.g. 3FA16 can be interpreted as

= 3162 + F161 + A160

= 3256 + 1516 + 101

= 768 + 240 + 10

= 101810

**Number Systems in General**

In general, the interpretation of ‘abcdefg’ in a base ‘n’ positional number system is:

a\*n4 + b\*n3 + c\*n2 + d\*n1 + e\*n0 + f\*n-1 + g\*n-2

where a, b, c, d, e, f and g represent symbols or digits of the number system.

**Conversion between bases**

(1) To convert any base to base 10, expand the positional notation as above:

17316 = 1 \* 162 + 7 \* 161 + 3 \* 160

= 1 \* 256 + 7 \* 16 + 3

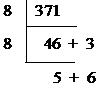
= 256 + 112 + 3 = 37110

5638 = 5 \* 82 + 6 \* 81 + 3 \* 80

= 5 \* 64 + 6 \* 8 + 3

= 320 + 48 + 3 = 37110

(2) Conversion from base 10 to any other base is best done by repeated division. e.g. 37110 = (?)8

 i.e. 5638 = 37110

**Exercise 6:**

6.1 Convert the following decimal (base 10) numbers to their binary (base 2) equivalents.

305 165

0 1024

6.2 Convert the following binary (base 2) numbers to their decimal (base 10) equivalents.

10112 10000012

101112 100012

6.3 Convert the following decimal (base 10) numbers to their hexadecimal (base 16) equivalents.

31 64

325 360

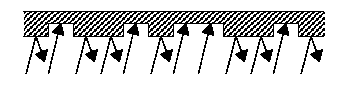
**Remarks:**

Computers today use the binary system to represent information. It is called binary because only two different digits are used. It is also known as base two (humans normally use base 10). Each zero or one is called a bit (binary digit). A bit is usually represented in a computers main memory by a transistor that is switched on or off, or a capacitor that is charged or discharged.

When data must be transmitted over a telephone line or radio link, high and low-pitched tones are used for the ones and zeros. On magnetic disks (floppy disks and hard disks), bits are represented by the direction of a magnetic field on a coated surface, either North-South or South-North.



CD-ROMs store bits optically — the part of the surface corresponding to a bit either does or does not reflect light.



One bit on its own can’t represent much, so they are grouped together in groups of eight, which can represent numbers from 0 to 255. A group of eight bits is called a byte.

The speed of a computer depends on the number of bits it can process at once. For example, a 32-bit computer can process 32-bit numbers in one operation, while a 16-bit computer must break 32-bit numbers down into smaller pieces, making it slower.

Ultimately bits and bytes are all that a computer uses to store numbers, text and all other information. In the next few concept tutorials we will see how other kinds of information can be represented on a computer.

**Week 7: Tutorial A**

**Methods**

**Objectives**

To introduce you to creating methods which can be called from other methods or click event methods. To show you how to pass information to methods and how to create and use methods which return a value. You will also learn how to document methods appropriately.

**Reading**

Before coming to this tutorial, you should have read Chapter 6   
of Gaddis. You should also have read the section on Menus in Appendix B of Gaddis (pages 739 – 745). Also watch the Panopto recording on creating menus for a more visual guide.

**Summary**

1. Creating methods.

2. Passing information by value.

3. Creating methods which return a value.

**Exercises**

1. What is the general template for writing a method?

Private return-datatype name(parameters)//method header

{//code-body

}

Parameters :special variables that will store values passed to the method

All parameters variables are local to the method

2. Write a method that does not return a value and is not passed any values called **DrawSquare** that draws a square with the top left-hand corner at 50, 50 with a side length of 100.

Private void DrawSqauare()

{

Graphics paper =pictureBoxDisplay.CreateGraphics();

Br= new SolidBrush.color

Paper.FillRectangle(Br,50,50,100,100);

}

3. Write the code to call the method from a click event method.

Private void Button\_click( obj sender, Event e)

{ DrawSquare(); }

4. Extend the method so that the x and y position of the square is passed to the method and used when drawing it.

Paper.FillRectangle(Br,x,y,100,100);

5. Change the code in the click event method to create two variables, x and y with the values 10 and 20 respectively. Call the method to draw the square at position 10, 20.

Int xpos =10;

Int ypos =20;

6. Extend the method so that the graphics object is passed to the method as well as the side length of the square and the colour the square will be drawn in.

Private void DrawSquare(int x, int y, Graphics pape, int length, Color sColor)

{

SolidBrush br = new SolidBrush(sColor);

Paper.FillRectangle(br,x,y,length,length);

}

7. Change the click event method to call the method correctly.

DrawSquare(xpos,ypos,canvas,size,Color.Blue);

8. **(Tutorial Handin Exercise)**   
Extend the click event method so that it draws 5 blue squares across the picturebox with a gap of 5 pixels between each square using the method you created.

For(int i=1;i<=5;i++)

{

DrawSquare(xpos,ypos,canvas,size,Color.Blue);  
xpos+=size+5;

}

9. Write the method header for the following windows forms application methods and an example of how to call them:

a) A method called **DisplayInterestTable** which is passed a deposit amount, an interest rate and the number of years for a term investment.

Private void DisplayInterestTable(decimal deposit,double rate, int numyear)

{

}

b) A method called **DrawTriangle** which is passed a Graphics object, three point objects for the corners of the triangle and the colour the triangle will be drawn in.

private void DrawTraingle(Graphics paper, Point c1, Point c2, Point c3, Color tColor)

{

}

**Week 7: Tutorial B**

**Methods continued**

**Objectives**

To allow you to work through more complex examples of   
using methods.

**Reading**

Before coming to this tutorial, you should have read Chapter 6   
of Gaddis. You should also have read the section on Menus in Appendix B of Gaddis (pages 739 – 745). Also watch the Panopto recording on creating menus for a more visual guide.

**Exercises**

1. Write the code to call the method below passing it a base of 10.5 and height of 5.5 and display the area in a message window.

//return te area of a triangle

private double CalculateTriangleArea(double baseValue,

double height)

{

double area = 0;

area = (baseValue / 2) \* height;

return area;

}

Private button1\_click (object sender, EventArgs e)

{

Double tArea=0;

tArea = CalculateTriangleArea(10.5,5.5);

messagebox.show(“Area:” +tArea);

}

2. Write method headers for the following:

a) A method called **CalculateVolume** which is passed the radius and height of a cylinder and returns back the volume of the cylinder. Assume all values are double values.

//Private double CalculateVolume(double radius, double height);

b) A method called **CheckDeposit** which is passed a deposit amount and returns back true if the deposit value is valid or false if it is not valid.

Private bool CheckDeposit(decimal Deposit)

{

}

3. Write the pseudo-code for a method that will convert an integer value into a binary number string. The pseudo-code should follow the division method of conversion covered in the tutorial.

METHOD ConvertToBinary (number) returning string

Declare variables

While number>0

the remainder

Add the remainder to the front of binary string

Divide the number by 2

End wile

Return the binary string

ENDMETHOD

4. Write the code for the **ConvertToBinary** method using the pseudo-code in Ex 3.

Private string ConvertToBinary(int number)

{

Int remainder=0;

String Binary=” “;

While(number>0)

{

remainder = number%2;

binary =remainder+binary;

number=number/2;

}

Return binary;

}

1. Write the code which gets a value from a textbox and displays the binary number for that value in the console/output window. Error checking is not required.

Int number =0;

String result = “”;

Number = int.parse(txt\_number.text);

Result = ConvertToBinary(number);

Console.WriteLine(result);

6. **(Tutorial Handin Exercise)** Re-write the code from Ex 5 so that the user enters a number and the application will display the binary number of each value from 1 to the number the user entered.

For (int i=1; i<=number; i++)

{

Result = ConvertToBinary(number);

Console.WriteLine(result);

}

7. Discuss and write down the advantages and disadvantages of using methods in code. Is it worth writing a method if the code is only called/used once?

Adv:

Allows us to breakup large code into smaller blocks

Write the code once in a method and call many times

Easier to make changes to the code

Easier to work in teams

Disadvantage:

Can’t see the code consecutively

Calling methods uses a very small amount of time and resources

8. Study this code:

Random rand = new Random();

private void buttonDrawRectangle\_Click(object sender,   
EventArgs e)

{

Graphics paper = pictureBoxDisplay.CreateGraphics();

SolidBrush br = new SolidBrush(Color.Red);

int colorNum = rand.Next(40);

if (colorNum < 10)

{

br.Color = Color.Blue;

}

else if (colorNum < 20)

{

br.Color = Color.Orange;

}

else if (colorNum < 30)

{

br.Color = Color.Pink;

}

paper.FillRectangle(br, 100, 100, 50, 50);

}

private void buttonDrawCircle\_Click(object sender,   
EventArgs e)

{

Graphics paper = pictureBoxDisplay.CreateGraphics();

SolidBrush br = new SolidBrush(Color.Red);

int colorNum = rand.Next(40);

if (colorNum < 10)

{

br.Color = Color.Blue;

}

else if (colorNum < 20)

{

br.Color = Color.Orange;

}

else if (colorNum < 30)

{

br.Color = Color.Pink;

}

paper.FillEllipse(br, 200, 200, 100, 100);

}

Imagine that you have to add three more colours to the above code. Discuss how you could use a method to reduce the repeated code in the above example and make it easier to change. Then write the method.

**Week 8: Tutorial A**

**Text Files**

**Objectives**

To introduce you to accessing text files. You will learn how to open and close text files and how to read and write from a   
text file.

**Reading**

Before coming to this tutorial, you should have read sections 5.6 and 5.7 on text files in Chapter 5 of Gaddis.

**Summary**

1. Text Files.

2. File stream objects.

3. Opening and closing text files.

4. Reading from a text file.

**Exercises**

1. What is a file? What is a text file?

**File**:A series of binary numbers stored on a secondary storage device

**Text File**:A series of ASCII values stored as binary numbers on a secondary storage device

2. Write statements to open the following files using the filename variable given:

string filename = @"H:\students.txt";

1. Open the file to store information in and erase any previous information.

Streamwriter writer;

writer =File.CreateText(filename);

1. Open the file to store new information at the end of the existing information in the file.

Writer=File.AppendText(filename);

1. Open the file to read information from the file.

SteamReader reader;

Reader =File.OpenText(filename);

3. What does the following code do? What should go in the if statement? Does it need an else statement? Explain.

const string FILTER = "Text Files|\*.txt|All Files|\*.\*";

openFileDialog1.Filter = FILTER;

if (openFileDialog1.ShowDialog() == DialogResult.OK)

{

//Open the selected file

Reader=File.OpenText(openDialog1.FileName);

//Process the File

Reader.Close();

}

No close statement so no else section

4. The **objects.txt** file contains information about objects which can be used to draw the object on the form. For each object in the file it will have the name of the object followed by its information, for example:

Rectangle

<X Position>

<Y Position>

<Width>

<Height>

Line

<Start X Position>

<Start Y Position>

<End X Position>

<End Y Position>

.

.

.

The pseudo-code given describes an algorithm that will read in information about 1 object and draw it and repeat this until the file has been processed.

Declare variables

Set the filter for the dialog control

IF user has chosen a filename THEN

Open the chosen file for input

WHILE not end of file

Get type of object

Get the first number

Get the second number

Get the third number

Get the fourth number

IF object is a Rectangle

Draw a rectangle

ELSEIF object is a Line

Draw a line

ENDIF

ENDWHILE

Close the file

ENDIF

Complete the code on the next page based on the algorithm given above.

int num1, num2, num3, num4;

string objectType;

Graphics paper = pictureBoxDisplay.CreateGraphics();

Pen pen1 = new Pen(Color.Blue, 2);

const string FILTER = "Text Files|\*.txt|All Files|\*.\*";

StreamReader reader;

//Set the filter for the dialog control

openFileDialog1.Filter = FILTER;

//Check to see if the user has selected a file to open

if (openFileDialog1.ShowDialog() == DialogResult.OK)

{

//Open the selected file

reader = File.OpenText(openFileDialog1.FileName);

//Repeat while it is not the end of file

while (!reader.EndOfStream)

{

//Read the object type from the file

objectType = reader.ReadLine();

//Read the four numbers

Num1 = int.Parse(reader.ReadLine());

Num2 = int.Parse(reader.ReadLine());

Num3 = int.Parse(reader.ReadLine());

Num4 = int.Parse(reader.ReadLine());

//Check if the object is rectangle

If(objectType == “Rectangle”)

{

Paper.DrawRectangle(pen1, Num1,Num2,Num3,Num4);

}

Else if(objectType==”Line”)

{

Paper.DrawLine(pen1, Num1,Num2,Num3,Num4);

}

}

//Close the file

reader.Close();

}

5. **(Tutorial Handin Exercise)**Identify the errors in the following code segment.

const string FILTER = "Text Files|\*.txt|All Files|\*.\*";

const int BAR\_HEIGHT = 20;

/// <summary>

/// Loads a file of vote data and displays the number of votes

/// for each party as a bar graph.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void buttonCalc\_Click(object sender, EventArgs e)

{

Graphics paper = pictureBoxDisplay.CreateGraphics();

SolidBrush br = new SolidBrush(Color.Gold);

Pen pen1 = new Pen(Color.Black, 2);

StreamReader reader;

//Set the filter for the dialog control

openFileDialog1.Filter = FILTER;

//Check if the user has selected a file

if (openFileDialog1.ShowDialog() == DialogResult.OK)

{

//Open the selected file

reader = File.OpenText(openFileDialog1.FileName);

//Repeat while it is not the end of file

while (!reader.EndOfStream)

{

//Get the name of a political party

string party = reader.ReadLine();

//Get the number of poll votes

int numVotes = int.Parse(reader.ReadLine());

//Calculate the width of a bar

int barWidth = numVotes/1000;

//Set the position of the bar on the graph

int x = 0;

int y = 0;

//Draw the bar

paper.FillRectangle(br, x, y, barWidth, BAR\_HEIGHT);

paper.DrawRectangle(pen1,x,y,barWidth,BAR\_HEIGHT);

//Shift down to the next bar position

y += BAR\_HEIGHT;

x=0

//Add up the votes

int total =+ numVotes;

}

MessageBox.Show("Total votes is " + total.ToString());

reader.Close();

}

}

// error1:total declared inside the while loop

//error 2: total += numVotes;

//error3: y=0;

**Week 8: Tutorial B**

**Text Files and methods**

**Objectives**

To solve more complex problems involving text files and writing to text files.

**Reading**

Before coming to this tutorial, you should have read sections 5.6 and 5.7 on text files in Chapter 5 of Gaddis.

**Exercises**

1. **(Tutorial Handin Exercise)**Consider the code below:

try

{

numUnits = int.Parse(textBoxUnits.Text);

if (numUnits > 0)

{

totalCost = numUnits \* ITEM\_PRICE;

textBoxTotalCost.Text = totalCost.ToString("c");

}

else

{

MessageBox.Show("Invalid number of units");

textBoxUnits.Clear();

textBoxTotalCost.Clear();

textBoxUnits.Focus();

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

textBoxUnits.Clear();

textBoxTotalCost.Clear();

textBoxUnits.Focus();

}

1 (cont) Discuss how a method could be used to improve the code. Write the method and make the changes to the code above to use   
that method.

2. The following code below will open a file of student names, in the file it has the first name and surname of a student on   
separate lines.

StreamReader reader;

string firstname = "";

string surname = "";

if (openFileDialog1.ShowDialog() == DialogResult.OK)

{

reader = File.OpenText(openFileDialog1.FileName);

while (!reader.EndOfStream)

{

firstname = reader.ReadLine();

surname = reader.ReadLine();

}

Reader.Close();

}

Modify the code so that it displays in a MessageBox window the number of students in the file.

Modify the code so that it displays in a MessageBox window the number of students with the first name of Mary.

3. The code given below will process a file of student information and write out information to a new file. The *surname*, *firstname*, *age* and *address* are stored on separate lines for each student in the input file. It will then ask for the name of the report file. The two files are then opened and it will write to the report file the student firstname and surname, age and address on one line for students whose age is greater than 40. The report file should be neatly formatted with the information in nice columns.

a) Use the OpenFileDialog control to get the name of the input file and check if the user clicked on the Ok button.

b) Use the SaveFileDialog control to get the name of the report file which information will be written to.

c) Open the input and the report file.

d) Write a loop which will repeat until the end of the file is reached.

e) Read the surname, firstname, age and address from the file and store them into the variables given in the code.

f) Write out all of the student's information to the report file if their age is greater than 40 all on one line, neatly spaced out so that the information is in columns.

g) Close the input and report files.

h) Display an appropriate error message if an exception occurs.

i) How would you add a header to the report file to show what each column of information is?

Use the code below to answer Exercise 3.

StreamReader reader;

StreamWriter writer;

string surname;

string firstname;

int age;

string address;

//Answer a) below

if (

{

//Answer b) below

if (

{

try

{

//Answer c) below

reader =

writer =

//Answer d) below

while (

{

//Answer e) below

surname =

firstname =

age =

address =

//Answer f) below

}

//Answer g) below

}

catch (Exception ex)

{

//Answer h) below

}

}

}

**Week 9: Tutorial A**

**Arrays and CSV Files**

**Objectives**

To introduce you to the concept of Comma Separated Value (CSV) files and how to process them. You will also learn the concept of arrays and how to manipulate a single element of an array and how to process the entire array.

**Reading**

Before starting this practical, you should have read sections 7.1 up to and including 7.8 and sections 8.1 and 8.2 of Gaddis.

**Summary**

1. Declaring arrays.

2. Initialising and manipulating arrays.

3. Reading information from CSV files.

**Exercises**

1. What is an array?

2. How do you declare an array in C#?

3. Write array declarations for the following:

a) An array named scoresArray to hold the golf scores for 10 golfers.

b) An array named namesArray to hold the names of a class of 50 students.

c) An array names pricesArray to hold the price of 12 items.

d) An array names pointsArray to hold 6 Point objects.

e) An array named colorsArray to hold 7 colour values and initialise the array with colour values.

4. Write c# statements for the following:

a) Store a golf round score of 66 for the first golfer in the array.

b) Store the name "Din Djarin" for the 5th person in the names array.

c) Store the value $15.50 for the last item in the prices array.

d) Store the point 100,150 at the beginning of the Points array.

e) Change the fourth colour in the rainbow array to Orange.

5. Write c# statements for the following:

a) Display the name of the first person in the names array in the **textBoxName** control.

b) Set the colour of the brush named **br** to a random colour in the colours array.

c) Draw lines between all of the points in the points array.

d) Calculate the total cost of all the items in the prices array.

6. **(Tutorial Handin Exercise)**Assume the following array has been declared:

int[] numsArray = new int[] {12, 56, 4, 15, 33};

a) What will happen if the code below is executed?

MessageBox.Show(numsArray[10].ToString());

b) What are the error in the code below which adds up all the values in the array?

int total = 0;

for(int i = 0; i <= numsArray.Length; i++)

{

total = numsArray[i];

}

7. The **objects.csv** file contains the following information:

Rectangle,<X>,<Y>,<Width>,<Height>

Line,<Start X>,<Start Y>,<End X>,<End Y>

Rectangle,<X>,<Y>,<Width>,<Height>

.

.

.

Write code to open the file **objects.csv** for reading and then write a loop which will repeat until the end of the file is reached. Each time through the loop, read a line from the CSV file into a string variable, then split the individual values into a string array using the split method of the string variable. Store each value from the string array into appropriate variables and objects and display those values in a listbox called **listBoxDisplay**. Then draw the object to the form and then read in the next line.

Use the partial code on the next page as a starting point.

Graphics paper = pictureBoxDisplay.CreateGraphics();

Pen pen1 = new Pen(Color.Black, 2);

string line = "", objectType = "";

string[] csvArray; int num1 = 0, num2 = 0, num3 = 0, num4 = 0;

if (openFileDialog1.ShowDialog() == DialogResult.OK)

{

reader = File.OpenText(openFileDialog1.FileName);

while (!reader.EndOfStream)

{

try

{

//Read a line from the file

//Split the values in the line using an array

//Check if the array has correct number of elements

if (

{

//Extract the values from the array into

//separate variables in the correct datatype

if (objectType == "Rectangle")

{

paper.DrawRectangle(pen1, num1, num2,

num3, num4);

}

else if (objectType == "Line")

{

paper.DrawLine(pen1, num1, num2, num3, num4);

}

}

else

{

Console.WriteLine("Error: " + line);

}

}

catch

{

Console.WriteLine("Error: " + line);

}

}

reader.Close();

}

}

**Week 9: Tutorial B**

**Arrays**

**Objectives**

To solve more complex problems using arrays.

**Reading**

Before starting this practical, you should have read sections 7.1 up to and including 7.8 and sections 8.1 and 8.2 of Gaddis.

**Exercises**

1. Assume that an array called **temperatures** has been filled with temperature values, write an algorithm which will find the highest temperature value.

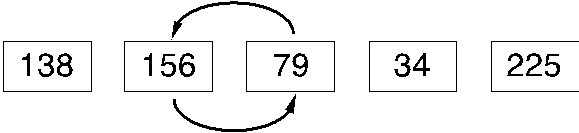
2. Write the method called **FindMaxTemp** which does not need any parameters and returns the highest temperature value found. Write the code for the algorithm in Ex 1 in the method.

3. Bubble Sort

a) Lay all of the cards out in one long row, ready to be sorted.

i Point at the leftmost card with your left hand.

ii Compare the card you are pointing at with the card to its right. If the number on the right is lower, then exchange the two cards, otherwise leave them alone. Comparing the two cards counts as one comparison, regardless of whether they were swapped or not.



iii Move your left hand one card to the right.

iv Repeat steps ii and iii until you have reached the rightmost card. Note how with each pass, the highest card will sink to the bottom (right), while the low cards will slowly “bubble up” to the top (left).

v Repeat steps i to iv until no exchanges are made in a whole pass. The deck is now sorted.

b) **(Tutorial Handin Exercise)** Write the algorithm for Bubble Sort

Declare variables

WHILE a swap has been made

Set swap made to false

ENDWHILE

c) Write the code for Bubble Sort based on your algorithm in b) on the array given below.

const int SIZE = 10;

int[] numbersArray = new int[SIZE]

Random rand = new Random();

for (int i=0; i<SIZE; i++)

numbersArray[i] = rand.NEXT(100);

**Week 10: Tutorial A**

**Lists**

**Objectives**

To introduce you to the concept using Lists to store information.

**Reading**

Before coming to this tutorial, you should have read section 7.9 of Gaddis.

**Summary**

1. Declaring Lists.

2. Manipulating elements in a List.

3. Manipulating multiple Lists.

**Exercises**

1. What is a list? How does a list differ from an array?

2. How do you declare a list in C#?

List<int> idList = new

3. Declare a List called **namesList** to store the names of people.

4. Display the contents of the list along with index positions after the following code is executed:

namesList.Add("John Diggle");

namesList.Add("Green Arrow");

namesList.Add("Felicity Smoak");

5. Display the contents of the list after the following code is executed.

namesList.Insert(1, "Laurel Lance");

6. Display the contents of the list after the following code is executed.

namesList.RemoveAt(0);

7. How could you always add names to the beginning of the list rather than the end of the list.

8. Assume the contents of the list is the 3 names added in Ex 2. Write the statement which will display the second person's name in a message window.

9. (Spoiler Alert!!!) If the second name in the list is **Green Arrow** then change it to **Oliver Queen**.

10. **(Tutorial Handin Exercise)**   
Write code that will search through the list for **Green Arrow** and if the name is found, change it to **Oliver Queen**.

11. State the advantages and disadvantages of using a List against using a normal array.

Advantages:

Disadvantages:

12. Declare lists to store information about a class of students. The information that needs to be stored about each student is:

• A list named idList to store all ID numbers.

• A list named surnameList to store all surnames.

• A list named firstnameList to store all firstnames.

• A list named hasPaidList to store whether students have paid their fees or not, which is a Boolean value.

13. Using the lists declared previously write the code to retrieve information about a student from textboxes and store the student's information into the lists. The textboxes are called **textBoxId**, **textBoxSurname**, **textBoxFirstname** and a checkbox called **checkBoxHasPaid** is used to show if the student has paid their fees or not.

int id = 0;

string surname = “”;

string firstname = “”;

bool hasPaid = false;

try

{

//Get information from controls and store into variables

//Add the values to the lists

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

14. Assume that a listbox named **listboxStudents** has been added to the form. Write a method named **UpdateListbox** which will clear the contents of the listbox and then display the firstname and surnames of all of the students inside the listbox.

Clear listbox

FOR each student in the lists

Add student name to the listbox

ENDFOR

**Week 10: Tutorial B**

**Lists**

**Objectives**

To introduce you to the concept of using Lists to store information.

**Reading**

Before coming to this tutorial, you should have read section 7.9 of Gaddis.

**Summary**

1. Retrieving values from a List.

2. Loading a csv file of information into lists.

3. Selecting items from a listbox.

**Exercises**

1. Create a method called **Initialise** that will clear out any student information that has been stored in your lists you defined in Ex 10 of the last tutorial.

Discuss the concept of Garbage Collection. What is it and why do you need it?

Private void Initialise()

{

idList.Clear();

surnameList.Clear();

firstnameList.Clear();

hasPaidList.Clear();

UpdateListBox();

}

Junk Memory: Memory locations that have been used but can no longer be accessed.

The .Net Framework is a memory managed environment.

The .Net Framework performs garbage collection, it looks through memory for junk memory and frees it back to the OS.

2. Write the menu event method for the **Save As…** menu item which will get the filename to save the information to and then saves all student information in the list to a CSV format file.

Declare variables

Set the filter for the dialog control

IF the user selects file and clicks on OK THEN

Open output file

FOR each student in the lists

Write student info in CSV format to file

ENDFOR

Close output file

Set titlebar to show app name and filename

ENDIF

StreamWriter writer;

saveFileDialog1.Filter = FILTER;

if (saveFileDialog1.ShowDialog() == DialogResult.OK)

{

Writer=File.CreateText(saveFileDialog1.FileName);

For(int i=0;i<idList.Count;i++)

{

Writer.WriteLine(idList[i] + “,”+firstnameList[i] + “,” + “surnameList[i]” + “,” hasPaidList[i]);

}

writer.Close();

this.Text = saveFileDialog1.FileName + " - Student System";

}

3. Write a menu event method for the **Load File…** menu item which will ask the user for the file to load and then loads the student information from the specified file and populates the student list.

Declare variables

Set the filter for the dialog control

IF the user selects file and clicks on OK THEN

Open input file

Initialise student list

WHILE not end of file

Read csv line from file

Split csv line storing values in an array

TRY

IF array has correct length THEN

Extract data into separate variables

Add student information to lists

ELSE

Display corrupt CSV line to console

window

ENDIF

CATCH

Display corrupt CSV line to console window

ENDCATCH

ENDWHILE

Close the input file

Set titlebar to show app name and filename

Update the listbox

Clear input controls

ENDIF

const string FILTER = "CSV Files|\*.csv|All Files|\*.\*";

StreamReader reader;

string line = "";

string[] csvArray;

int id = ;

string firstname = "";

string surname = "";

bool hasPaid = false;

openFileDialog1.Filter = FILTER;

if (openFileDialog1.ShowDialog() == DialogResult.OK)

{

reader = File.OpenText(openFileDialog1.FileName);

Initialise();

while(!reader.EndOfStream)

{

try

{

//Read a line from the file

Line=readr.ReadLine();

//Split the values in the line using an array

csArray=line.Split(‘,’);

//Check if the array has correct number of elements

if (csvArray.Length==4)

{

//Extract the data into separate variables

Id=int.Parse(csvArray[0]);

Firstname=csvArray[1];

Surname=csvArray[2];

hasPaid=bool.Parse(csvArray[3]);

/\*If(csvArray[3]==”True”)

{

hasPaid=”True”;

}

Else

{

hasPaid=”false”;

}

\*/

//Store the values in the lists

idList.Add(id);

idList.Add(firstname);

idList.Add(surname);

idList.Add(hasPiad);

}

else

{

Console.WriteLine("Error: " + line);

}

}

catch

{

Console.WriteLine("Error: " + line);

}

}

reader.Close();

this.Text = openFileDialog1.FileName + " - Student System";

UpdateListBox();

ClearControls();

}

4. Create a **SelectedIndexChanged** event method for the listbox so that when the user clicks on a different name in the listbox it will display if that person has paid their fees or not.

Int index=ListBoxStudents.SelectedIndex;

If(hasPaidList[index]==true)

{

MessageBox.Show(“Has Paid fees”);

}

Else

{

MessageBox.Show(“Has not Paid fees”);

}

1. Write the code that will remove the student that has been selected in the listbox.

Int index=ListBoxStudents.SelectedIndex;

idList.RemoveAt(index);

firstnameList.RemoveAt(index);

surnameList.RemoveAt(index);

hasPaidlist.RemoveAt(index);

UpdateListBox();

1. **(Tutorial Handin Exercise)** Create a method called **CountPaid** which will return back the number of students that have not paid their fees.

Private void Count\_notPaid()

{

Int count=0;

Foreach

}

**Week 11: Tutorial A**

**Problem Solving Exercises**

**Objectives**

This tutorial will be an opportunity to practice solving problems   
using algorithms.

**Exercises**

1. You will be creating an application to store information about Bluray movies that can be hired from a movie rental store. Each Bluray has a unique id number (int), title (string), genre (string), new release (bool) and rental price (decimal). Create class scope lists to store all information about the Blurays in the store.

List<int> idList =new List<int>();

List<string> titleList =new List<string>();

List<string> genreList =new List<string>();

List<bool> isNewReleaseList =new List<bool>();

List<decimal> priceList =new List<decimal>();

2. Assume the following method has been created:

/// <summary>

/// Updates the list box will all Bluray information.

/// </summary>

private void UpdateListBox()

{

//Clear the listbox

listBoxDisplay.Items.Clear();

//Loop through all blurays

for (int i = 0; i < idList.Count; i++)

{

//Add the current Bluray's information to the listbox

listBoxDisplay.Items.Add(idList[i].ToString().PadRight(5) + titleList[i].PadRight(50)

+ genreList[i].PadRight(20)

+ isNewReleaseList[i].ToString().PadRight(10)

+ priceList[i].ToString("c"));

}

}

Write the code to load a csv file of Bluray information and store the information into the lists created in Ex 1 and also display the information in a listbox using the **UpdateListBox** method.

const string FILTER = "CSV Files|\*.csv|All Files|\*.\*";

StreamReader reader;

string line = "";

string[] csvArray;

int id =0;

string title =””;

string genre=””;

openFileDialog1.Filter = FILTER;

if (openFileDialog1.ShowDialog() == DialogResult.OK)

{

reader = File.OpenText(openFileDialog1.FileName);

while (!reader.EndOfStream)

{

try

{

line = reader.ReadLine();

csvArray = line.Split(',');

if (csvArray.Length == 5)

{

Id = int.parse(csvArray[0]);

Title= Array[1];

Genre=Array [2]);

isNewRelaese=bool.Parse(csvArray[3]);

price=decimal.parse(csvArray[4]);

//Add the values to the list

IdList.Add(id);

titleList(Add(Title);

genreList.Add(genre);

isNewReleaseList.Add(isNewRelaese);

pricelist.Add(price);

}

else

{

Console.WriteLine("Error: " + line);

}

}

catch

{

Console.WriteLine("Error: " + line);

}

}

reader.Close();

UpdateListBox();

}

3) Create a method called **FilterListbox** which is passed a genre as a string and it will only display the title of Blurays in that genre, in a listbox called **listboxDisplay**.

Private void FilterListbox(string genre)

{

listBoxDisplay.Items.Clear();

for(int i-0; i<idList.Count;i++)

{

//Check if the current movies genre matches the genre to filter on

If(genreList[i]==genre)

{ //use the code to add

listBoxDisplay.Items.Add(idList[i].ToString().PadRight(5) + titleList[i].PadRight(50)

+ genreList[i].PadRight(20)

+ isNewReleaseList[i].ToString().PadRight(10)

+ priceList[i].ToString("c"));

}

}

}

}

4. Using a button event, call the filter method getting a genre from **comboBoxGenre** which is then passed to the method.

If(textBoxGenre.Text==”All Movies”)

{

UpdateListBox();

}

Else

{ FilterListbox(textBoxGenre.Text);

}

}

5) **(Tutorial Handin Exercise)**Create a method called **FindBlurayIndex** which is passed an id number and returns the index position of the Bluray in the lists. If the Bluray is not found then return -1.

Private int FindBlorayIndex(int id)

{

Int index =-1;

For(int i=0; i<idList.Count && index ==-1;i++)

{

If(idList[i]==id)

{

Index = I;

Break;

}

}

Return index;

}

6) Create a method called **RemoveBluray** which is passed a unique ID number and removes that Bluray from the list. Then call the method from a click event.

Private void RemoveBluray(int id)

{

Int index =0;

Index =FindBlueRayindex(id);

If( index !=-1)

{

idList.RemoveAt(index);

titleList.RemoveAt(index);

isNewReleaseList.RemoveAt(index);

pricelist. RemoveAt(index);

}

7) Create a method called **GetRentalPrice** which is passed a unique ID number and the current day and returns back the rental price of the Bluray. If the current day is Tuesday and the Bluray is not a new release movie return back $1.25 as the rental price, otherwise return back the current rental price. Assume that the elements of the list are not sorted by ID number.

**Week 11: Tutorial B**

**Problem Solving Exercises**

**Objectives**

This tutorial will be an opportunity to practice solving problems   
using algorithms.

**Exercises**

1. Assume a class scope list named **temperaturesList** contains a temperature value in each element of the list as a double. Write the pseudocode which will calculate the average temperature from all the values in the list.

2. **(Tutorial Handin Exercise)**    
Write a method called **CalcAveTemp** which returns the average temperature based on your pseudo-code in Ex 1.

3. Write the code for a click event method which displays the average temperature in a message window using the **CalcAveTemp** method.

4. Write the pseudo-code which will count the frequency of each letter in a string. The case of the letter should not matter.

5. Write a method called **CountFrequency** which returns an array of ints that is the frequency of each letter, based on your pseudo-code in Ex 4.

*Hint: word[0] gives the first letter in the string held in word.*

6. Write the code that reads a string from **textBoxInput** and then displays the frequency of each letter to the console window.

**Week 12: Tutorial A**

**Exam Preparation**

**Objectives**

To work through questions so the student is more prepared to answer questions in the exam.

**Question 1**

a) Circle all of the problems with the code below, which converts a Fahrenheit value into a Celsius value, where C = (F – 32) \* 5 / 9?

private void ConvertTemp(double fahrenheit)

void as it returns a value

{

double celsius;

celsius = (textBoxFahrenheit.Text – 32) \* (5 / 9);

}

private void buttonConvert\_Click(…)

{

double celsius = 0;

celsius = ConvertTemp(textBoxFahrenheit.Text);//text box must be passed

textBoxCelsius = celsius;

}

b) Write the correct C# code for the method in part a).

private double ConvertTemp(double fahrenheit)

void as it returns a value

{

double celsius;

celsius = int(fahrenheit– 32) \* (5 / 9);

returns Celsius;

}

private void buttonConvert\_Click(…)

{

double celsius = 0;

double Fahrenheit=int.Parse(textBoxFahrenheit.Text);

celsius = ConvertTemp(Fahrenheit);//text box must be passed

textBoxCelsius.Text = Celsius.ToString();

}

**Question 2**

a) Write the pseudo-code that draws 20 steps of a staircase in the pattern below.

For steps to draw

Draw step at current x and y position

If current step is in first half Then

shift x to the right by half the width

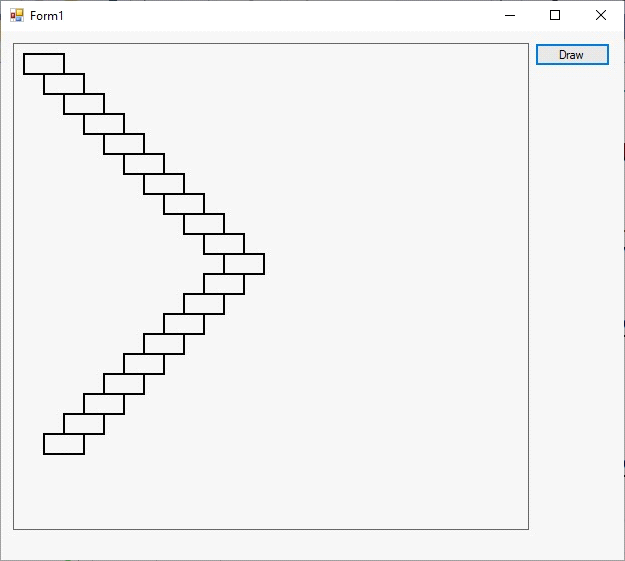
Else

shift x to the left by half the width

ENDIF

Shift y down by the height

End for

For each 

b) **(Tutorial Handin Exercise)**    
Write the C# code for your algorithm in part a).

const int NUM\_STEPS = 20;

const int WIDTH = 40;

const int HEIGHT = 20;

Graphics paper = pictureBoxDisplay.CreateGraphics();

Pen pen1 = new Pen(Color.Black, 2);

int x = 10;

int y = 10;

for(int i=1;i<=NUM\_STEPS;i++)

{

Paper.DrawRectangle(pen1,x,y,WIDTH,HEIGHT);

If(i< NUM\_STEPS/2)

{

X+=WIDTH/2;

}

Else

{

X-=WIDTH/2;

}

Y+=HEIGHT;

}

**Question 3**

a) Create a method called **CountPositive** which is passed an array of integer temperature values and returns back the number of positive temperature values in the array.

Private int **CountPositive(int []temparray)**

**{**

**Int count =0;**

**For(int i=1;i<= temparray.Length;i++)**

**{**

**If(temparray[i]>=0)**

**{**

**Count++**

**}**

**Return count;**

**}**

b) Create a method called **CalcNegativeAve** which is passed a list of double temperature values and returns back the average of all negative values in the list.

Private int **CalcNegativeAve** **(List<double> temparrayList)**

**{**

**Int ngativecount =0;**

**Decimal sum=0;**

**Int Avearage=0;**

**For(int i=1;i<= temparrayList.count;i++)**

**{**

**If(temparrayList [i]<0)**

**{**

**Sum+= temparrayList[i];**

**ngativecount ++**

**}**

**Average=sum/ ngativecount;**

**Return Average;**

**}**

**Binary numbers**

**Question 4**

Write C# statements for the following:

a) Create some lists to store information about cats. The application should store the name, age and weight of a cat.

b) Add the following cats to the lists:

Maggie, 5.5 years, 3.5kg

Mac, 2 years, 2.5kg

Tosh, 1.9 years, 5.1kg

c) Display names and ages of all the cats into a listbox called **listboxNames**, using one line per cat.

d) Write the C# code to remove the second cat from the application.

**Week 12: Tutorial B**

**Exam Preparation**

**Objectives**

To work through questions so the student is more prepared to answer questions in the exam.

**Question 1: Lists**

(a) Draw the contents of the data variable after the following C# code has been executed. Show the indexes of each value as well as the value itself.

List<int> data = new List<int>();

data.Add(22);

data.Add(44);

data.Add(33);

(b) After the following extra code is executed, draw the new contents of the data variable.

data.Sort();

data.Add(88);

data.Insert(1, 55);

(c) Write some C# code that gets a number from the textBoxNum control and displays the number of values in the list that are less than the number from the textbox.

**Question 2: Loops and selection**

a) What is displayed by the following code:

string info = "";

for (int i = 1; i <= 10; i++)

{

if (i % 4 == 0)

{

info += "\*";

}

else

{

info += "-";

}

}

Console.WriteLine(info);

b) **(Tutorial Handin Exercise)** What is displayed by the following code:

string info = "";

for (int j = 0; j <= 5; j++)

{

for (int i = 1; i <= 10; i++)

{

if (i % 4 == 0)

{

info += "\*";

}

else

{

info += "-";

}

}

Console.WriteLine(info);

info = "";

}

c) Write the code for a click event button that will draw 10 circles with a width and height of 20 at a random x position in a picturebox and a y position of 100. Declare all variables/objects as necessary.

**Question 3: Arrays**

Assume the following arrays have been declared:

String[] dateArray = new string[] {"1/6/17", "2/6/17",…}

int[] tempArray = new int[] {5, -1, 10, 2, -4,…}

a) Write the code to display how many negative temperatures there are.

b) Write the code that gets minimum and maximum values from textboxes and displays the number of temperatures between those values inclusive. Error handling is not required.

c) Write the code to display the date of the lowest temperature value.

**Question 4: Methods**

a) Write a method called **CalcUnits** which is passed a monetary amount and the price of an item and returns back how many units can be bought with the amount.

b) Call the method and display the number of units in a message window.

c) Write a method call **FindName** that checks if the name passed to the method is already stored in a class scope list called **namesList**. It should return back true if the name is already in the list or false if the name is not in the list.

d) Write the code that will add the name "Cara Dune" to the **namesList** only if it isn't in the list already.

1. What is a variable and what is it used for?

A variable is a named storage location in a computer's memory that holds a value. It is used to store and manipulate data during the execution of a program. Variables allow programmers to work with data and perform various operations on that data.

2. How do you create a variable that can be accessed by any method in the form’s class? Where should this variable be declared?

To create a variable that can be accessed by any method in the form's class, you should declare it as a class-level variable. This means declaring the variable outside of any specific method, directly inside the form's class. By doing so, the variable becomes a member of the class and is accessible throughout all methods within that class.

Here's an example of how to declare a class-level variable:

```csharp

public partial class Form1 : Form

{

// Class-level variable accessible by any method in the Form1 class

private int myVariable;

// Other methods and code for the form

}

```

3. What is a constant and why would you use one?

A constant is a value or an identifier that remains fixed throughout the execution of a program. Once a constant is defined, its value cannot be changed during the program's execution. Constants are typically used for values that remain constant throughout the program and are not expected to change.

The main reasons to use constants are:

- To make code more readable: Constants provide meaningful names for fixed values, making the code more self-explanatory and easier to understand.

- To avoid magic numbers: Magic numbers are hard-coded numeric values without context, which can make the code harder to maintain. Constants provide a way to give a name to such values, making the code more maintainable.

- To prevent accidental changes: By defining a value as a constant, you ensure that the value remains fixed and cannot be accidentally modified during program execution.

4. What does the Parse method do? Give an example of how to use it.

The `Parse` method is used to convert a string representation of a data type (e.g., int, double, bool) into its actual data type. It is commonly used to convert user input or data read from external sources, like files or databases, into the appropriate data types that can be used in calculations or other operations.

Example of using `Parse` method:

```csharp

using System;

public class Program

{

public static void Main()

{

// Example using Parse to convert string to int

string userInput = "42";

int intValue = int.Parse(userInput);

Console.WriteLine("Parsed integer value: " + intValue);

// Example using Parse to convert string to double

string userInputDouble = "3.14";

double doubleValue = double.Parse(userInputDouble);

Console.WriteLine("Parsed double value: " + doubleValue);

}

}

```

In this example, we use the `Parse` method to convert a string "42" into an integer `intValue` and a string "3.14" into a double `doubleValue`. The `intValue` and `doubleValue` can now be used as integers and doubles, respectively, for further calculations or processing.

Let's evaluate the expression using the given values: A=3, B=3, C=2, D=1.

Expression: A + B \* C / D

Substitute the values: 3 + 3 \* 2 / 1

Now, let's follow the order of operations (precedence) in mathematics:

1. Parentheses (not applicable here)

2. Exponents (not applicable here)

3. Multiplication and Division (left-to-right): 3 \* 2 = 6

4. Addition and Subtraction (left-to-right): 3 + 6 = 9

1. **Write an If statement equivalent to the following english expression: A is equal to B or A is greater than C and B is equal to D**

if ((A == B) || (A > C && B == D))

{

// Code block if the condition is true

}

else

{

// Code block if the condition is false

}

1. **Given the following values: A=3, B=3, C=2, D=1, is the above expression true or false? What is the problem here in terms of precedence?**

The result will be true.

(A == B) => (3 == 3) => true

(A > C && B == D) => (3 > 2 && 3 == 1) => (true && false) => false

Now, combining both parts using the logical OR operator:

(true) || (false) => true

No problem with the code, but logically thinking the steps will be executed even B==D condition is false

1. **State the benefits of writing comments in program code.**

* **Code Understanding**: Comments help developers and readers of the code understand the purpose, logic, and functionality of the code. They act as a guide, making it easier for someone else (or even the original author) to understand the code's intentions.
* **Code Maintenance:** Comments make code maintenance easier. When revisiting code after some time, comments can serve as reminders of the code's design and functionality, reducing the time needed to make changes or fix issues.
* **Collaboration:** Comments facilitate collaboration among team members. They enable developers to communicate ideas, explanations, and thoughts about the code, leading to better team understanding and productivity.
* **Debugging and Troubleshooting**: When debugging code or troubleshooting issues, comments can offer insights into the code's behavior and the intention behind specific decisions, speeding up the process of finding and fixing bugs.
* **Code Review and Quality Assurance:** During code reviews, comments provide context and rationale behind the code, making it easier for reviewers to assess code quality and suggest improvements.