

Data Type

أنواع البيانات

Primitives data type

محض - حقيقية - كسرية - جمل ذات واحد

Structure , non Primitives data type

«Complex DT» مفروقات من الأساسية غير سطحة

non Primitives و Array بالشكل Structure نقول

للتغا هي عبارة عن أنواع تكون منها Object أي حاجة تستخدم مما في C++ new

خاصية ما يعلق بال

byte → 1 byte short → 2 byte

int → 4 byte long → 8 byte

Primitives ⇒ char , int , float ...etc.

non Primitives

Real ⇒ float , double , decimal

Structure Datatype ⇒ Array , Array of Array

ـ مجموعة من العناصر المختلفة
ـ التي تجمع بعضها في وحدة واحدة

Interface

Enumerated Data Type

تعاملنا معها مفروض

ـ عبارة عن نوع من الموارد Constant (ثابت) اسم واحد

ـ عبارة عن نوع من أنواع Data Types وهي نوعين input و output

ـ حروف معينة وكما تعرف على أنها قصيرة

ـ «تعامل معها من حيث كذا»

أ. تعامل معها بطرقٍ مختلفٍ:

- عن طريق اسمه **كما هو**.

- يقول **enum**. وهذه تستخدمها إذاً وجد هذه الاسم.

بعد **enum** داخل البرنامج فلن تعرف بيها تستخدم هذه الطريقة
هذا النوع يستخدم تدريجياً أو **visual** حيث بناءً بشكل
كبير وستستخدم دائمًا.

فتسأل **enum** موجودة في كل مكان.

float x = 1,234f \rightarrow **لعرف أينها**

double d long l \rightarrow **ومن هنا بالسبيكل**

لو حطناها بدون كثيـرة هذه الأحرف يأخذها

float a = 23.0f \rightarrow **الساند في الأرقام هو int**

float a = 23 \leftarrow **int**

Operators

معلم واحد \rightarrow **Unary**

معلمات متعددة \rightarrow **Binary**

ثلاث معلمات \rightarrow **Ternary**

وكلها تختلف في الأولوية

a = 10 , b = 20 \Rightarrow x = + + a * b + +

x = 220 , a = 11 , b = 21

a = 9 , b = 10 \Rightarrow x = a + + * + + b

x = 99 , a = 10 , b = 11

Logic Operations: $>, <, \oplus, \ominus, \wedge, \vee$

Relational Operators: $<=, \geq, >=, \leq, =, \neq$

Bitwise Operators: $\ll, \gg, \neg, \wedge, \vee, !$

هذه العمليات تتعامل على مستوى البت مباشرةً بخلاف عمليات التعامل على المعاينات - يقيّم عشرة \geq و \leq التَّعامل معها كمتغير ثنائى.

(١٥) ومنها كذلك $\ll, \gg, \wedge, \vee, \neg, =, \neq$

٨ And يُعد ١ إذا كانت كل بิตات يساوى ١

٩ OR يُعد ١ إن أحد البتات يساوى ١

١٠ XOR يُعد ١ إن البتات مختلفتين

١١ Not يعكس قيمة البت

١٢ Left Shift إزاحة بعديات من اليسار

١٣ Right Shift إزاحة بعديات من اليمين

إذا أعددنا إزاحة للساز بعديات من اليمين واحد حاصل على إزاحة بعديات من اليمين

للمرين يمكن إزاحة حاصل على إزاحة بعديات من اليمين

لتصير عبارة عن عددين لغسته والهجري

Priority: الأولوية

١٤ Prefix إذا كنت $+,-,\wedge,\vee,\neg$

١٥ Unary $(\cdot), *, /$

ـ Conversion ٨٠

ـ ١ Implicitly Conversion

ـ في هذه المقدمة سنعلمكم بـ ٤ طرق لـ conversion، الـ implicit conversion هي أسرع وأبسط طرق، حيث يتم التحويل من النوع الأعلى إلى النوع المنخفض.

ـ مثال: `byte x = 5; short y; y = x;`

ـ ٢ Explicitly Conversion

ـ هرديج

ـ مثال: `int a; short b = 2345;`

ـ `a = (short) b;` or `a = checked((byte)b);`

ـ هرديج التحويل السابقة تسمى بـ **ـ conversion**، بينما الـ **ـ casts** هرديج التحويل.

ـ كما ولـ **ـ conversion** الأنواع المختلفة غير ملائمة.

ـ Convert using Class `int.Parse()`, `int.TryParse()`

ـ تحويل البيانات من نوع إلى نوع آخر.

ـ Convert using Convert Class

ـ تحويل بين أنواع البيانات المختلفة، «ـ تتعامل مع المجموعات من واحده، لغافون فمه».

ـ Convert using Encoding & Decoding

ـ تحويل البيانات من شكل قابل القراءة واللغة! له شكل آخر **ـ Encoding**.

ـ يمكن ترميمه ونقله بطريقة آمنة.

ـ يستخدم في الـ **ـ sockets** مع ارسال الملفات عند ما يرسل في

ـ الـ **ـ sockets** رسالة باللغة.

ـ ASCII, Universal Text Format

ـ `UTF-8`, etc.

ـ **ـ Encoding** ← يطلع أفرز من نظام تشغيل

إذا أشفرت البيانات بنظام معين تترجمه بنفس النظام فإذا أشفرت بالآسي ترجعه بالآسي وإذا أشفرت بالUTF-8 ترجعه بالUTF-8

```
string str = "abcdef";
```

```
byte[] b = new byte[str.Length];
```

$b = Encoding.ASCII.GetBytes(str);$

$str = Encoding.UTF8.GetString(b);$

* short = char = un byte

فهي بيانات 2 byte و هي unsigned هو char

{2, 4, 8, 16, 32, 64, byte} int

byte Converter Class

سيستخدم المتحول صن اب ل النوع byte والعكس

```
int[] n = new int[10];
```

$n[0] = 1; n[1] = 2; n[2] = 3;$

هذا إذا أردنا طباعة حجم المحفوظة فسوف تظهر لنا الحجم كأمثلة

ـ بالرغم من أننا استخدمنا 3 مواقع، فكيف يقوم باظهار حجم

ـ المحفوظة

ـ التعامل مع المحفوظات

ـ دائمًا هنا نتعامل مع `size` حجم المحفوظة كفرد أخر عمل لها

ـ سعة معينة ولاستخدام يكون حسب الفعل \rightarrow حجم يكون أقصى

ـ احتمال حسب الحاجة \Rightarrow

ـ الطرف الأيمن له n عدد معلوم من العناصر

والطريقة الثانية هي العلامة في النهاية. لا يكون لدي علامة في نهاية المعرفة مما هو الحال في المعرفات التقنية والتقنية وهذه الطريقة يكتب صيغة في الملفات والشبكات. عندما ترسل رسالة إلى طرف آخر فهو سبقها حتى آخر علامة يتوقف فيها بداخل بالباقي) بين سبق (رسالة وحسب مورثة (رسالة يفتح) البيانات الأولي وليتم تفاصيل رسالة مثل الجمل والعلامات وسما وحسب الجمل سمي حتى آخر رسالة وهذه طبيعة تراسل البيانات.

1 int [] a = new int [5] { 1, 2, 3, 4, 5 };

a = { 6, 7, 8, 9, 10 } ✓

2 int [] a = { 1, 2, 3, 4, 5 }

a = { 6, 7, 8, 9, 10 } ✗

لذا أعرفت معرفة وبعدها أتمت بما لا ينتهي لها لا تستطيع

يقول يساوى بالقواس يدوس new

في C++ بعد ما تعرف المعرفة لا تستطيع أن تأسى لغير

دخل رقواس مرة واحدة

في C * علوا دل new لغير الكلام

```
int [] array = new int [5] {1,2,3,4,5}
```

```
for (int i=0 ; i < array.length ; i++)
```

```
Console.WriteLine (array[i]);
```

foreach (int x in array) ← تعلم مفهوم
تكرار واحد

```
Console.WriteLine (x);
```

```
string [] str = {"aaa", "bbb", "ccc"};
```

```
foreach (string s in str)
```

```
Console.WriteLine (s);
```

```
double [,] a = new double [3,4]
```

→ [,] a = { {1,2,3,4},
{5,6,7,8}, {9,10,11,12} } ;

→ [,] a = { {1,2,3,4}, {5,6,7,8} } ;

مصفوفة متباينة

```
double [,] a = new double [3][4],
```

→ { {1,2,3},{1}, {1,2} } ;

مصفوفة مصفوفات

```
int [,] a = new int [3][];
```

كل صف يحتوى على
كتل اولا (اجباري)



اليوم: مدة درس:
الموافق: / /

foreach (int [] divisors in a)

{ foreach (int divisor in divisors)
Console-Write (divisor); }

int D [] array = new int [3] { }

array [0] = new int [5];

 [1] = ~ ~ [4];

new جمل

 [2] = ~ ~ [2];

كلمة

String

H.W

إكانتات ودوال وحتميات Object

ToUpper, ToLower, Length ... etc.

الواجب:

تعريف دالة بالـ signature // يكتفى بكتابته في البرنامج

```
int add (int a, int b) {
    return a+b; }
```

int defun (int a, int b, int (*f)(int, int)) {
 return f(a,b); }

معنى ارسال دالة لدالة

لداة عن ماقعوق داخل دالة main هي عبارة عن متغير في الذاكرة (أساساً) فإذا ذكر هذا المتغير بدلاً من scope حقه، وإن معناه المقصود حقه. فإذا ذكر حقوق فهو عام تذكره وبين مانستي. لكن إذا ذكر في دالة main وعرفتها داخل دالة main ما يمكن مستخدمها! لأن دالة main لا يمكن استخدامه فتحت دالة main.

إذا عرفنا دالة Prototype فوق أو داخل دالة main نعيث بها - main دالة بعد

main ()

int (*Ptr)(int, int) ; // مُؤشر على دالة
أي أنه إذا كان هناك دالة مدخلاتها int و int ونعيد int نفع المؤشر ساري للدالة بحيث يبدأ من ذكر الدالة تذكر المؤشر.

لأن إذا ذكرت لهاته أو المؤشر (ما نفس) العمل // وـ

int T = Ptr (10,30) ; // ≡ add (10,30) و

int x = 20 و y = 70 و

int t = defun (x,y, add) و

أقدر أرسل أي دالة

لذرنا تستقبل مؤشر من نوع دالة

الواجب:

```

defun (sides x, y, sub) j i tui) bba mi
defun (x, y, mult); fidae ericav
defun ((x; y;), div) g i s d tui e tui) m tui tui

```

نستخلص ملخص معرفات من نوع مؤشر دالة f مساتوشر على دواله و
نتعامل معها على حسب f بعد اتساعه

{ int [] (*ptr)(int,int) = { }

Ptr[0] = add; Ptr[1] = sub;

Ptr [2] = mult; Ptr [3] = div;

```
for (int i=0; i<4; i++)
```

```
{ Console.WriteLine( x,y ,Ptr[i] ); }
```

فیل ۲۰۰۰

class A

```
{ private: int value; }
```

public = A() { value = 0; }

```
public: A(int a) { value = a; }
```

public: int (*pf) (int, ~~int~~);

public void print()

$\{; f(p_f != null)$

```
Console.WriteLine(pf(value));
```

```
int foo ( int x )
```

```
{ if ( x < 0 ) return 0 ;  
else return 4 ; }
```

```
main ()
```

```
{ A a ( 100 ) ;
```

لدي امكانية المقادير هنا الكلاس من // و

البرنامج (رئيسى على هىئه خاصه

استعيرها عند التعديل او عند ما يتغير

وال value . وعند بدأه التعديل « عند ما يحصل شيئاً ما » .

الحداثة هي عبارة عن موصرات لحوالى

Click Click . عبارة عن منشئ دالة معنوية (

```
{ Button b = new Button () ;
```

~~But~~ b.Text = "OK" ;

b.Location = new Point (100 , 50) ;

b.Click += onClick ;

}

منشئ داخل الكلاس هذا المؤشر يستطيع أن يؤثر على دالة إذا كان

تحت

ساوى null ينفذ ما

صيغة ، لأن شاء بل الكلاس تكون في طبيعته بينما أول ما تسئل نشأ

وأول دالة تسئل هي طبيعة .

في زل C++ لأنني موشر الـ delegate void show (string Text);

موشر يوشر علىه دالة في void علىه هيئه كلاس

في زل C++ التعامل على هيئه كلاس

delegate كي تتساهم بسطرواحد مثل مثلك (سابق). اعتبره

void اسوي delegate اسوي show يوشر عليه دالة تعيد

void ونأخذ نعم.

delegate يغير كلاس ، طلو منه كلاس يوشر عليه دالة

Invoke method دالة ارسل لها (كلاس) ، دالة نوعها طرقا دالة

وهي مثل دالة Call

ـ (كانت اعرف من كلاس) delegate يوشر عليه دالة ذات الصفة

ـ (شي) ومقتضاها من نوعها

ـ كيف سترة دالة delegate

ـ عن طريق دالة Object name بالـ new له

ـ show psh new show(); وـ (اسـ دالة . show();)

ـ psh += اسـ دالة (+=)

هناك طريقة لتعريف حاتم صناعي delegate وبيانه للدالة وبهذا
هذه الدالة في نفس المكان. «يُشير على الدالة في نفس المكان»
 $\text{show ps} = (x) \Rightarrow \{\text{Console.WriteLine}(x); \}$

السبت ١٨/١٢/٢٠٢٦

المحاضرة ٣
مدونة المحاضرة
التاريخ: ٢٥-٢٥-٢٠٢٦

Class Distance

{ private int kilo ; في صورة ، طلاق تكررت
private float meter ; أبا تكون نوع البيانات

=

{ أحد الحلول مثل كلاس آخر ينبع بيانات التي نريد

template وطريقة أخرى باستخد

class Distance < A , B > // هنا يأتي رمز عادي

{ private A Kilo ; كلها مثل المثال ترسل نوع بيانات

private B Meter ;

=

{ Distance (A kval , B mval)

{ Kilo s Kval ; Meter s mval ; }

A getkilo () { return Kilo ; }

B getmeter () { return Meter ; }

void setkilo (A val) { Kilo = val ; }

void setmeter (B val) { Meter = val ; }

=

{

main ()

{ Distance < int , float > d = new Distance

int , float > (10 , 20 , 5) ;

`Distance <int, int> d2 = new Distance`

`int, int> (10, 20) ;` شكل ٤٥

لقد حاولنا في هذا المثال حرقنا مانع `int, int` ، ونراها مرة أخرى هل يمكننا من جديده؟ أعلمكم من هو ما يمكننا ما هو يعمل جدول للأساس ، لكن شكلاً من قبل فهو إذا لقاها ما يمكنها مرة أخرى .

يمكن تصميم أنواع template بالنسبة للدوال و طريقة أخرى وهي Overloading

Generic د. Overloading،

Generic د. هي في الأساس دالة أو عمليات تتعامل مع أكثر من شكل

`int add (int a, int b)`

↓ ↓ ↓ ↓

`A add <A> (Aa, Ab) ;`

وعن الاستدعاء

`add <int> (10, 20) ;` هنا يدخل Compiler

`add <float> (10.0, 20.0) ;` Overloading

يمكن نسخها

`delegate A add <A> (A a, Ab) ;`

يمكننا نشير على دالة وأعطيه نوع البيانات التي دالة سير

`add <int> d = new add <int> () ;` اسم الدالة

و تكون نوع المدخلات

function جاهز delegate هنالك

`Func<int, int>` int وترجمه int مستقبل

`Func<int, int, int>` int وترجمه Two int

`int add (int a, int b)` فتحناه (وقتناها) لـ add

بيانات استطاعه تعریف انت

`Func<int, int, int> d = new Func<int, int>(add);
d(10, 20);`

`Func<int> a =` int وترجمه a

concrete function

⇒ delegate Action

`Action<int, int>` void وترجمه Two int مستقبل

. Procedure - all, one باشر

`Action<string> a = new Action(Console.WriteLine);`

`a("aaa");`

⇒ delegate Predicate

`Predicate<int, int> ≡ Func<int, int, bool>`

محض انه يرجع على دالة مستقبل صفات خصوصيّة النوع وال مجرد

bool result =

`bool chodd (int x) { return x % 2 != 0; }`

`Func<int, bool> p = new Func<int, bool>(chodd);`

`Predicate<int> p2 = new Predicate<int>(chodd);`

Compiler I. Overload, هو عمل Generic.

foo (object x) \Rightarrow اتصال بـ object
يأخذ نوع عند الاستدعاء

foo (100); \Rightarrow عمر متحدة و ذلك

foo ('abc'); \Rightarrow object

int w; float f; \Rightarrow يُشير على كل اثنين

foo (w); foo (f); \Rightarrow وهو الأعم من السابق

فإذاً لم ينادى طرقاً لعمل Generic

Generic template - Generic Overload -

polymorphism \Rightarrow تَبَيَّنَ عَلَى هَذِهِ Generic

object \Rightarrow يُوْزَعُ عَلَى الْأَيْنَادِ كَمَا يُوْزَعُ عَلَى object

C \Rightarrow كل المضادات المحوودة في C و خلافها

List \Rightarrow قائمة عناصر لها إمكانات متفوقة منها إمكانيات مع

آخرها Remove At, Remove, Add \Rightarrow هذه كل

عند التعريف لا يتم تحديد أي مما في المفروضة العادي، لأن زرارات مسماة

array \Rightarrow ذلك للغرض التعريف سهولة إرسال نوع

Generic \Rightarrow المفروضة حيث تقبل أي نوع

int [] a = new int [10]

قلنا هنا new حتى أستطيع عمل = فيما بعد

Linked List \Rightarrow تحل محل القوائم المتسلسلة

main ()

```

{ int [] p = new int [ ] { 10, 30, 40, 50 };
List < int > l1 = new List < int > ();
l1.Add(10); l1.Add(20);
l1.Add(30); l1.Add(40);
for (int i = 0; i < l1.Count; i++)
{
    Console.WriteLine(l1[i]);
}
p = new int [ ] { 50, 60, 70, 80 }; // new list
p = sum (new int [ ] { 90, 80, 70, 60 });
}

```

هذا المثال يوضح في المشروع عدد 85 سطح مفتوحة (Open files) في

الذي يستعمل لأول مرة خطاً في المشروع.

int Countt (Fun < int, bool > p)

```

int t = 0;
for (int i = 0; i < count; i++)
{
    if (p(x[i])) // بافتراض أن p مصوّفة
        t++;
}
return t;

```

Countt (chodd)

int Countt2 (Fun < int, bool > p) int sum (x, n)

p(x[i], n) // لاستدعاء عددة لشرط

`int Count3 (Func<int, bool>p, int[] x, int n)`

عوسيه عمل و لست \Rightarrow Count3 (check, x, n)

عوسيه بيانات \Rightarrow check (x) \Rightarrow $x \% 2 == 0$

L1. Count (lp);

delegate bool check (int x);

٣

كيفية إرساله

- أولاً ترسل الدالة كيئها ذات ودّاً فـ `L`.

- ثانياً ترسل مكان `lp`.

استخدم `lambda expression`.

٤ ١١. `Where ((a,i) => (a % 2 == 0)) & (< 3)`.

لـ `Count` عدد `ToList<int>`

لـ `index` شرط معين `where`

الآن `Count` تعيد `List<int>`

`IEnumerable<int> pp`

`List<int> t = pp. Aggregate ((s, x) =>`

$s = s + x * x)$. `To String ()`

والآن `ToList` ترجع `List<Type>` وهي نفس عمل `To List` سيعطيها النوع

صيغة `Object` أعمل من موسّر فعل `Interface`،

الآن `pp` يدخل بالفعل زرائباً. نرسل له الدالة وهو يتنفيذ فحص `delegate`،

يعني المعاشرة كيئية. تعود بـ `واحدة` والوصول إلى `لـ داد` \leftarrow `Form`

ماذا تعني new ؟

هي حانة - ذكر بالنسبة لها هي حالة وبالنسبة لل Compiler

متغير يخزن في الـ heap وظيفته ترتيب الكائن

{ int x[] = {17, 20, 7, 8}; // عرفنا المعرفة ك النوع }

في هذه الحالة لا أستطيع أن أستدلل المعرفة قيم فيما بعد

في لغة C++ هنا وفقط في بيلغواول C++ new ،

int x[] = new int(17, 20, 7, 8); //

يبيت أنه داخل البرنامج يأتي لحظة كنت تحتاجها لعنصر وهذا

أُمر وليس عليه استبيان.

ـ new حمله مستقلة عن ذاتها يعني تكون ~~جزء من نفس المعرفة~~ معرفة

ـ قد لا تتعامل معها على هذه الأبيات أفضل صيغة تتعامل معها كأنها

member data

لتعرف أن لدي مثلاً دالة جمع عناصر المعرفة قد

int sum (int x)

{ = }

عند إرساله

sum (x);

يسأل المعرفة و () -

ي نفسها الخطأ

ـ فإذا أرسلنا له x فالمعرفة تأس على x وإذا أرسلنا له المعرفة فإنها توثر على hera .

للمعرفة تابعه زياره الاسم والـ

المستوى لمعرفة الآخر

$S+ = (new int(1) - i[i] \rightarrow S+ \times i[i])$

لوقلنا نعمل بطبع كل خصائص المعرفة

لـ `foreach` دالة `list` فإذا كان `i` يدعى المعرفة (سطر)

أرسل له دالة للطباعة بحيث يطبع العناصر وبالتالي هي عبارة

عن كل

أحد إجابات لسؤال الثاني بالاختبار

`string digitText = new string ('Zero', 1), "One", "Two",`

"Three", "Four", "Five")

`button1_Click ()`

{
is Convert.ToInt32(x);

Label1.Text = digitText[i];

}/

يسأله معرفة في وـ `ToString()` يطلع صاعده

`new TextBox(); new distance();`

يطلب منك أن تكتب في الكورة على

ـ `object` يساوي له صاعده ويطبع في الكورة على

ـ `delegate` يساوي له صاعده

يُأذن بـ `new` لـ `Object` بـ `new` لأن الأسماء هنا
لا تستطيع الوصول إلى أي في كل الأخطاء تكون ذلك `object`
ويمكن له عنوان في الذاكرة وادخلت الوصول إليه خذلها
المكان بعد ما تخرج من تقدير الوصول له.

متغير `reference` (مُؤشر) لما نعمل به `d`

له `new` يرجع (كائن) وهذا متغير قائم في الذاكرة

متوصلة عبارة عن موقع في الذاكرة يُؤشر إليه عنوان سوف يكون في المكان بعد

ـ `object`

- لست أنت هو الذي تحدى ذلك ستقبل ما توجه `new`

`print(new TextBox());`

⇒ `TextBox T1 = print(T1);`

ـ بذلك ترسل المتغير وتحصل على (كائن) في نفس المكان حيث

ـ `T1` يأخذ `new` في كل الأخطاء

⇒ `new distance(+,-,*,/)`

ـ `new` لا يُؤثر على ملائمة النقطة على سبيل المثال `new`

ـ `class A`

{ `public A()` }

ـ `=`

}

ـ `main()`

{ `new A()` }

ـ `new` إسناذ `Object`

ـ استعين بالـ `Constructor` وهو ملحوظة تقع في نقطتين

ـ برنامج > `Constructor`

عند ما تتعامل مع Compiler ضروري تفهم حينما ينزل ما و ماذا يفعل في كل حالة.

(ا) استنساخ - أخذ المكتوب من السطر

{ int [] x = {17, 20, 7, 8}; }
int [] y = x; // هنا يُشار على x بعنوان المنشورة

لديها مساحة x و y

y = x.clone(); // عبارة عن دالة محددة

تحل على سخ فحوى متغير آخر.

و هي مساعدة داخل object فاز بإعيد object وبالتالي

y = (int[]) x.clone(); // هنا نوع

عند ما يكون لديك Form ليس تكون له إلا في namespace main

- إحدى طرق كتابتها أن تأتي في داخل Form

منفصلة ب بنفس الأسم، أو تجمعها في صلف واحد . أو تأخذ

Form و تأتيه داخل main

subform بالنسبة لـ menustrip

Class Program : menustrip تعرّف و بناء قائمة

{ MenuStrip menu;

ToolStripMenuItem close;

ToolStripMenuItem file;

```
ToolStripMenuItem operation;
ToolStripMenuItem add;
ToolStripMenuItem sub;
public Program()
{
    menu = new MenuStrip(); // القائمة
    file = new ToolStripMenuItem("File");
    operation = new ToolStripMenuItem("Operation");
    add = ... ~ ... ~ ... ("Add"); // إضافة
    sub = ... ~ ... ~ ... ("Sub"); // حذف
    file.DropDownItems.Add(operation);
    operation.DropDownItems.AddRange(new
        ToolStripMenuItem[] { add, sub }); // إضافة مجموعات أدوات
    this.MainMenuStrip = menu; // على شكل مجموعات أدوات
    this.IsMdiContainer = true; // تعدد نافذة
    mainmenu = this.Controls.Add(menu); // حفظ المنهج
}
// سهل تدبر، الجميع واللوبي بكل عنصر
```

وليس مقتصر على الأدوات فقط تستطيع إضافة عناصر لها

```
List Box1.Items.AddRange(new string { "17", "9", "8" });
```

ـ Keydown ـ حدث

حدث Keydown عبارة عن اضطراب على أي مفتاح - وهو غالباً ما يدعى Keypress هو جسم أو طفاف يمليء ذلك أزرار المفاتيح، يستخدم لـ Keydown لـ Keypress لا يتعامل مع أزرار المفاتيح

this.KeyPress += new KeyEventHandler(mText);
 this.KeyDown += new KeyEventHandler(mText);

void mText(object sender, EventArgs e)
 { if (!(e.KeyCode > 48 && e.KeyCode < 57)
 || (e.KeyCode != 8))
 e.Handled = true; }

خاصية ClientSize أو الحصول على أبعاد المنشئ
المطلوبة للنموذج بدون الحواف والاطارات.

`this.ClientSize = new System.Drawing.Size(200, 100);`

لـ `Form` أو `Form` لا يأخذ دوراً لحواف ورلامارات، الخارج

ـ معناه الكلاس الذي أنشأ فيه `this`

ـ غير الشاء أداة تقوم بعملية تقوم بأساساً `new` جديد و

يختلف عن `ClientSize`، لما يحتوي عليه `ClientSize` والعمل

ـ بأمراته.

`class closeButton : Button`

```
{ public closeButton()
    { this.BackColor = Color.Red;
        this.Click += (s,e) =>
        { Form1.Close(); }
    }
}
```

ـ يقوم بعمل الأحداث واتخاذها، طلبات في المنشئ.

ـ دالة `Close()` على إرجاع النموذج الذي يحتوي على العنصر

ـ ممنوع في `oop` (=) في حالة الخروجة.

`class Distance < A , B>`

{ A Kilo ;

B meter

public A kilo

{ set { Kilo = value; } // set

get { return kilo; } } د واحده get

public B Meter

{ set { meter . value; } set

get { return meter; } د واحده get

استاد تستغل ، get

}

اداً مسخة لاجيل المودع على this. Perform Layout();
اعادة ترتيب عناصره (طابعه) لمحددة مثل الجداول
المؤمن بها، لارساد والمحاذاة.

ـ أداة عاديّة مثل `Form` ولما نظرنا لها كانتا `OpenFileDialog`
 «ـ نعرف `Form` داخل `Form` وليست لها «ـ مثل `Add` لها `Form`
 عند ما نسأله سيرها من سريحة الأدوات لا تظهر داخل `Form` وإنما
 خارجها لأنّه لا يوجد لها `Add`ـ

ـ أي أداؤه ينبع من `Form` هذه الأداة `OpenFileDialog`

ـ عند استعمال الأداة `OpenFileDialog` يتهيئ بعدها `OpenFileDialog`

ـ كمتغير عام وعندما نختارها نعمل `new` و `Show` لأخذ نتائج

ـ تأتي حيتاً من النافذة طوال فترة تشغيل البرنامج. بعد الانصراف

ـ من استعمال الأداة نقوم بتمريرها بواسطه `Dispose()`

ـ أي أداؤه هي عبارة عن علامة `Dispose()` أو `Form` أو `TextBox`

ـ تقدّر بـ `Dispose()` الأداة يواسطه `OpenFileDialog`ـ

ـ `FromArgb (red, green, blue)`

ـ دالة لاختيار الألوان بحيث يمكن تحديد كل لون من 0

ـ `Click()`

```
{ OpenFileDialog op = new OpenFileDialog();
op.Filter = "(*.Bmp|*.bmp)";
if (op.ShowDialog() == DialogResult.OK)
{ Bitmap b = new Bitmap(op.FileName);
Graphics g = pictureBox1.CreateGraphics();
}
```

```
g.DrawImage(b, new Point(0, 0));  
}
```

Drag & Drop صحيح وللأمثلات.

```
public Form()
```

```
{ OpenFileDialog.ShowDialog();
```

```
if (OpenFileDialog.FileName != null)
```

```
{
```

```
Bitmap b = new Bitmap(fileListBox1.Path
```

```
.ToString() + "\\" + fileListBox1.FileName.
```

```
ToString());
```

 لى المارجع، لاسم الملف

```
pictureBox1.Image = b;
```

 فتاجير Bitmap
ملف

```
fileListBox1.MouseDown += new MouseEventHandler
```

 ←

```
Handler();
```

```
panel1.DragDrop += new DragEventHandler();
```

```
panel1.DragEnter += new DragEventHandler();
```

```
}
```

```
// MouseDown ()
```

```
{ if (e.Button == MouseButton.Left)
```

```
{ string str = null;
```

 حفظ ناحية
مسار الملف

```
≤
```

```
str += fileListBox1.Path.ToString();  
str += "XX";  
str += fileListBox1.FileName.ToString();  
this.DoDragDrop(str, DragDropEffects.All);  
}  
}  
} DragEnter ← DragDrop DoDragDrop ←
```

// DragEnter()

```
{ string str = Data.GetData(DataFormats.  
Text).ToString();  
Bitmap b = new Bitmap(str);  
panel1.Image = b;  
}
```

عمل اللياً خذناه في المقرر كان عبارة عن كفالة التعديل على الـ Form وإضافة خصائص التي هي عبارة عن بيانات وإضافة دوالات، التي هي عبارة عن أحداث لذلك التعديل على السابق override، override هو أنه احنا نكرر على الدوال السابق بجيت أنه ٩٥% من عملنا هو تكرار على السابق.

إذا حصل خلل في السابق (وكان قد عملنا بها نسبة) ونعد له في الدالة trycatch والأولى تحصل كما هي.

الفكرة هي، لاستدعاء التعديل على السابق ليس ~~يكتبه~~ معناه أن تلفي الأول فقد تأتي في شرط من، سر وطر نسبة يعني الدالة كما هي قبل الإضافة أي تتبعها، القاعدة، النازلة، واتنة، الدالة الأولى.

في الـ override تكتب الدالة كما هي ~~شكلاً~~ ومن دون أي مراجعات، ونوع الدالة داخل الدالة نسبة يعني الدالة، السابقة لكن يشترط، لـ this حان فيه خلل في الدالة.

السابقة استثنى «اتساع الخلل واتنة نفس الدالة»، فوق نسبة ٥٥% فتخرج إلى الدالة السابقة.

طريقتان لاستدعاء Form:

this، إنشاء Form من Form، استوافقاً.

نذكر Form وأهل لها إضافة للـ Controls، بينما نعرف Form Controls، الأدوات ونضيفها لداخل مجموعات، Controls التابعة للـ Form.

«ولـ Form هنا عبارة عن Object».

في لغة C عند استدعاء، لـ الدالة، لسابقتها
اسم الدالة :: **اسم الدالة** (الأب)

في لغة C من طريق **المكتن** **base**.

في لغة Java

Super. **اسم الدالة**

في درس **override** إذا أدخل خلل في السابقاً استقراً وأبدل بالليست
والسابقاً يدخل كما هو.

- قد تكون الدالة لا تتجاوز سطرب برمجي لكن قد تكون هناك قيمة
تسبي، لهذا فتفحص قيمة كل في تعريف الدالة.

لورانس هي إحدى أهم ميزات OOP.

الاستعاق هي مفتاح من مفاتيح تعديل الأدوات.
ستحتاج إلى تطوير الأدوات، التي نعدل عليها أو نشير إليها حتى
نجعل الأداة عامة عندها مستوردها من قاعدة البيانات

{ Form f = new Form();

Button b = new Button();

f.Controls.Add(b); } } **النوع المذكورة**

صيغة إنشاء Form

. SubForm معارة عن Panel

<< user Control >>

عما يُعرف بـ userControl

ـ الأدوات مبنية على Control

ـ Control وهي أكبر من Array مبنية على Controls

ـ يمكن استخدام Control أو userControl لكي تمتلك

ـ أدوات مبنية على Control

ـ userControl ي繼承 panel لـ panel

ـ الأدوات تغيرها في panel userControl وهي مبنية على

ـ بحسب تطبيقات برمجيات

ـ panel ي繼承 Form وـ panel

ـ اعطى مبنية Create Graphics

ـ this.CreateGraphics على أي رسالات

ـ panel1.CreateGraphics

ـ panel في ~ ~

ـ

ـ وهذا =

class ExitPanel : panel // = userControl = Control

{ Button B1;

ExitPanel ()

{ B1 = new Button ();

this.Text = "Exit";

B1. Click + = Foo;

B1. width = this.width - 2;

B1. Height = this.height - 2;

B1. Location = new Point(1, 1);

this.Controls.Add(B1);

this.Resize += REF; } و

بعد تنفيذ وقمنا بسحب الأداة داخل Form ، الأهم أننا ندخل

باسم الأداة منها 1 PanelExit لكي هنا تتطلع Exit زرته

سهي تنفيذ أسلوب في اللحاظ ونظهر الزر مع . سعى

وهذه المراحل الأولى للتنفيذ بالنسبة لـ B1 ، لـ B2

المراحل ، لـ B3 للتنفيذ على تشغيل المشروع بيئة ، كل ما حصل

«أثناء التنفيذ يتم تنفيذ جميع الأكاسات الموجودة»

يبدأ التنفيذ new Form();

عن ما يشوف new بروح الكهابستير .

في المسند بالفتح ، الأدوات new كل أداته

كل ما يلعن new يذهب وبنها المسين يبعده

وهذا بالسيئه كل أدوات .

في هذا ، الأكاس لـ B1 زر داخلي Panel والمطلوب أن يكون زر

بـ مساحة ، new Panel();

نعمل سعى ، لـ B2 نفسه new أو العكس .

عند تغيير الأدوات يغير Panel والزر يغير كما هو والأمثل أن Resize يتم في هذه عناصر (حد) حد تغيير حجم Panel.

حدث، لـ Resize من الأحداث التي تتعقب بعد المسحية مباشرةً، ويتم إنشاء الأدوات بعد ها يتم رسم كل أدواته.

حدث هو الـ Load الذي يوجد في باراميتر الدالة وهي التي ترسل الحدث.

جود حدث لـ Resize ي العمل في التفعيم وعنة زلستينه، حدث لـ Resize مفهوم في المفاهيم على تشبيه الأدوات في عند تغيير الحجم، Form.

// تابع الكلاس //
// Load ()

{ // Public الوصول للأدوات إذا أقيمت
ExitPanel.B1.Texts "click" ;
ExitPanel.B1.click += Fcl ; }

Void Fcl (,)

{ =
= }
= {

// الوهمول (المحمولة) عن طريق دوال لـ تعليل

```
public string Text
{
    get { return BL.Text; }
    set { BL.Text = value; }
}
```

// ربط حدث

```
public EventHandler mclick; // delegate
this.mclick (this, sender, e);
```

// Load() دخل

```
{ ExitPanel.mclick += Fcl; }
void Fcl (object s, EventArgs e)
{ this.FindForm().Close(); }
```

3

CSC C Sharp Compiler

طريقتي لكتابه (كتاب) هو طريقة لتأليف طريقة الاستفادة

// In Java *Call this line* اداً من هنا

class Form1 extends JFrame // implement ActionListener
{ JButton B1; // ساختة بانتر سير //
JTextField Tbl; // ج تفاصيل المكتبة فقط.

Form1 ()

{ B1 = new JButton ("OK"); و

this.getContentPane().add (B1); و

// إضافة العنصر لل فرم بال نسبة إلى أنا كافية معرفة

// C* Controls د.

B1.addActionListener (this);

// إضافة حدث ويختلف عن C* يختلف في دلائل اضافة

// الحدث باسمه = + سباق إيجاد باسمه الدوال كازها عارة

// عن oop بالاتي كله عبارة عن

// > الدالة Add ActionListener تحتاج object وللائن داخله الدالة

// فاما نعمل له this معناه أن الدالة تكون داخل (كلاس) أو نعمل

// كلاس خارج

A a = new A (); و

B1.addActionListener (a); و

لو نستوي فريدة أكثر دالة حدث عدد الأحداث س كلاسات مختلفة

B1. setTitle (" _ "); // = B1 Text interface

- أصل ما في بطاقة بحدات

- أي حدث يمكن له اسسه عدد

class A implements ActionListener // = interface

{ Action performed (ActionEvent e) // →

{ // e.getSource() → *(e.getSender())*
// . Sender لـ الذي أطلق منها،

Tb1.setText (" _ ");

B1.setTitle (" _ ");

=

{ // *this* this يكون داخل دالة نفسها

{ class Action Listener هي نفسه، الذي داخل Delegate

- تركيبة هو نفسه، لما يكون صنف دالة، مثلاً

. يأخذ حقاً ملائمة

interface ActionListener

{ void Action performed (ActionEvent e); }

هذا هو Delegate

- هي بالفعل موصّلة، فما تكون واحد من الأصل

- يكتسب الذي معه و ما شئ عليه، يريد

- أني حاجة داخل ياتلاقا أو بار ~~C~~ فروع عبارة عن interface مُؤشر دالة،
يترجم مُؤشر دالة لما شئناه (من الأساسي والأساسي) يأشر على
التي تمت (الجديدة).

سواء بـ Chiled Base يأشر على دالة،
يقلت دالة ويرجم كأنه مُؤشر على دالة.
أو فيDelegate دالة وما داخله يستغل يعزم دالة،
معروفة في النظام أو ما شئنا.

Add ActionListener (ActionListener obj)

```
{ // click() / - هنا دالة Click حقنا بدل ما نعطيها اسم الدالة
    { obj.ActionEvent() - نعطيها دالة ملي فيها دالة
        // implement لـ دالة معاشرةً داخل دالة
        // ونستخدم دالة معاشرةً داخل دالة
        // وول زبه هنا سلاح أين يكون دالة من حيث مسئو
        // بـ interface دالة
```

هذا دالة Delegate دالة وليس دالة العادي
نعرف دالة ونعمل دالة دالة، نستوي دالة من ذلك وارتبط
من ذلك معروفة دالة دالة تعرّفها نفسها تعرف دالة
interface دالة حاسمة ياستخدام Java

والـ interface تعرّف دالة واحد وداخله عدة أحداث والـ obj

برهنة الحالات تقدر تقليلها باش علىه اكتر من حدث داخل الكلاس نفسه

// نكهة لدالة لسابته في حال أرسلنا 2 قيمة

if (e.getSource() == B1)

{ = }

else if (e.getSource() == B2)

{ = }

لو نسي نرجع للأحداث المترتبة بالحدث نستخدم `(ActionCommand)`

ترجع له نفس الأسلوب المترتب بالحدث

لور حاصل على الـ `getSource()` يجيب للعنوان لـ `Button` ما ياخذ عار يرجع لكائن

لور حاصل على الـ `getSource()` موسّر الـ `Object` ما يتقرب لـ `Button` ليس له ملحوظ

هي فالى القدرة على `Interface` حيث إنها تتعامل مع الدالة الموجودة فيه

ونتيجة لـ `Button` الموجودة بالأساس الدين

حيث ترجي `Interface` لـ `Text` `Box` التي تسرها حيث يرجعها موسّر الدالة

حيث تكون ذر عنده الفتح على كل مره ينشأ نقط

عندما يفتح على غيره أخرى ينشأ عشرة `Text` `Boxes`

إذاً معانا `Text` `Box` سابقاً وعرفنا `Text` `Box` جديده وقلنا أن

الجديد \Rightarrow السابق هنا يأش على `Button` لا يأخذ خصائصه هناك

دالة `Button` تأخذ خصائصه أو تأخذ الموضع نفسه بالكامل في

الذاكرة وهذا تفع مشكلة لأنها تليق بـ `Button` \Rightarrow `pointer`, `Image`, `Image`

في معانٍ أسلوب `Properties` تصل له هذه عمل الخصائص الذي

عمله وتنقله مباشرةً للآخر، نستعمل `foreach` لـ `TextBox` كل حاصل على `T`، لا ولد إلى `T` يجد.

// Click()

{ TextBox T = new TextBox();

T.Text = "100".ToString();

T.Location = new Point(1, 1);

this.Controls.Add(T);

} // هنا كل الأحداث تمتلك `T` ولذلك يمكنها إيجاد `T` فقط بـ `foreach`.

// خروج خارج عبارة عن Local

- هنا `new` ينبع من `Controls` وهو `new` ولذلك هو الموضع فقط للأخر

. `T.Dispose()` نقل له (إلا إذا)

فإنما أشار إلى الملف من حيث يعنيه يعني الموسّر وليس الملف.

فإذا أشار الملف من حيث يعنيه يعني الموسّر وليس الملف.

لما يتعلّم `Controls.Add` هنا يختلف الموضع.

`new` هي الأساس والتي تكون موضع `new` الأساسية هي موسّر إسماً الموضع.

- في حدث `Click` إذا قطعنا صرحة أخرى بـ `Top` فوق `100` وبالتالي

لا يمكننا خطتها لذلك نقيس للسابقة.

int Tp = 1 // متغير عام

T.Location = new Point(1, Tp)

~~Tp Height + Tp = T.Height + 5;~~

اذا احتفظت قيمة عشوائية

Random Rd = new Random();

Rd.NextDouble() * 100.ToString();

- لوستي ١٥ بثمرة واحدة

for (int i = 0; i < 10; i++)
≡

- لوكان العدد حسي ادخل المستخدم

int n = Convert.ToInt32(textBox1.Text);

for (int i = 0; i < n; i++)
≡

لكن لومنحك عنك ادخال، مستخدم ١٥ شئ، وادخل سرة
آخره ٥ شئ، فيت الساقط ما خلوا ارتدتا عن كل مرة ادخال بيستي
يعني العدد المدخل قمة هنا خارج فعل لا ادخال الحدود اكبر من
السابق او اقل ؟ اذا كان اقل من السابق فقم بالخطوة بعد ادخال
الفارقا وادخال اكبر بزدين بمقدار الفارقا.

const int MAX = 100;

TextBox[] T = new TextBox[MAX]

int count = 0;

int tp = 1;

// click ()

```
{ Paint ns Convert.ToInt32 ( TextBox1.Text );
```

```
if ( n > count )
```

```
{ for ( int i = count; i < n; i++ )
```

```
{ T [ i ] = new TextBox ();
```

```
T [ i ]. Location = new Point ( 1, Tp );
```

```
Controls . Add ( T [ i ]);
```

```
· Tp += T [ i ]. Height + 5; }
```

```
count = n; }
```

```
}
```

```
else if ( n < count )
```

```
{ int a = count - n;
```

```
for ( int i = count; i >= n; i-- )
```

```
{ T [ i ]. Dispose ();
```

```
Tp -= ( T [ 0 ]. Height + 5 ) * ( count - n );
```

```
count = n; }
```

→ this . Controls . Remove (T [i]);

```
}
```

يُسمى مثلاً رقم 2 الموجود بالملف
this.KeyPreview = true;

تشغيل دل Keypress على Form حتى لو وحدة Form
(إذا Form ما يشغل دل Keypress حتى لو قليناها) False

لو عدنا لـ Form على دل KeyDown

في حدث دل KeyDown

ذر A ينزل صار

ـ 5 ينزل بيب

ـ D يروح الكارت المفقـ

كلمة الماهمة ٨

مدونة المدرس:
التاريخ: ٢٥/٩/٢٠١٧

العنوان:

العنوان: ١٤٤٢/٨/١٨

أي أداة سعيرها وعادلها بالقوس «تطلع قاتل» Timer Add OpenFileDialog

وـ PictureBox وـ Panel Form Add OpenFileDialog
وـ Form عبارة عن Form Add Global variable
وما زاد ما يعما Add Form وما زال يطرحها في

TextBox Text Button Form Add Form Add Form Add
وـ Form هو عبارة عن أداة.

أداة عبارة عن Button أو Form Form Add Form Add
لوكان معنا Form Add Form Add Form Add
أداة ونكون object ونعمل معه.

صغيرة لا تكون

صغيرة أداة Button - TextBox أداة Mouse RGB اللون
 تكون منطقه بالطبع عبر والcold تكون مبنية على كل لون

السيء والآلات

النوعين السيء، السيء - السيء من داخل المشروع - السيء من خارج المشروع
 MouseDown Drag Drop آخر

Mouse DragEnter - Mouse Up

أولاد، لعب، والألعاب من بين ٣/٢ بالمئة، الماهمة ١٢

user control و control في Form هي عقارات مسطحة رسملة كل لاملاحة قد تكون صورة أو سهلة، (رسملة معناه)، Image حقاً لاملاحة قد تكون صورة أو سهلة، (رسملة معناه قبل لاملاحة)، فإذا وصلنا إلى MouseOver للدالة نغير نعمل على ترتيبه.

نأخذ المؤشر حقاً لاملاحة ونعمل على Bitmap ورسملة أو يمكن استخدام Image.

هذا المؤشر ليس سويسراً رسملة وإنما عبارة عن هدف يحتوى عليه مؤشر هذه لاملاحة.

Graphics g; // و g يحتوى على هدف الرسم وهذا الذي يزيد صفات لاملاحة للرسم

g = this.CreateGraphics(); // إنشاء لاملاحة للرسم

g.DrawLine(,, , ,); // دالة DrawLine لرسم خط

Paint() حدث، (رسملة

حيث أنه يتضمن الحجم أو الموضع بعد رسمه

Paint, (رسملة صياغة) يأتى بعده حدث، (رسملة

يغير دماس تريل، لاملاحة علىه، لـ Form يرتسم

لـ Bitmap لاستعمال بكترة في (رسملة لـ Bitmap)

public enum ShapeType

{
 Rectangle,
 Circle
}

```

class MyPaint : UserControl
{
    ShapeType shape = ShapeType.Rectangle;
    [Description("شكل"), Category("Appearance"),
     DefaultValue(ShapeType.Rectangle),
     Browsable(true)] // ←
    public MyPaint()
    {
        this.Paint = new PaintEventArgsHandler(Paint_Shape);
    }
    public ShapeType Shape
    {
        get { return shape; }
        set
        {
            if (shape != value)
            {
                shape = value;
                Invalidate();
            }
        }
    }
    protected override void OnPaint(
        PaintEventArgs e)
    {
        base.OnPaint(e);
        Graphics g = e.Graphics;
    }
}

```

Rectangle rec = this.ClientRectangle; //

، سم) ، الطول و العرض حفاظة ←

rec.Width = 1;

rec.Height = 1;

switch (shape) {

{ case ShapeType.Rectangle:

g.DrawRectangle(new Pen(Color.Black, 1),
width, height → rec); break; } المعلمات

case ShapeType.Circle:

g.DrawEllipse(new Pen(Color.Black, 1),
rec); break; }

}

}

[STA Thread]

STA → Single Threaded Apartment

ـ البرنامج يبي أنيماليه نوهج خطوط مفردة

ـ Main لـ فوق و تكتب فوق



ـ بداية مشروع بسيط لحماية الصور:-
ـ موجود في المزمنة مع الكود.

- ماتم ذكره عن كيفية عمل الـ Form
- جعل الصورة فاتحة (Light) او غامقة (Dark) عن طريق إنشاء أدوات للتحكم بذلك كـ مفكرة الصورة بعد لحماية.
- ياتي الأشياء لضروريته ليجعل الـ Form يعمل بشكل مثالى.

* قاعدة مهمة:

- عند العمل على أي نظام واجهات يتم وضع المسؤول التالي في حين لا يعتبار "كيف سيكون شكل لوحة الرئسية" وهذا يعتمد عليه child (Parent & child) حيث

* Windows للتعامل مع الـ forms

ـ هي = 1-Tree

ـ 2-List View

ـ هذه المقدمة تعتمد على الـ key للتعامل مع كل حقل.

الجيل الجديد

ـ Grid



- listview لـ listview استخدام ويمكن اسم النوع.
- View listview

(Toolstrip) أداة عمل يتم فتح *

- (Statusstrip) أداة
- استخدم منها لعرض بياناتي:
- ١- اسم, العنوان من لدخول.
- ٢- الوقت, حالتي / وتحت دخول, بياناتي.
- لادا تحتوي على *
- DropDown Button - Progressbar - StatusLabel)
- (SplitButton

ملحوظة - *
- هذا كان لدينا 3 بيانات تحتها نقوم بنفس, لهم

- ١- هم ربط حيث لادا لادا لادا حيث لادا لادا.
- ٢- ونحن استعملنا ذلك, حيث الادواة, الادواة في لادا لادا.





مشروع كود بخطه (فتح ملف) *
 (Form1.cs) = "open" (فتح الملف) click on -
OpenFileDialog fileopen = new OpenFileDialog;
 if (fileopen.ShowDialog(this) == DialogResult.OK)
 ↗ نوع قيمة من نوع
 DialogResult

لما نفتح الملف فـ (Form1.cs) يفتح الملف

1.1. Form1.cs newfile=new Form(); Form1.cs

فيه يتم تعيين بعض القيم له مثل،
 newfile.ShowDialog(); try catch (Exception ex);
 وغيرها لكن دا ما يهم

try { } catch (Exception ex);

newfile.MdiParent = this; → (newfile)
 newfile.Text = fileopen.FileName;

newfile.show();

3.1. catch (Exception ex);

catch (Exception ex)

{}

{}





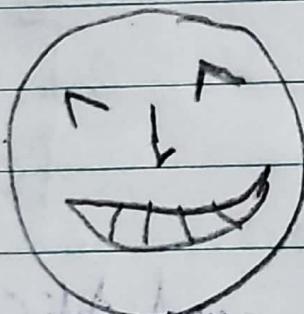
٢٠١٣ مخطوطة

يتم ترتيب موافع لذوات داخل (load)

نظام *

عمل آدأة لبعض مفعلاج وحيث من الصورة الآلة كلها مكتابي.
عن لنقر على الماوس و لسحب في مكان معين على الصورة
يظهر شكل سريع شفاف (مثل كرة لنقر و لسحب منها مكان فارغ
على سطح المكتب فإذا يظهر سريع شفاف عنده بـ فلاش
يتم أخذ هذا الشفاف بأحد

لـ دكتور فهد فالله يقول أن ليومية البريئة أو ليومية
طغمة سهلة جداً لأنها مجرد خنزير وليس سحر



Fact ← حقيقة



= PictureBox و ScrollBar في المنشئ *
= (Form6) أداة تشغيل، تزويق، تمرير

Private void VScrollBar1_Scroll(--, --)

{

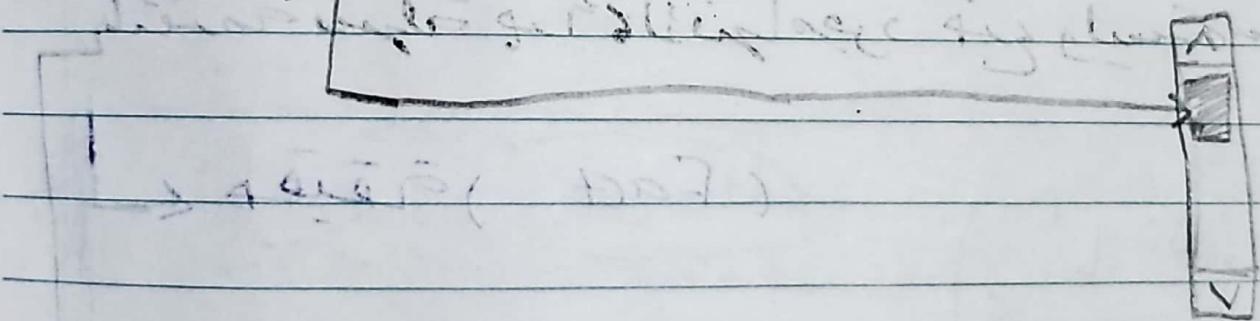
 PictureBox1.Top = - VScrollBar1.Value ;

}

- نشتري غرفة أداة سبعة ماسناد، لـ (الـ ٦) على (الـ ٣)
 ـ تشغيل (VScrollBar1) بـ (Top)

this.VScrollBar1.Scroll += VScrollBar1_Scroll;

(Scroll Box) (لـ (Top)، (Value)، (Move))



- من خلال الرسم لـ (Top) يكفيه (Move)، لأن فتحهم تالية (تشغيل)

= PictureBox و ScrollBar في المنشئ

ذلك = عندما تغير (Value) يـ (Move) (Top) -

سكونه = (PictureBox1 = 0). (Value = 0) -

لكن عند سحب scroll Box إلى الأسفل يـ (Move) (Top) (Value) -





فإذا أخذنا قيمة `Value` و `LargeStep` يساويان معاً
فإن `Picturebox1` يتحرك بخطوة `LargeStep`.

يمكن هنا أن نفترض أن `scrollBar1` له سهل
قيمة `LargeStep` هي `1`, لقيمة `LargeStep` هي `1` من `Horizontal`
- `Maximum` يعني أن قيمة `LargeStep` هي `scrollBar1`,

لذلك يتم تحديدها في `LargeStep` من مثلك، ولكن:

`vScrollBar1.Maximum =`

`Picturebox1.Image.Height - Panel1.Height;`

* أهم لخصاته، التي شجع `scrollBar` على تحريرها ← `Value` -

صغار، لزيادة أو لتفهان عن `SmallChange`
أو لزر `Up` أو لزر `Down` ← `TabIndex` -



* مثل، لو سُئلناه، لمسنا بقعة، لي تعااملتنا ببعضها مع
ذلك، يشكل عبوديًّا كأن تعاامل بعضاً،
لي يشكل أفقها لكن مع اختلافها
بسقطة، ستدرج فعل فيها في المزحة.



(Save)

* شرح كينة مفظ لصورة لي في Form 6.
ذلك إذا كان لدينا شرحة (Form 1) مفاسخ، ونحن نشغل
Form 1، لي داخلها، لي سؤال، لي داخلها، لي داخلها،
عن طريق

ActiveMdiChild

—أكود في المزحة →
Form 1 (داخل).

((Form6)this.ActiveMdiChild).pictureBox1.

هذه لغوية على أنها كذا
نقدر نوصل للـ (pictureBox1)
ـ داخل Form 6

* يتم الـ *ctrl + C* كـ *Copy* لـ *Cut*
و *ctrl + V* كـ *Paste*



٢٠١١

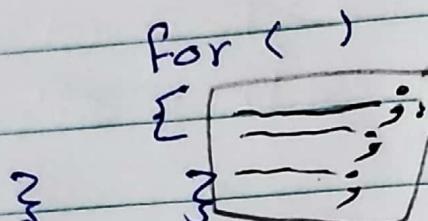


dark

Light

* فكره كييفيه جعل الصورة خاتمة ونهاية :-
الكلمات المفتاحية لصورة ستحتاج لها لوب مدخل :-

For ()



عملية حسابية تطبقها

Pixel \rightarrow ل

الملف وسيلة لتخزين البيانات.

Filestream → Read, write

C++ صدر في تغذية البيانات، يوجد ٤ طرق في XML

معامل مع ملفات XML

- توزيع البرنامج على تغذية وليس برمجي

- إذا كان سطح برمج وبكل مرة تفتحه وتتغير من اعداداته وتريد

- عند فتحه مرة أخرى يكون بنفس الاعدادات عنه (غلقها آخر مرة)

إما تكون الأعدادات بملف

في هذه الحال دوال تتعامل مع الملفات.

File stream : F = new Filestreams (_____)
فرو (_____)
_____ اس. ملف الفنزيلائي ↑
فراء - حركة ~

F.read F.write

stream Reader للقراءة فقط

stream Write للكتابة فقط

Text Reader → F.ReadLine

Text Writer → F.WriteLine

ـ القراءة حتى نهاية السطر ، أو يابا

ـ المراة حتى الملاحة . ، القراءة بحجم مجموع

- اليمكانيات ، المويودة في ال Stream writer ليس موجودة في

ـ Reader وهو التعامل مع الملفات

ـ كل ما في Stream Reader، ـ مفهوم Reader

```
{ TextWriter FW = new TextWriter ("MyFile.Txt");
    string T;
    while ( (T=Console.ReadLine ()) != null)
        FW.WriteLine (T);
    FW.Close ();
```

```
TextReader Frs new TextReader ("MyFile.Txt");
while (!Fr.eof())
    T = Fr.ReadLine () != null;
    Console.WriteLine (T);
```

ـ يمكن استخدام Stream Reader، ـ يمكن استخدام Stream Writer
 لـ ـ يمكن استخدام Stream Reader & Stream Writer
 معاً ـ معاً

ـ ـ نعم، لشيء واحد قيل للأخرين بالطريق