* استخدام الـ (Fn.) للتحكم في الـ (variables)
يسمى (interface) !!

oop (1)

Class car {};

هيزة الـ (Encapsulation)

car C1; // object

~~C1. CurrentSpeed = 20;~~

غلط استخدام اسم (variable) ..
عشان لو غيرت الإسم هحتاج أغير ده كمان ..

لكن الصح اتحكم في الـ (variable)
عن طريق (Fn.) !!
والـ (Fn.) دي بتستخدم في كل مكان
في الـ (code) بقى ..
ولو عدلت اسم الـ (variable) هيتعدل
مرة واحدة كمان بس في الـ (Fn.)
اللي بتتحكم فيه داخل الـ class {}

Ex: C1.Accelerate();

* افهم فكرة ال (streams) في ال (C++)

* ال (Cout) وال (cin) عبارة عن (objects) من ال (io stream) :)

Solid principles

من ضمنهم ← open For Extensions , Closed For Modifications

يعني أقدر أضيف عليه (code) جديد بس متعدلش على القديم --
وده فكرة ال (inheritance) :)

* مش لازم تحط (typedef) عشان تختصر ال (struct) :)

Dynamic Allocation in (C) vs (C++)

```
struct Employee *Ept;

Eptr = ( struct Employee*)
```

مختلف عن ال (Value) وال (Address) ---
عبارة عن ميزة من ال (C++) :)

4| pass by Reference

* مشكلة ال (By Address)
أن كتابته معقدة بسبب
ال (Syntax) ← (*, &)

B	3
A	5

* وعشان كده ظهر ال (By Reference)
بيكون عبارة عن (Alias)
لل (variable) الأصلي --
مش بعمل (variables) جديدة أو أخد نسخ

مش بيتعملوا مكان
في ال (memory) ---
عشان مجرد (Alias) :)

* فتلاقي أن ال (memory Address)
بتاع (X) و (A) زي بعض :)

الرمز بيكون (&)
في ال (C++) :)

هنا جملة جديدة :)

swap (int& X, int& Y)

Reference to an
an integer

دي الزيادة الوحيدة على
(code) ال (By value) :)

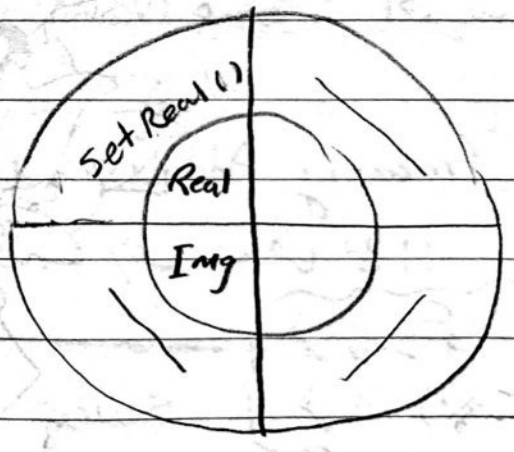
* ممكن نستخدم (String) بدل (char) في ال (C++)
ابحث عن الفرق ما بينهم ☺!! ..

Struct → Default is **public**

Class → Default is **private**

Class to represent imaginary numbers

sqrt(-1);
↓
5 + 3i



26/10/2022

2 opp

ال (C++) مش (pure opp) عشان تدعم

ال (Backward compatibility) مع ال (C) ...

هتلاقي إن عشان كده متاح ال (global variables) في ال (C++) :)

D1-Task

* مدام أنا بستخدم (member Function)

هتقدر تأ (Access) ال (private Attributes)
بتاعت ال (Caller) ..
وأي object تاني معاها يقدر يوصل
لل (private Data) بتاعته بردو :)

* لو هستخدم (stand alone Function)
متعرفة برة ال (Class) ..

هحتاج استعمل جواها ال (getters)
عشان أوصل لـ (private Data) ..
غير كده هيبين (Error) !!

هام جداً جداً !!
بتيجي كتير في
ال (interview) !!

* لاحظ أن ال (Fn. overloading)
ينفع في حالة ال (member Fn.)
أو ال (stand alone Fn.) ..

ال (compiler) بيفرق مابينهم
عن طريق ال :
(1) No. of inputs
(2) Data type

مش بيبص على ال Return type !!

الشرح دقيقة 40 ← (Make sure) !!

هتعتمد على الـ (compiler) هيقرأ من أول اتجاه
بين أوقات هتحسب (overloading)..
وأوقات هيبين (Error) !!

Ex

```
int MyFun (int x, char *str) {}
int MyFun (char *str, int x) {}
int MyFun (int x, int y) {}
int MyFun (int x) {}
void MyFun (int x) {}   XX
```

الـ (Return Type) مش هيعمل (overloading)

بالتالي هيبين (Error) !!

note → 53

عن ال (Reference Type) , ال (Value Type)
وعلاقتهم بال (C++) , (Java) , (.net)

* * * *

* ليه الكود كان شغال من غير (Constructor)
في مسألة ال (complex numbers) ؟!

← عشان ال (Compiler)
بيعمل (Automatically) ← (parameterless constructor)

ولكن
لو عملت أي (Constructor)
ال (compiler) مش هيعملك ال

Dynamic Allocation

```
Complex *ptr = new Complex (5,6);
```

* الـ (Destructor) الباسموندس لتحذمه تعليمياً

لازم يكون (public) !!

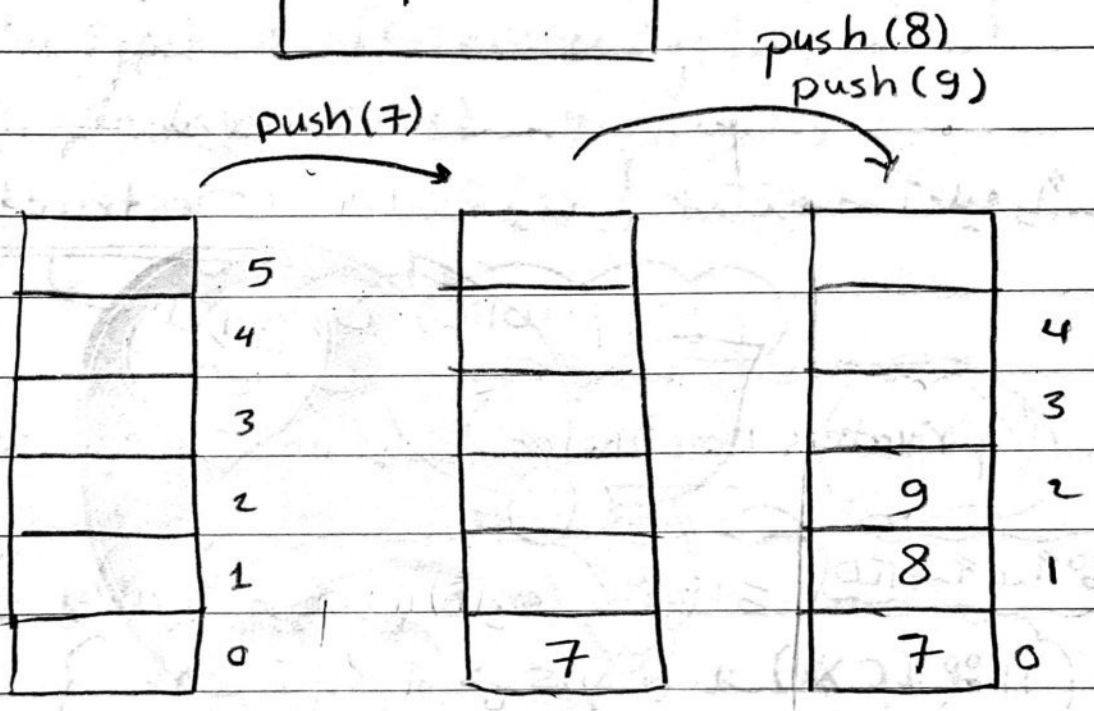
private constructor → فيه حالات كتير بنستخدمه فيها !!

هام جداً جداً !!

الباسموندس هيجي أمثلة عليه قدام أو في كورس الـ (C++) !!

1:30

OOP-2-2



Top OF stack

Tos : 0

Tos : 1

int stK [5];

int R = pop()

← في (R) هتبقى بـ (9)

بيجيب القيمة الي من فوق ☺!!

Full ? → Tos = 5

Empty ? → Tos = 0

this → Real

بتكون مكتوبة حتى لو
مكتبتهاش ☺!!

* لو محدد تش (Return type) و خليته (Void)..
فهيخليه ال (Default) ← (int)

* منيه (optimization) جوه ال (C++) بيقللي ال (Constructor)
من (6) لـ (5)..

هام جداً جداً ☺!!

* عشان تعرف ال (Constructor) ده بتاع أنهي (object)
يستخدم (this)
و (Track) ال (Address) ☺!!

* عشان تكمل (Test) لمسألة ال (complex)
بندخل الأرقام دى :) ←

3
-4
3
5

27/10/2022

Summary of D-2-OOP

called only once

Constructor → ctor

الإختصار اللى بيستخدمه الباشمهندس
فى الكلام عن ال (constructor) :)

Complex C1;

الجملة دى كافية لحجز (object) فى ال (memory)
فى ال (C++) ... ← عشان شغالة (By value)

على عكس ال (C#) أو ال (Java) اللى لازم تستخدم
(new) عشان تشغالين ب (Reference type)

complex C1 = new complex

فى ال (C++) حتى بنحتاجوا غير مع
ال (pointers) وال (Dynamic Allocation) :)

* الى بيحجز الـ (memory) هو الـ (Run-time) -
وبعد كده بينادى على الـ (constructors) عشان
عملية الـ (intilization) :)

in C++

parameterized constructor

1 stack → Complex C1 (3,4)

كده بقى عندى (8 Bytes in stack)
بتوع الـ (Real, Img) :)

2 Heap → complex *ptr = new complex (5,6)

(8 Bytes in Heap) :)

* إيه هى الـ (Clean up process) ؟!

(1) لما ينادى الـ (Destructor) فى نهاية الـ (code)

(2) لما يقفل الـ (Connection) مع الـ (server) بعد ما أخلص شغل :)

Cont. oop

Destructor

(1) no explicit Return type.

(2) Must be parameterless

(3) No overloading

Ex)

```
~Complex ()
{
    cout << " Destructor \n"
}
```

بعمل (Destructor) واحد

وبيتعمله (Call) لكل (object) اتعمل عندنا ؟؟!!

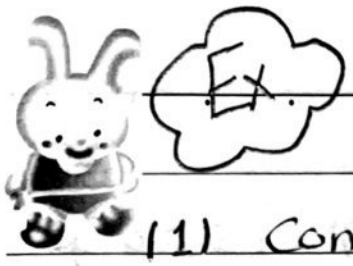
Cont. oop

* كتبت جوه ال (Destructor) عشان أعمل (Tracing)
لما يتم استدعاءه :) !!

delete ptr; ← في حالة ال (Dynamic Allocation) بستخدم

عشان اعمل (call) لل (Destructor) يمسح
ال (object) في الوقت اللي أنا عاوزه :) !!

private constructor? → بعمل (object) من جوه
ال (class) بس :) !!



Ex

(1) Complex *ptr ; ← (pointer) بيشاور على (object)
من ال (class complex) ;)

(2) ptr = new complex (5, 6) ;

بستخدم ال (parameterized constructor)
عشان أعمل (object) جديد يكون (Dynamic)
وبوصله عن طريق ال (pointer) ← (ptr) ;)

(3) delete ptr ;

calling the destructor to Free() ←
the space ;)

* عندى (array) عاوز الترتيب دخول و خروج العناصر كده ..

(First in → Last out) → ده مبدأ الـ (stack) !!

Ex

① int stK[5];

② push (7) → كده الـ (TOS = 1)

③ push (8) → بيبص على الـ (TOS) عشان يشوف أى الـ (values) الفعلية المتسجلة وإيه الـ (garbage) !!

→ كده الـ (TOS=2)

④ int R = pop(); → Last in .. First out

∴ (R = 8)

→ كده الـ (TOS = 1)

(5) push (13)

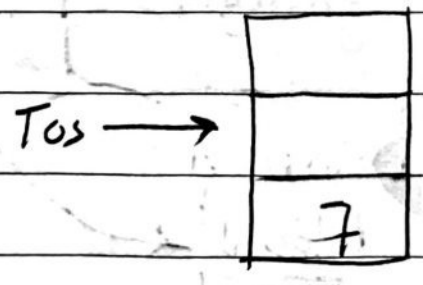
كده ال (Tos = 2)

Limitations

stack is Full → (Tos == Size)

" " Empty → (Tos == 0)

لاحظ أن ال (Tos) أزيد من ال (array) بـ (1)



(Tos = 1)

* طب لو عاوز أخلي ال (stack size) تبقى بال (user input) !؟

هينفعش احط ال (size) بتاع ال (array) ك (variable) !!

لـ ← فـ لازم استخدم (Dynamic Allocation)

لاحظ فرق الأقواس !!

```
My stak (int L)
{
    stk = new int [L];
}
```

int * stk

ده عرفته فوق (pointers) !!

this Allocates an array of (L) numbers of Adjacent integers

نفس اللي كانت بتعمله ال (malloc)

لاحظ أن دي الطريقة العادية لعمل (array of integars) لـ (Dynamic Allocation) !! في (C++)

note → new int (50)

بتعملك (variable) واحد وبتحطهولك بـ (50) !! (212)

عشان بيشاور على (Attributes) :)

void (Complex *this, int R)
{ this -> Real = R ; }

شكل الـ (Method) اللي بيعمله الـ (compiler) :)

هاتا جدة جدة :)

delete [] stK ;

دي طريقة عمل (Free) لـ (Dynamicall allocated Array) :)

* هل كل (object) لازم يكون ليه الـ (Data) بتاعته ؟!
يعني مع كل (object) هيتعمل (Dynamic Allocation) :)

functions

لو الـ (Methods) بتتعمل مرة واحدة
وبتستخدم مع (objects) ..

* طب إزاي بتعرف القيم بتاعت الـ (Data) بتاعته ؟!

بيتعمل (pointer) اسمه (this)
ده بيكون (Extra input) لـ (Method)
احنا مش بنشوفه :)

ده تحايل عشان نحقق الـ (OOp) :)

ولما احط (break point) على ال (call) ده هيظهرلي (this) لوحدها - وفيها ال (Address) بتاع ال (object) :)

* ومن ناحية ال (Fn. call) بيكون --

CA.setReal (&CA, 5)

بيبعت ال (Address) بتاع ال (object) عشان يتساوى بال (* this) :)

Complex * this = &CA;

ده الشكل النهائي :)

الكلام ده هيساعدني في تفريق ال objects عن بعض :)

هام جداً جداً :)

ممكن تكتب object name & في ال (watches) وهيورلي ال (Address) برده --

هيكون نفس اللي شاورت عليه ال (this) :)

* الكلام اللي فات ده بيحصل مع ال (struct) برده
بيكون عندي (this) !!

```
void Set Real ( int Real )
{ this -> Real = Real ; }
```

(*this) . Real

بعمل (DeReference)
لل (Address) اللي في (this)
فبوصل لل (object)..
بعدين بستعمل (.) عادي
عشان أوصل لل (Attribute) بتاعته !!

ده بيأخد (Zero)
من ال (clean code)
ودي (Code Readability) !!

D2- oop - Assignments

1

```
Class complex
{ int Real, Img;
  public:
      ctor (int, int)
      ctor (int)
      ctor ()
      Destructor ()
      int Get Real ()
      set Real (int)
      int Get Img ()
      set Img (int)

      print ()
```

compiler ال
عمل واحد مش نفسه

Copy Constructor

بس عشان مكتوب
في مظهر بيش من
ال Tracing

Copy constructor بيتعمل
عشان ال (pass by value)

بيعمل Destructor
لكن مش ظاهر
constructor ال

(1) (C1) F8

(2) (C2) F0

(3) (C3) e8

00

00

جوه ال sum

Fn input

08

e8

F0

F8

C2 → 08

الى جوه ال (Fn)
(inputs)

obj جديد
عشان يتساوى بـ (C3)

object جديد

```
[ ] complex sum (complex C)
{
    complex Result;
    return Result;
};
```

Result: obj جديد

محدد نتيجة (Return type) فبيكون الـ (Default) (int) :)

this: & C1

```
main ()
{
    complex C1 (1,2) , C2 (5) , C3;
    C3 = C1.sum (C2);
    C3.print();
}
```

* of objects → 6 constructors

* ctors

* Destructors

ممكن يبقوا (5) بسبب (optimization) في الـ (C) :)

مجرد هتطبع الأعداد دي ولو فيه خلل هتقول من مين :)

لمجرد (Tracing) :)

2 Class MyStack

```
{ int *stK, tos, size;

  public:

    ctor (int)
    Destructor
    bool isFull ()
    bool isEmpty ()
    int pop ();
    void push (int)

    int peek ();
```

يتورلك القيمة الى هتخرجها
الـ (pop) المرة الرجاية من غير
ما تغيرها !!
فكل مرة هترجع نفس الـ (value)
لحد ما تستعمل (pop)

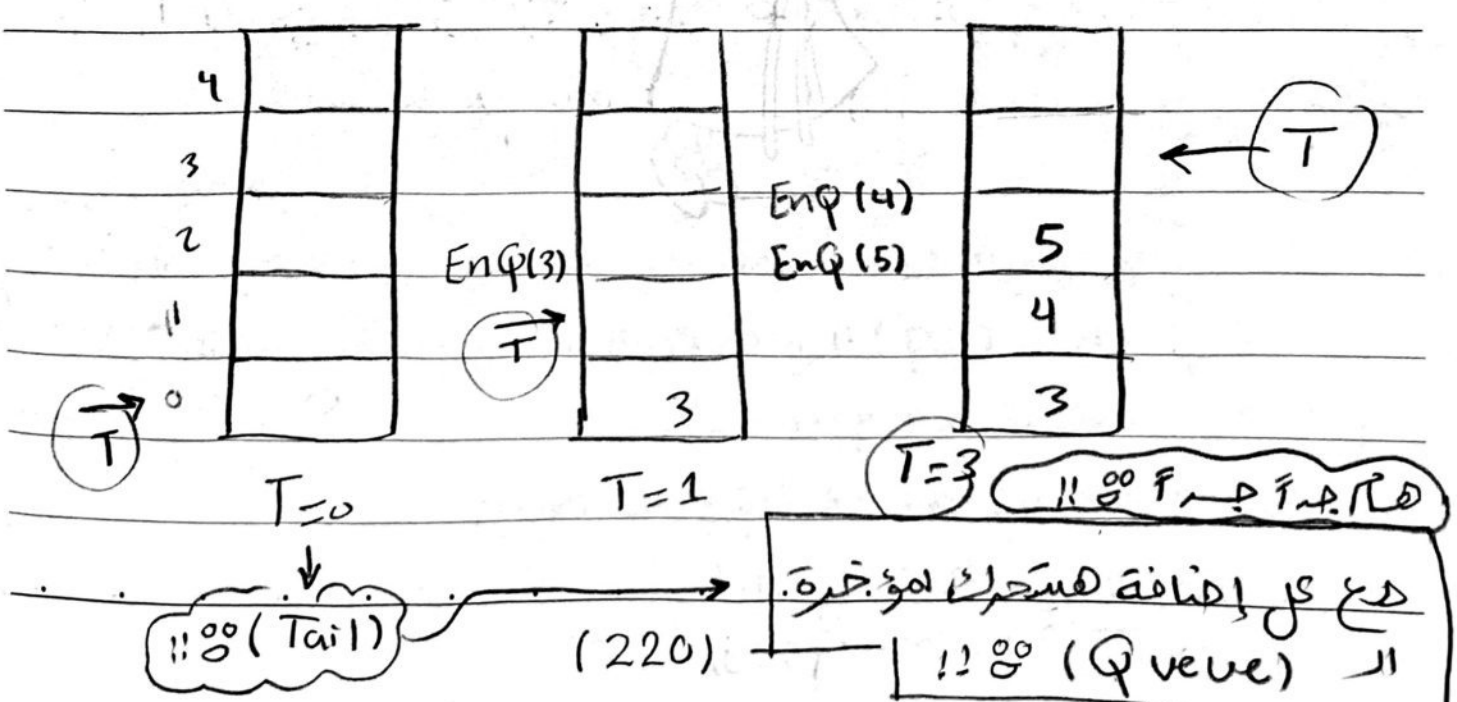
مجرد هنثبت
الـ (TOS) !!

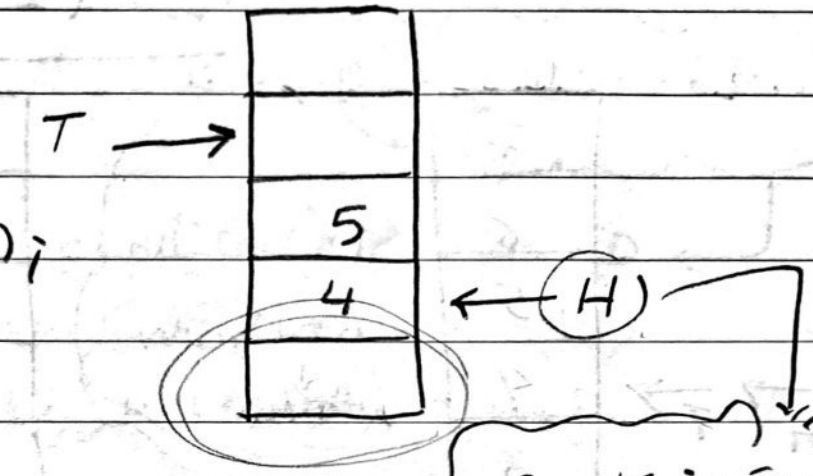
```
int GetCount();

}
```

هتقولي عدد ال (Elements) اللي عندي ؟؟
يعني هطبع ال (TOS) بس ؟؟

3 class Queue

```
{
public:
    cotr (int)
    Destructor()
    bool IsFull ()
    bool IsEmpty()
    void EnQ (int)   → input
    int DeQ ();      → output
    int Get Count();
}
```





cout << DeQ();

لازم يبقى عندى سهم
على الـ (Target) الى حوره
الجاى فى الدخول !!

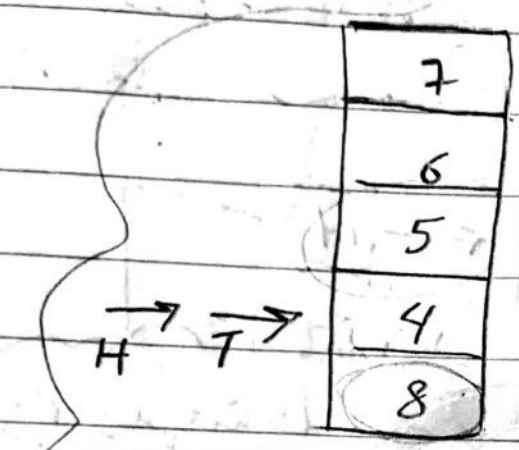
EnQ	→	T ↑
DeQ	→	H ↑

والحركة هتكون (Circular) !!

فى حالة (T) أو (H) == Size
بساويهم بالصفر تانى !!

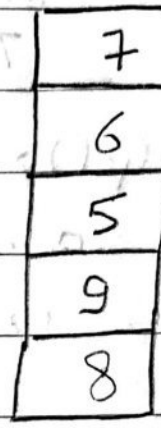
كود ال (circular) ← أصعب في ال (implementation) ولكن أخف على ال (memory) :)

يستخدم فكرة ال (2 pointers) :)



عملت (8)

وعملت (5) DeQ



DeQ()

EnQ(9)

اشيل (5) مرات

المفروض الباسمونتس قال مش شرط ال (T = H) عشان يبقى ال (Queue is Full) :)

Test case

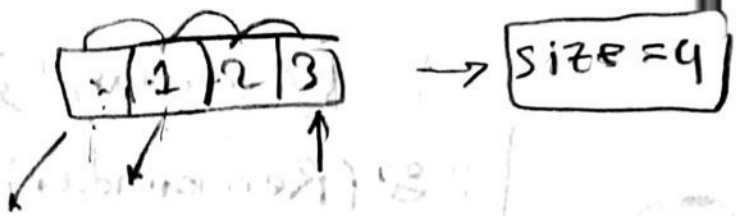
so.. (1) when Empty ?!

(2) when Full ?!

(3) How to move (H) and (T)

ال (Tail) فضل ثابت مكانه بيراقب ال (8)

ال (start) لف مع ال (5 * Eq()) ورجع بيساوي ال (Tail) ولكن ال (Queue) فاضية مش ملبانة :)



النسخة الثانية

هستخدم (Tail) بس ..
وفي كل مرة هسحب حاجة بال (DeQ()
هيعمل (shift) للأرقام وال (T)

بحيث أن دائماً الي عليه الدور هيكون عند
ال (index) بـ (zero) ::

الحوار ده أسهل في التنفيذ
ولكن تقيل على ال (memory)

* أي حاجة فيها (Loop) أو (Circular) --
خلي في بالك نقطة ال (Remainder) !! ^^

هنعتبر ال مفيش (Elements) هنشاور على (-1) --
بلاش اللغبطة اللي اتشرحت في المحاضرة ^^ !!

$((rear + 1) \% N) == Front$

array size

6	7	3	4	5
0	1	2	3	4

T (1) H (2)

$2 \% 5 = 2 == 2$

Full ✓✓

0, 4

5 | 2

$\frac{-0}{2}$

Remainder

بقلل ال (2) لرقم يطلع
رقم صحيح مش كان (0)

6	7			

H (0) T (1)

$2 \% 5 = 2 == 0 \longrightarrow$ not Full

5				

H T

5	4	

S T

when inserting ... we use the (Tail) ☺!!

3 Circular Queue Summary

1 When it's Full ?!

size = 5

0	1	2	3	4
8	9	5	6	7
	T	H		

(1)

* هنلاحظ أن فحالة عم بعد خطوة من ال (H) بكون (Full) ..

(2) فحالة الترتيب الطبيعي ..

(1) فحالة أن ال (T) رجع للبداية وبيعد تاني ☺!!

1	2	3	4	5
H				T

(2)

$\therefore \cancel{T + 1 = H}$

بس لاحظ أن ال (T) ممكن يـ (Loop) في هيعدى ال (Size) ..

$\therefore (T + 1) \% size == H$

هش هيعدى ال (Size) .. لما بيوصل للنهاية بصفره تاني

6	7	8	9	5
			T	H

هتديك الرقم الأصلي من 0→4 (225) ☺!!

* لما باجي بضيف بستخدم المعادلة السابقة بردو
عشان أفضل جوه الـ (Range) حتى لو هعمل (Loop) !!

Tail = (Tail + 1) % Size

الـ (Modulus) هيساعدك
في الـ (Looping) !!

الحتة دي لوحدها كانت هتبقى
(Tail ++)
لكن دي تطلعنا برة الـ (array Range) !!

بتزود الـ (Tail) عشان تجيب مساحة
فاضية وبعدين تخزن فيها بـ
Q[Tail] = n;

[2] when it's Empty ?!

(1) starting point

→ Head = Tail = -1

(2) Head = Tail

بشكل عام !!

			8	

H
T

بس بطبع ال (Element) المتطابقين
عم بعض فيه بعدين ارجع لـ (-1)

note → لازم أكون عامل (if) قبلها
عن ال (Head == -1) & (Tail == -1)

عشان لو اتحقق هيعملش حاجة
غير إعلان أن ال Q فاضية

هام جداً !!

حالة تساوي برده
ولكن (specific)
عم (-1) !!

3 How to (print the Quene) ?!

تتأكد الأول أن الـ (Q) مش فاضية !!

```
i = Head;
while (i != Tail)
{
    cout << Q[i] << endl;
    i = (i + 1) % size;
}
```

هتتجدد !!

يعتبر عايز تزود أي قيمة ..
لازم تعملوا بالطريقة دي !!

```
cout << Q[Tail] << endl;
```

لازم بطبع الـ (Tail) لوحده عشان
الـ (while) بتخلص قبل ما يتطبع !!

4 shifting Queue

كل ما اسحب حاجة من قدام ..
بعمل (shifting) لكل ال (Elements)
وأحرك ال (Tail) بردو لآخر (Element) :)

1. Full ? → Tail = Size - 1
2. Empty ? → Tail = -1

شبه ال (stack) :)

3. EnQ

بزود ال (Tail) عشان أقف على مكان فاضي
وبعدين أعمل (Assign) لـ Element جديد :)

④ DeQ

بعمل (print) للرقم الي عنده Q[0]
وبعدين استخدم (For Loop)
عشان أعمل (shift) لكل ال (Elements)

```
For ( int i=0; i<(size-1); i++)
{
    Q[i] = Q[i+1];
}
```

Tail--;

بعد ال (Element shifting)
بقلل ال (Tail) خطوة واحدة
عشان الي خرج (Element) واحد !!

الاثنين القادم الأسبوع الجاي
امتحان (٢)

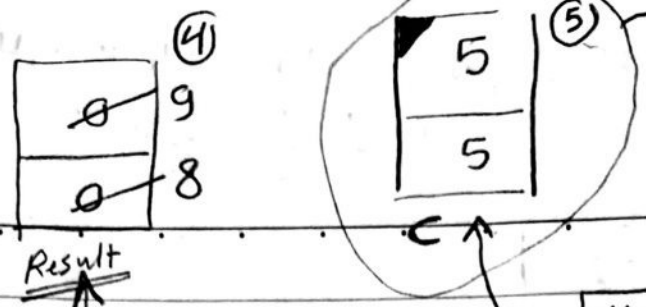
30/10/2022

OOP (3)

```
class Complex
{
    int Real;
    int Img;

public:
    Complex (int, int)
    Complex (int);
    Complex ();
    ~ Complex();
    int GetReal();
    int GetImg();

    void setReal (int);
    void setImg (int);
```

```
int Complex::GetReal()
{ Return Real; }
```

عبارة عن (Scope operator)
بيحدد أن ال (Fn) دي
تبع ال (class complex) ::

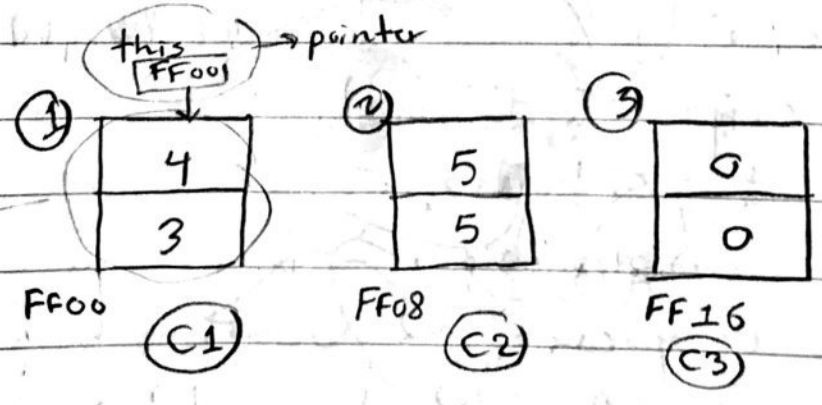


بيشتغل بطريقة اسمها (Bitwise copy) أو (member wise copy)

عبارة عن نسخة (By value) :)

```
Complex Sum ( complex c)
{
    Complex Result;
    Result Real = Real + c.Real
    Result Img = Img + c.Img

    Return Result;
}
```

محطوطلي (this) من غير ما اكتبها :)



each object is 8 Bytes :)

ال (operating system) بدل ما يعمل ال (object) ال (constructor) .. بيأخذ القيمة (bit by bit) و يسجلها في مكان ما :)

بيظهر له (Destructor) سي :)

ده ال (logic) بتاع ال (compiler) :)

6 constructors
6 Destructors ← المفروض يبقوا

```
int main ()
{
    complex C1 (3,4) , C2 (5), C3
    C3 = C1 . Sum ( C2)
```

10

6

9
8

Temp Anonymous object

ده ال object السادس

شرح جه مش --

ده عبارة عن Temp object
بيعمله ال (system) عشان
يسجل فيها ال (Return)
طب ما يسجله في ال (C3) ؟؟

RVO

Return value optimization

بنقدر نوقفها
من ال (Compiler)
عشان نعمل (Tracing)
حقيقي ؟؟

هام جداً جداً جداً !!

في حالة الـ RVO : ON
السطور المشطوبة مش
هتظهرلي !!

steps

1. Result (X) → عملية الـ (Const.) لـ (Result) من غير ما أقول ؟!
2. ~~member wise copy~~
3. Destructor Result
4. Dest. C
5. Assignment , C3 = Temp Annoynmous object
6. Dest. Temp Annoynmous object (X)
7. Dest. C3, C2, C1

```
Class My stack
{
    int *stk;
    int TOS, size;

public:

    My stack ()
    {
        TOS = 0;
        Size = 5;
        stk = new int [size]
    }

My stack (int L = 5)
{
```

لو مدخلتش قيمة
بيحطوا ب (5)..
فده بيسموه
الـ (parameterless) :)

```
}
```

هسأل جداً جداً :)

parameterized + parameterless
(At the same Time !!)

```
My Stack (int L = 5)
{
    TOS = 0
    Size = L;
    stK = new int [size];
}
```

```
int main ( )
{
    My Stack S1;

    My stack S2;

    Cout << MyFun01 (10, 20, 30); → 60

    Cout << MyFun02 ();
         << MyFun02 (10, 20, 30);
         << MyFun02 (10);
         << MyFun02 (10, 20);
```

int MyFun01 (int x , int y, int z) { Return x+y+z; }

int MyFun02 (int x=1 , int y=2, int z=3) { Return x+y+z; }

Not valid

~~MyFun (___ ,20) XX~~

مينفعش تنسش !!

* قيمة واحدة ← هيديها لأول (variable)
قيمتين ← هيديها لأول فيمتين
وهكذا ..

ممكن تجمع الـ مفيهمش (inputs) على الشمال ..
والباقي على اليمين ..

int MyFun03 (int x , int y , int z=3) { ____ }

int MyFun04 (int x , int y=2 , int z=3) { ___ }

Not valid cases

1. int MyFun05 (int x, int y=2, int z) { }

2. int MyFun06 (int x=1, int y , int z=3) { }

لكن ده مينفعش !!

ده بيشتغل على الـ (array)
بتاعت الـ (object) اللي اتعمل
(Destructor)

ممكن كان (password)

وبكاني مسحتوا - عشان محدش يعرف
يوصل للقيم اللي كانت متسجلة هنا

هنا حد أحب أ

```
~ stK ()
{
    For (int=0; i<Tos; i++)
        stK[i] = -1;
    delete [] sK;
}
```

مجرد بقول لل (system)
الجزء ده مبقاش محجوز وممكن تستخدمه

هنا أ أ أ

stK	F100
size	5
Tos	ϕ_3

S1

F100 → | 3 | 5 | 7 | | |
0 1 2 3 4

note
دول كان ممنوع التعديل عليهم
في حالة لو كاتب عليهم (const)

```
Friend void View Content (const MyStack S)
};
```

نهاية الـ Class

مجرد بقول لل (class)
لو لقيت (standalone fn.) بره
اعتبرها صديقة وتقدر تشوف
الـ (private member)

مش هيتعملوا (inheritance)

interview (C++)

↳ Feature that violets (OOP) ?!

Friend Function → it's only in (C++) because it's SO wrong

1:05

Tracing

```
void viewContent (MyStack s)
{
    For (int i=0; i< s.Tos; i++)
        cout << s.stk[i] << ", " ;
    Cout << endl;
}
```

1:10

مشكلة ال (Bit wise copy)

مشكلتي أن عملت (pass by value) وبالتالي هيستخدم Temp Annoymus object لما أخلص ال (Fn.) هيعمل (Dest.) وهيمسح ال (Dest.) جوه ال (Dest.)

Call أن هيعمل ال (Destructor) أول ما تخلص ال (Fn.) غصباً عنك حتى لو لسه البرنامج كامل مخلصش

```
int main ()
{
    Mystack S1;
    S1.push(3);
    S1.push(5);
    S2.push(7);
    viewContent (S1);
    Cout << S1.pop();
```

(-1)

بيعمل copy لنفس قيمة ال (Address) لي كان في (S1)

فهتلاقي القيم بتاعتها بقت بـ (-1) بعد ال (viewContent)

* وبالتالي .. محتاج ← (pass by Reference)

عشان ميتعملش (object) جديد بسبب ال (Fn.)
في مفيش أي (call For Dest.)

بطيفيا في ...
بسطر ال (Friend) بردو !!

```
void viewContent (const MyStack &S)
{
```

ده أداله (location) جديد خالص
وعمل (Dynamic) جديد ...

```
S.Tos = 5;  X    (S.stK) = new int [7];  X
S.size = 20;  X   S.stK[0] = 33;  ✓
```

بس كده ممكن حد يعدل
على ال (Data) الأصلية ..
فجعلها (const.) عشان
أخلي ال (Data) تبقى

Read-only

ملاحظ أن وصلت لـ (Array)
اللي بيتحكم فيها ال (object)
ونفع يتعدل بسيطا ...

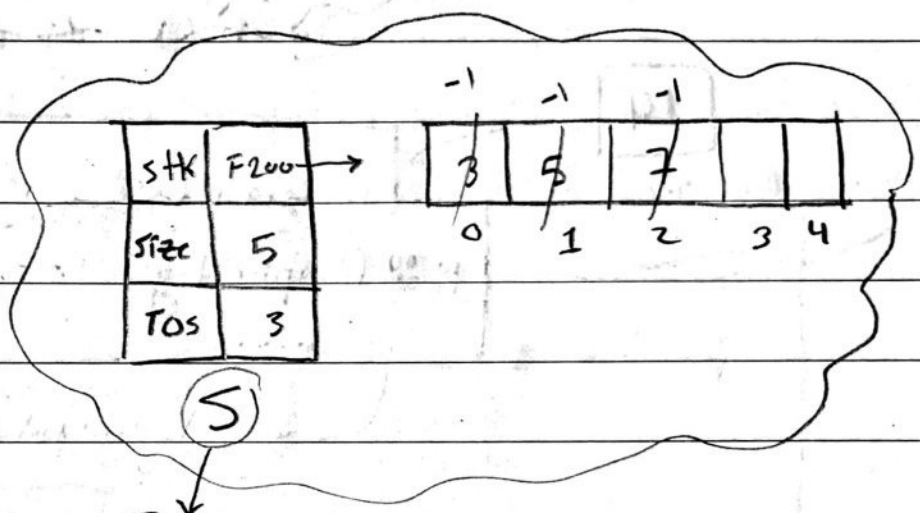
وكده حصل مشكلة بردو بال (cons.) !!

الحل جاهز !!

الحل هو ← عملية ال (Deep copy) !!

① محدش هيعدل على ال (Array) من جوه ال (Fn)

② حتى ال (Destructor) هيخلي النسخة ب (-1) .. مش الأصلية !!



⇐ (object) جديد عملته ـ
اشتغل بيه جوه ال (Fn)
وأخد منه النتائج قبل ما يتمسح
أو يتعدل عليه من ال (Destr)

بيعمل (call) لل (Destructor) بناء
على ال (object) ال (copied) ـ
فبيعدل على ال (Array) بتاعته بس !!

OOP 3-2

Copy constructor

14

حاجات كتير بيستخدم فيها
وتكون حالات (implicitly) !!

by value input

Mystack c (Mystack K)

ال (Compiler) بيكون عارف أن ده
ال (copy constructor) :)

passing object by value

8 MyStack (MyStack & olds)

this: s

{

Tos = olds.Tos;
Size = olds.Size;

بحط نفس القيم من ال (old) :)

stK = new int [Size];

For (int i=0 ; i < Tos ; i++)

{

stK = olds.stK [i];

} }

لازم أعمل (By Reference) :)

هنا جاية :)

عشان لو (pass by value) هيحاول
يعمل (Temp Annoying object)
ال بيستعمل أصلاً بـ (Copy construct)
ال كان (implicity) ـ لكن هنا عندي (Copy constructor)
فـ هيحاول يستخدمه ـ
وبالتالي هيدخل في (infinite Loop) :)

ميتفعش
ال (input)
يكون (By value)
وإلا هيدخل
في {infinite Loop}

جوه ال (class) :)) ← member fn.

17

بداية شرح ال (Task) :))

```
Mystack Reverse ()
{
    Mystack R;
    |
    |
    |
    Return R;
}
};
```

① memebe wise copy
② R → Temp Array
③ Destruct R
④ pop () -1

دي كده مشكلة :))

دي الحالة الثانية ال (Copy constructor)

عايز أعمل (object) في حالة ال (Return) عشان ال Temp Array object ميشتغلش بردو :))

stk	F200
size	5
Tos	3

R

~~6~~	~~4~~	~~2~~	

F200 0 4

stk	F100
size	5
Tos	3

2	4	6	

F100

kt

stk | 6 | 4 | 2 |

retur. ret

042

245

```
int main ()
{
    My stack S1;

    S1.push (2);
    S1.push (4);
    S1.push (6);

    temp = ~~X~~
    Cout << S1.Reverse ().pop ();
```

S1.reverse().pop()

S1(r).pop()

S1.print

stack

S2 =

temp

S2 = S1.Reverse()

S2.pop()

STK	F300
Size	5
TOS	3

6	4	

F300

View Content (S1.Reverse ());

with copy constructor

View Content(val, &)

Copy "Copy"

* الحالة الثالثة لإستخدام الـ (Copy constructor)

34

Assignments (Tracing) أغلبها

هي عملية الـ (Deep copy)
بعمل نسخة كاملة من (S1)

2	4	6	

F400
5
3

S2

Disable (RVO)

↳ settings → compiler → other compiler options

↳ -fno-elide-constructors

Task oop(3)

(1) class complex

with copy ctor

without copy ctor

Complex sum (complex & []);

(2) Complex MyStack

{

Friend view content (Mystack & &);

My stack Reverse ();

with copy ctor

without copy ctor

}

RVO → Return valve optimization

متخصصة في ال (Return) بس
مليش علاقة بال (input)
عشان كده كان بيظهر (5 Destructors)
للكلاس بتاع ال (input)

لما أعطلها (Disable) هيظهر ال (Destructor)
السادس -- بتاع ال (Return) عشان
اتعامل ال (obj.) واعمله (Assign)
بحاجة قبل ما يتمسح :)

لاحظ الفرق بين ال (copy life cycle)
وال (Normal constructor life cycle) :)

شرح هام جداً لـ (Copy)

في الكلام على

4 constructors → دائماً

عشان عندي دائماً الـ

Fn. input and Return

الـ (2) دول معمولين بـ (Copy) :)

4 destructor + 2 Destructors

input → بيظهر دائماً

Return → بيظهر في حالة عملت (Disable) لـ (RVO)

Return value optimization :)

هدف الـ (copy constructor)

* لاحظ أن الغرض الأساسي لـ (copy) أن يعمل (object) تاني داخل الـ (Fn) تتعامل بيه ..

ولما الـ (Fn) تخلص .. الـ (obj) الـ (copy) ده هو اللي بيتأثر بتغيرات الـ (Destructor) وميأثرش على الـ (Data) الأصلية ^^ !!

$\frac{N}{2}$ $\quad$ N

1	2	3	4	

0	1	2	3	4
1	2	3	4	5

Stack

4	3	2	1

~~Solve (arr(start~~

$\boxed{i = Top, i > 0}$ $\quad$ $i--$

R.push(start+6) ; stack(i)

(i = Top-1 ; i >= , i--)

1. push → إضافة رقم إلى ال (stack)
2. pop → بتشيل أول قيمة على سطح ال stack وبتقولك قيمتها ☺!!
3. peak → عاوز أعرف القيمة اللي على السطح من غير ما أعدل فيها حاجة ☺!!

1 Comple → No RVO

→ No copy → 4 —
6 —

→ Copy → 6
6

2 View

no copy → value → 1 Con
2 Des.

→ Ref → 1 con
1 Des.

copy → value → 2 Con
2 Des.

→ Ref. → 1 Con
1 Con

3

Reverse → Dest. 3 No Con 2

→ copy → without RVO → will Fail

→ copy without RVO

1 S1
2 R
3 copy at End of fn. → as Return

3

Dest. → 3

```
cout << S1.Reverse().Pop();
```

Return المفروض بيعمل
ال (object) :)

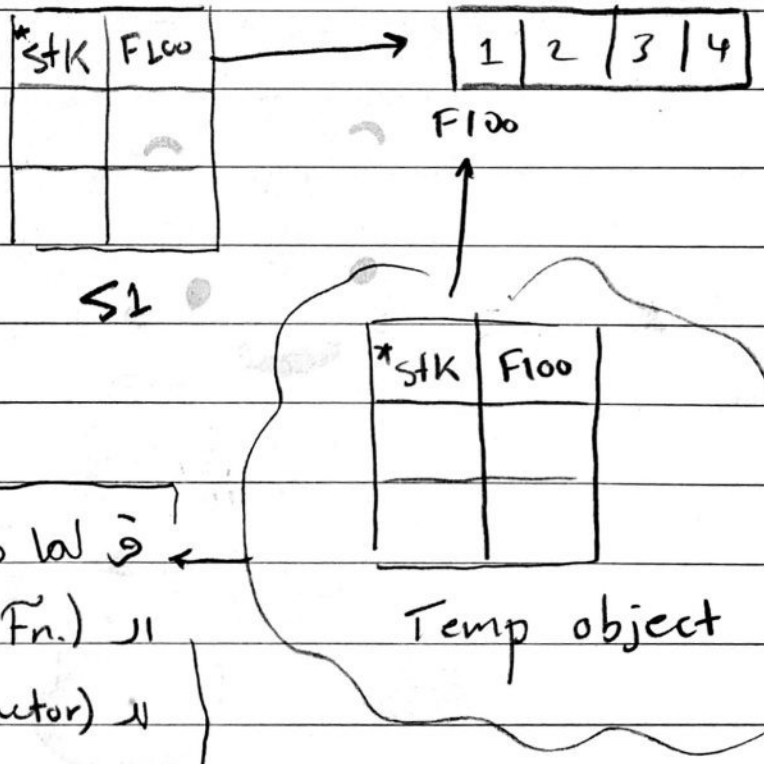
* كل مشكلتي مع ال (Temp object) أن بيعمل (copy)
لكل ال (member Attributes) بدون تفكير..
طب وايه المشكلة ؟! :) ..

في حالة أن كل حاجة على ال (stack) مكنش هيبقى فيه كلام
ولكن في حالة ال (heap) وأنى عامل (Dynamic Allocation)
ل (Array) ..

فهو لما يعمل (copy) لل (Address) بتاع ال (pointer)
→ (* stK)

* في نظره أن الـ (*stK) ده فاضي وهيبدأ يتعامل
بيه ... لكن هو بيدخل يلاقي أن الـ (pointer)
متسجل فيه مكان (Array) ..

* في الـ (copied object) بيتغاير وبيعدل في الأصلية
وهو مش عارف .. عشان دخل على نفس الـ (Address)



فلما ده يتنهي بنهاية
الـ (Fn.) هيعمل call
لـ (Destructor) .. وأنا قايل
جواه يروح لـ (ptr) يعمل Free
لـ (Data) ..
فيقوم مضيعلي الـ (Allocated space)
اللي كنت شغال بيها لـ (S1) :)

هاا جبت آخره :)

عدل ال (TOS) بتاعت ال (P) ..
عشان ميعكش (Stack is Empty)

31 / 10 / 2022

OOP-4

* عملت (const)؟ ← عشان امنع التعديل في ال (member Attribute) الأصلية بتاعت (S1) .. لكن مازال ممكن حد يـ (Access) ال (Allocated memory) ويعدل فيها !!

* عملت (S)؟ ← عشان مشكلة ال (Temp object). أنه بيأخذ نفس ال (pointer) في بيمسح ال (space) اللي بيشاور عليها (S1) مع أن (S1) لسه مكمل مهامها لآخر ال (Code) !!

مهم في حالة الـ (Dynamic Allocation) :)

طرق استخدام الـ (copy constructor)

1. input ⟶ لما يدخل (object) لـ (input) لـ (Fn) :)

2. Return ⟶ بعمل Return لـ (object) بس لازم استخدمه في نفس السطر بعده أو أعلاه (Assign) لحاجة تانية

3. Cloning ⟶ بعمل نسخة من (object) عندي

```
class complex
{
    int Real;
    int Img;

    public :
        // Ctor
        Complex (int R=0, int=0)
        { Real=R ; Img = I ; }

        // dest.

        ~ complex () { cout << "Dest" ; }

        Complex ( const. complex & old C )
```

عشان أضمن أن

هيحصلش لعب في ال (object) الأساسي !!

بسبب استخدامي ل (&) !!

```
        {   Real = old C.Real ;
            Img = old C. Img ;
        }
```

(Real الحقيقية) (Img) الخيالية

Complex Return ([ ] , [ ]) ← ممكن اختصر الشكل ده

جزء من تقليل الـ(obj) كده مش هيعمل (Copy ctor)

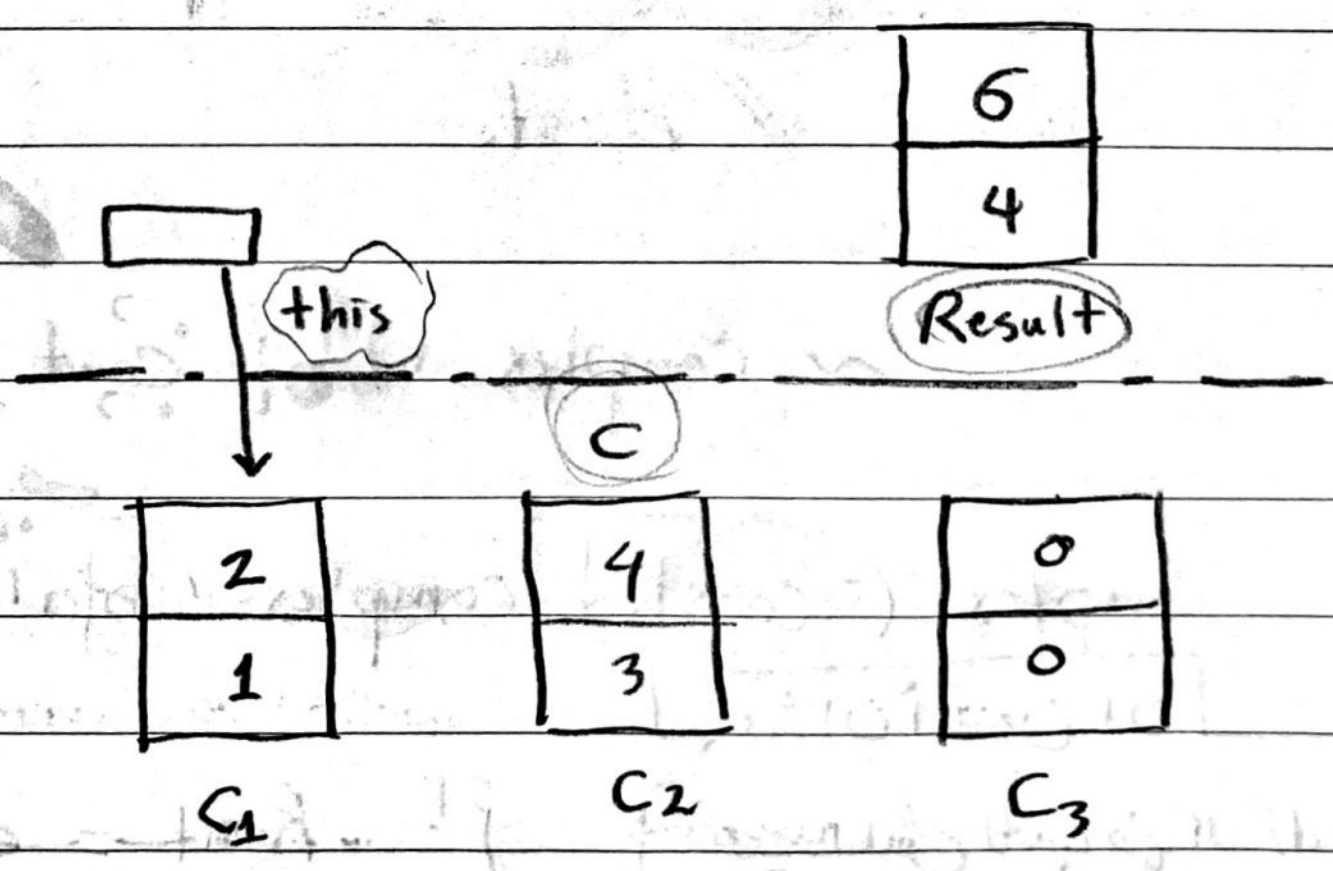
Const. & c

Complex Sum (complex c)

(operator +)

```
{
    Complex Result;

    Result.Real = Real + c.Real;

    Result.Img = Img + c.Img;

    return Result;
}
```

طريقة تقليل الـ(objects) أكتر



move constructor → (Return statement) في (ctor) أن بيتعمل

← متى هنستخدمها ← ليها جواز كده ؟؟

```
int main ()
{
    complex C1 (1,2),
            C2(3,4),
            C3;

// C3 = C1. Sum (C2);

C3 = C1 + C2 ;
```

this c

هستبدل الكلام ده بال (operator overloading) ؟؟

```
// C1.operator + (C2)
```

int + int

ليها شكل (stand alone) وشكل (member Function) ؟؟

→ complex + complex

C3 = C1 + 7 ;

لازم أعرفلها حالة
تأخذ فيها (int parameter)
مش (object) !!

1

member fn

```
Complex operator + (int R)
{
        Complex Result ( Real +R , Img ) ;
        return Result ;
}
};
```

ال (compiler) بيفرق بينهم عن طريق ال (signature)
بس لو نفس ال (signature) مش هيعرف يفرق ما بينهم !!

بفقد ميزة ال (this) !!

2 النسخة ال (stand alone)

```
Complex operator + (int L, const complex &R)
{
    Complex Result (L + R.GetReal(), R.GetReal())
}
```

الرقم الصحيح
بيتجمع على ال (Real) بس !!

استخدمت ال (getters) عشان
أوصل لـ (Attributes) ...
عشان مقدرش (this) تساعدني
أوصلهم !!

هـ ا ج آ ج آ !!

لاحظ ← في حالة ال (member)

(member Data) ← القيمة الي على الشمال هتحسب ك
بال (this) ... وبالتالي هطلب من
(input) واحد في ال (overloading) !!

في حالة ال (stand alone)

أعتبر غافلين (RVO) أغلب المحاضرة ؟!!

note

C3 = C1 + C2 ;

C1 += C2 ;

الى على شمال الإشارة اختلفوا ؟!!

Caller

(Return) هتدخل ال

By Reference يبقى &

Binary operator

Check

Complex operator += (const. Complex & C)

42

{

Real += C.Real ;

Img += C.Img ;

return * this ;

}

};

(this) لو حطيت هتدى

ال (Address) بتاع ال obj ← C1

لكن (*) بيقول

ال (obj-) نفسه ؟!!

ده (C1) ال (Scope) بتاعها

أكبر من ال (Fn) ؟!!

هتأخر حاجة ؟!!

هاها هاها هاها !!

+ نوفر objects --- نوفر (memory) ~

عشان كده معجبين بال (passing by Referenca) !!

$C3 = (C1 += C2)$;

6
4

$C1 \sim\sim\sim\sim += (C2)$

this ← C1 ، C ← C2

no RVO ⇝ complex

⇝ complex &

ده كان هيتعمل في حالة لو عملت (Return by valve) !!

انا يعتبر زودت قيمة (C2) على قيمة (C1) و بالتالي (C1) يعتبر بقى (C3) !!

هاها هاها هاها !!

C3 = C1

هاها هاها هاها !!

ميزة ال (Return By Reference)

ان هيتعملش (object) جديد وانا بعمل (Return) !!

فبوفر في ال (Memory) !!

Working with one object

هما أساساً جمعاً !!

Unary operator

أقدر أعمل (Return By Reference) في حالة ال (pre) بس ..
عشان في حالة ال (post) برجع ال (Temp) اللي يعتبر (local) ..
حتى معروف بو أصلاً !!

```
C3 = ++ C1;

C4 = C2 ++;
```

```
Complex operator ++ ()        // pre
{
    Real ++;
    return *this;
}
```

1

برجع ال (object) بعد ما عدلناه جزء ال (Real) !!

الكلام ده ينفع
ال (pre) بس ..
أما ال (post) !!

* محطتش (Return by Reference) في حالة ال (pre) أو ال (post)
عشان كانت هتعمل مشاكل في التفرقة :)

هاكاهم أيضاً :)

ده ال (Dummy) اللي بيفرقوا :)

2

```
Complex operator ++ (int)   // post
{
    Complex Temp (*this);
    Real ++ i;
    Return Temp;
}
};
```

note

افهم ال (Post) أكتر :)

بعمل (obj.) يكون (Temp)

4	Temp
3	

2	
1	

C1

4	
~~3~~ 4	

C2

* عملت (Temp) عشان يستخدم في السطر الحالي بدون أي زيادة
ولكن زودت الأصلي بحيث لو طلبته بعد كده في سطر تاني هلاقي ان قيمته زادت
وبالتالي بقى (post increment) :)

things you can't overload

→ ① * ② . ③ new ④ → ⑤ !

== → Comparison operator

بس لاحظ ال (Type casting) :)

```
int main ()
{
    complex  C1 (1,2)
             C2 (3,4)
             C3 , C4 ;

    if ( C1 == C2)
        cout << " EQ" ;
    else
        cout << " NEQ" ;

    int R = (int) C1 ;
```

1:05

not equal

ال compiler بيفهم
عشان مفيش غيرها :)

هيلاقي على يمينه (Complex) :)

~~(complex& R)~~ ← فل

```
bool operator == (const complex &R)
{
    return ( ( R.Real == Real )
             && ( R.Img == Img ) );
}
```

← (لازم الطرفين بـ True)

ممكن تحطها
ي; (int) S
ما بنعمل فيه (c)

↓

Backward
Compatibility

```
operator int ()
{ Return Real ; }
```

ممكن تحطش
أي (Return type)
!!

← هو بيقرأ عارف

← هعند بيش (input) عشان هو عارف
أن المفروض يدخل على (object)
اللي على يمينك
و كاتب (Real) على طول عشان
أنا جوه (obj.) وبأخذ (Attributes) بتاعته عادي

هام جداً
جداً !!

```
int x ;
long y ;
x = (int) y ;
```

→ same as int(y)

بدايـــة ← $\boxed{\text{cop (4-2)}}$

Ex 1 → راجع على كتابته وشرحه تاني

```
Class Time
{
    int H, M, S;

    public:

        Time ( int _h, int _m, int _s)
        { H = _h, M = _m, S = _s; }

        void print ()
        { cout << H << ":" << M
                << ":" << S; }

};
```

الهدف أن لما أدخله الأرقام - يتعمل (print)
لل (Time) في (Format) معينة !!

بيأخذ ال (object) اللي عمل يعينه
وبيقدر يا (Access) ال (Data) بتاعته !!

```
class Date time
{
    int Y, MM, D, H, M, S;

    public:

        Constructor
```

عملت (operator) باسم (Time) !!

```
void Operator Time ()
{
    Time T (H, M, S);
```

اسم ال (class) السابق !!

object name

constructor intilization

```
    return T;
}
};
```

برجع ال (object) اللي عملته

Ex 2

```
class MyStack
{
    int *stK;
    int Tos, size;

    public:

    MyStack ( int L=5)
    {  Tos = 0;  size = L;
       stK = new int [size];
    }
    // copy ctor
    MyStack ( const MyStack &S)
    {  Tos = s.Tos;  size = S.size;
       stK = new int [size];
       For ( int i=0;  i < Tos;  i++)
          {  stK[i] = S.stK[i]; }
```

عندنا ← C3 [=] C1 [+] C2

عند (2) operator overloading
ده هيعمل مشاكل ؟؟

R ← جاية (Read only)

فقرة الأسئلة 25

```
~ MyStack()
{   For (int i=0; i < TOS; i++)
        { stK [i] = -1; }
    delete [] stK;
}
Void print content ();

MyStack & operator = (const MyStack & R)
{   delete [] stK;
    TOS = R.TOS;
    Size = R.Size;
    stK = new int [size];
    For (int i=0; i < TOS; i++)
        { stK [i] = R.stK [i]; }

    return *this
}
};
```

دول مش مهمين بالنسبة لينا النهاردة ؟؟

ال هيتعمله (Free) هو المستقبل على الشمال عشان أضمن أن عنده Size كافي لإستقبال القيم ؟؟!!

R →

عشان فعلياً عندى (2 objects) خلاص -
مفيش حاجة لـ (object) جديد ..

operator overloading

وبالتالى لازم أعمل

السطر ده مش هيعمل (call)
لـ (copy ctor) عشان
يحل المشكلة ؟؟!!

```
int main ()
{
    My stack S1, S2 (3);

    S1.push (3);
    S1.push (5);
    S1.push (7);

    S2 = S1;

    Cout << S2.pop() ;  → 7

    S2.push (13);

    Cout << S1.pop ();
}
```

S3 = (S2 = S1)

بتعمل مشكلتين ← ① بعمل (memory Leak)
بسبب ضياع مكان (space) الـ (S2)
② (S2) هتبقى تقدر تعدل على قيم (S1)

* عشان نعمل (copy) مع بال (operator)..

لازم ← ① نعمل (Free) لـ memory الحالية
وبعمل (memory) تانية..

← ليه؟ ما استخدم الـ مكانها

← عشان معرفش الجديد هيكون
نفس الـ (size) الـ مكانها ولا لأ ☺!!

← في الأسول أعمل (new Allocation)
بال (size) الجديد وخلاص..
مش لسه هتارن ☺!!

② يساوي بقية القيم ☺!!

3	5	~~7~~ 13	

F300

✓ ①

3	5	7		

F100

F200 0 1 2

أول خطوة
عل (Free)
لـ (Allocation)
القديم 80 !!

stk	F100
size	5
TOS	~~0~~ 3

S1

stk	~~F200~~ F300
size	~~3~~ 5
TOS	~~0~~ 3

S2

* تدخل (By value) .. يبقى لازم ترجع (By value)

بتتكلم على نفس ال (variable) اللي ← أعتقد

Ex 3

Case 1

passing by value

```
int   Fun01 ( int x)
{
      ++ X ;
      return x ;
}

int   main ()
{
      int  A = 7 ;
      int  R = Fun01 (A);
                  8

      Cout << A ;

      Fun01 (A) = 13;
      8
}
```

X	7 → 8
R	8
A	7

Stack

* مجرد زيادة (*) عدلت على القيمة الموجودة
ال (Address) اللي راجع !!

Case (2)

we passed the Address

```
int * Fun02 (int * x)
{
    * x ++ ;
    return (x) ;
}
```

(*x) ← ال pointer بيستقبله !!

holds Address

X	int* FF00
R	8
A	~~7~~ 13 ~~9~~ 8

FF00 stack

```
int main ()
{
    int A = 7 ;
    int R = * Fun02 ( &A ) ;
    Cout << A ;
    * Fun02 ( &A ) = 13 ;
    Cout << A ;
```

&A → FF00

we Dereferenced (X) and then stored a new value For (A) !!

→ (13)

store this value in the Dereferenced location

Also Returning by Reference

Case (3) → Passing by Reference

```
int & Fun03 ( int &x)
{
    X++;
    Return X;
}

int main ()
{
    int A = 7;

    int R = Fun03 (A);

    Cout << A;  ⟶ (8)

    Fun03 (A) = 13;
        ↓
        A

    Cout << A;  ⟶ (13)
```

	R	8
x	A	7 8 8 13

FF00

هنا بقى بقى بقى :)

عشان كده قدرنا نحط (13) ونعمل (Assign) ل (Fn) عشان هي زي ما أخدت ال (A) هترجعها تاني بعد ما عدلت عليها :)

* مشكلتي مع ال (Array) ان ممكن أ (Access)
أي قيم خارج حدودها بمجرد أزود ال (index) ؟؟

Task → I think → [48]

```
Class int Array
{
    int* Arr;
    int size;

public:
    int Array (int s)
    {
        size = s;
        Arr = new int [size];
    }

~ int Array ();

int Array (const int Array &);

int Array operator[=] (int Array &);

int GetSize ();
```

علينا احنا
نكتب ال (ctor)
وال (Dest) ؟؟

لازم copy ctor ليه

طالما عندي (copy ctor)
بيعتبر لازم overloading
هعرفتها ليه -- ؟؟ مثال كده

بتأكد أن أنا في ال (Range) بتاع ال (Array) !!

```
void SetValue (int index, int Value)
{
    if ((index >= 0) && (index < size))
        {Arr[index] = Value; }
}

int GetValue (int index)
{
    if ((index >= 0) && (index < size))
        { Return Arr [index]; }
}
```

Member Fn. : Direct call

operator overloading → For Array

هيرجع ال Arr [index] تبقى
وبالتالى ال(overloading) هيشتغل
على يمين أو شمال علامة ال (=) !!

دى الإضافة الوحيدة !!

```
int& operator [] (int index)
{
    if ((index >=0) && (index < size))
        Return Arr [index];
}
```

هام جداً جداً !!

كده هينفع أ (Set)
أو أ (Get) أى قيمة بيها !!

عشان يرجع قيمة متغير (object) !!

دى هتخدمهم عليا
استخدام Get value() !!

size	7
Arr	F100

MyA

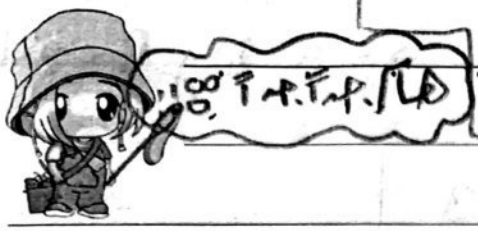
3	—		

0 6

هام جداً جداً !!

* هدفى أن أحصرك فى (Range) معين .. لكن عادى تدخل فى (index) أنت لسه مستخدمتش حاجة فيه .. ويطلعلك (garbage) !!

* الهدف استخدام [ ] مع اسم ال (object) على طول ---
بدل ما كنت بدخل لل (Array member) الداخلية --



~~S1.STK[i]~~ → S1 [i].

```
int Array MyA (7);

MyA.SetValue (0,3);
              (1,5)
              (2,7);

For (int i=0; i< MyA.Getsize(); i++)
{
  Cout << MyA.GetValue (i);
  Cout << MyA [i];
```

MyA [i] → MyA.Arr [i]

```
MyA [0] = 3;
MyA [1] = 5;
MyA [2] = 7;
```

شرح ← (char*) ← خر الدقيقة 8

S3 = (S2 = S1) ← (S2) مش هتتغير ← 2.5

S3 = My Result

Old Task - D3

1 Complex

No copy Ctor → RVO = off

4 ctor → C1, C2, C3, Result

6 Dest. → (Ctor) اي مش ظاهر ليهم
لها ال (C) ← (input)
Result ← (Return)

Copy

6 Ctor } → (copy ctor) عشان وفرت
6 Dest. } → (Ctor) ال هيبان بالتالي

سؤال

إزاى أعمل (print) لاسم ال (object)
يعنى أقول أن الرقم ال (complex)
ده من ال (print) بتاعت (C3)

1 / 11 / 2022

oop-5 → هنتكلم على ال (static)

stK	FF00
Tos	0
Size	7
SNum	~~0~~ 1

S1

stK	FA00
Tos	0
Size	5
SNum	~~0~~ 1

S2

بيفضل ثابت مع كل (object) !!

هنلغيه من هنا .. وهنعرفه
مرة واحدة في ال (Heap) عشان هيفضل
معانا طول ال code

~~1~~ ~~0~~ 2

SNum

this → SNum

بيتشاور على ال (object) الحالي .. ولكن (SNUM) تبع ال (class) كله

عشان كده الكلام ده غلط !!

[1] حطيته مع ال (public) ولكن ده غلط .. كده أي حد ممكن يعدل في قيمته ..

[2] int MyStack :: SNum = 0;

بيتحط ويكأنه (global) .. وبيشتغل حتى لو ال (SNUM) محطوط (private) !!

اللي عملوا ال (C) هم اللي أجبرونا على السطر ده !!

① معندهاش الـ (this)

(Does'nt have caller object)

static

الى بيفرقك عن (static) الباقى ؟؟

static int GetSNum()
{ };

(Object Attributes) الـ هيقدروش يوصلوا

ولكن لو بعت (input)

بـ (pass by Reference)

هيقدروا عادى ؟؟

~~static class~~ → مش موجودة في ال (C++)

Template class → (✓✓)

Fn. Template

* بكتب ال (Fn) مرة واحدة ب (Type (T))
وال (compiler) يختار ال (Type) الصح بناء
على ال (input) ☺!!

* لازم أعيد كتابة template <class MyType>
بعد ال (class) لو مستخدمها تاني --

↳ عشان دورها بينتهي بعد ال (class) أو بعد ال (Fn) ☺!!

↳ شوف ال (youtube) ☺!!

↳ هو ال (Block) اللي بعديه (class) or (Fn)
أو حتى (Line of code) عادي ☺!!

↳ ممكن أدي ال (class) أكتر من Type

ممكن (Fn) بتأخذ (Type) وبترجع تاني (Type) ☺!!

Template < class T Result , class T in >

UML → Unified MarKup Language

2:20

الترتيب من الأضعف للأقوى ..

درجة الإعتمادية بين طرفين العلاقة هي !!

Relationships

2:30

(1) Association

(has a)

شرح

(1) Relation Between objects

الإنسان لديه شقة !!

(2) Temporary relation

(3) peer to peer (no whole and part), can't be listed as part of Relationship

(4) Uni OR Bi-Directional

(5) No dependancy for creation, Removal of objects

(6) Represented by pointers (S), References (S)

Ex

```
class Teacher
{
    string Name;
    int ID;
    student * SArr;
}


class student
{
    string Name;
    int year;
    Teacher *TArr;
}
```

2:41

الكتاب والمكتبة ـ لو المكتبة
قفلت (الصحت) ممكن الكتاب
يتقرأ من البيت ☺!!

2 Aggregation → هنا شبه (implementation) الـ
الـ (Asscosiation) من
الـ (C++) ☺!!

(1) Relation Between objects

(2) Temporary relation

(3) whole / part Relation (OR)

(4) uni-Directional

(5) No Dependancy on Creation or Destruction

(6) pointer or Reference

(7) Stranger Form of Asscosiation

Ex → هنأخذ أمثلة ال (Aggregation) بشكل متعمق أكثر بعدين ☺!!

class whole	class Part
{	{
part *Parr;	
}	}

3 Composition

جزء لا يتجزء من الشيء
زي جناح الطيارة - Ex
لو اختفى (اتمسح)
الطيارة هتقع ☹

(1) strongest Form of Aggregation

(2) permanent Relation

(3) complete ownership (whole / part)

(4) Unidirectional

(5) complete Dependancy on Creation, Dest.

(6) represented by object as an Attribute inside another object

```
Class Floor
{
    int Number;
    int Room Count;
};
```

2:49

Task oop-5

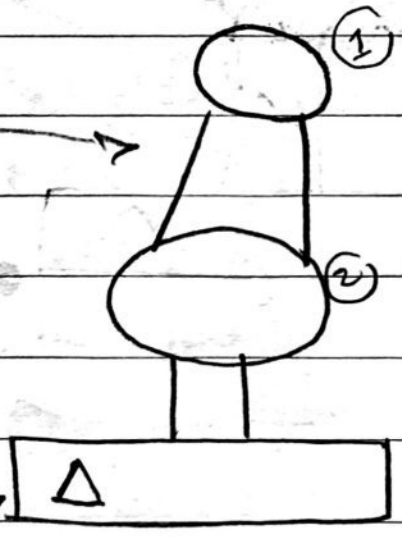
هتعمل أباجورة ؟؟!!

Class intArray

1. Temp → Array, stack, Queue

2. Graphics

Rect()
Line()
Circle()



* المهم نستخدم الأشكال دى فى الرسم

initgraph();

↓

(x, y, radius)

هنحطها في أول ال (code)
عشان نقوله أن الرسم
الجزء ده هيكون رسم

ارسم على ال (paint)
وخد الإحداثيات
وطبقها في ال (fn) ؛؛

(المفروض هنشتغل كده
Constructor chaining ؛؛)

22 → Triangle

* ممكن تعمل كل (class)
في ملف لو عاوز

Cont. Task (4)

1 Complex operator overloading

هستخدم جملة جديدة !!

(6) 7 -= C1 → عندى نسخة ثانية فنزا
فلازم أعمل دى (stand alone) !!

7 = 7 - C1

هنعتبر أن ده (Variable)
وهدخل قيمته ب (7) !!

وهعمل (Return) لـ (Variable) !!

Ex 7 = 7 - (-5) = 12

To_string | int → str) (char array)

Push_Back

* كل مرة احتاج أعمل (stand alone) عشان ..

يكون استخدمت ال (operator) ده لغرض ما قبل كده ..

فعشان أعمل نسخة تانية هحتاج signature مختلف !!

1 Memeber Fn. → C1 -= C2

C1 -= 7

complex & operator -= (int R)

{ }

2x complex مرة
ومرة complex + int
فمكنش فيه تعارض
مابينهم !!

2 stand alone Fn.

operator -= (int a, const complex &R)

{ }

7 -= C1

$C3 = C1 + C2 \longrightarrow$ C3 = C1.operator +(C2)

يعني بعمل (call) لل (member Fn) اللي جوه (C1) { شمال العادة

وبدخل (C2) لـ (input) ليها !! ∴ فالترتيب بيفرق !!

* طب ال (Compiler) فرق بينهم إزاي ؟!

← بناء على اللي محتاجه واللي اتوفر له ..

① في حالة ال (member) ..

C1 -= 7

أنا دخلته (int) بس وهو أخد ال (complex) المقابل على طول واستخدمه جوه ال (Fn)

5-9i → 7 -(-5)

12 ∴ 12

7 -= C1

② في حالة ال (stand alone)

هو متعرف أن عاوز (int , complex)

← ده ال (Error) اللي بيظهر من ال (Compiler) !!

note →

* ال (stand alone) هتشتغل مع الحالتين ..

ولكن هي مش مجهزة لـ (Img) فبالتالي

مش هتدي (C1) قيمها ال (Img) !!

Returns a string that textually represent an object --- ☺!!

To_string

(14) (char*) C1

عاوزين نحول مكوناتنا ال (c)
s (complex number) إلى (string)
ونرجعه s (string)

string = " Real +/- Img (i) "

1. Array of char
2. strcpy

خطوات تحويل (complex number) لـ (Char Array) ؟!

① لو عملت (Type casting) هيديك ال (char)
المقابل للرقم في ال (ASCII) ..
بس انت عاوز الرقم نفسه
في هنستخدم ال (to_string)

string s = to_string (Real) + "+" to_string (Img)
+ "i"

دول بيربطوا ال (strings)
مع بعض !!

② Dynamically Allocate an Array
to output values outside the Fn without
it being Removed.

```
char * arr = new char [ s.length + 1]
```

③ لازم نعمل (copy) لـ (string) في الـ (Heap array) دي
عشان أقدر أعمل (pass) لبره الـ (Fn).

لسه الـ strcpy() بتأخد (pointer to array of char) بس :)

a) S.c_str() ← وبالتالي سيستخدم

بترجع لي (pointer to char Array) :) جاهزة بتعبر عن الـ (string)
وفي آخرها '\0' عشان تشتغل (Automatic)
مع الـ strcpy()

```
strcpy( Arr, S.c_str())
Return Arr;
```

b)

الإختيار التاني تعمل (For Loop) وبعدها تحط
الـ '\0' في الآخر (Manual) :)

[2] operator Array

NEVER Return By Reference ----

iF the object is only inside the Fn.!!

عشان هو موجود جوه ال (Fn) بس و بيتمسح عند الخروج منها ..
انت بتعمل (copy) لل (values) لبره بال (copy Ctor) او خلاص ..

عشان لو حاولت تلمسه هاتجيب جرة !!
البرنامج هيرمي (Error) عشان هو اتمسح !!

0 X 61dF0

C1 = -7 - 2i
C2 = 3 + 4i
C1 is greater

* الـ (copy ctor) هو اللي بيـ (Extend the Life)

لـ (Data) عشان استفيد بيها بره الـ (Fn)

لكن الـ (obj) اللي كنت شغال بيه جوه ...
بيـ (Call the Destructor) أول ما الـ (Fn) تخلص على طول

على عكس الـ (copy ctor) اللي هيتمسح بالإنتقال لسطر آخر من الـ (code) الأصلي ☺!!

ده انتظار عملية الـ (=) بعد الـ (+) علشان عشان كانوا على نفس السطر ---

الـ (copy ctor) حد محترم جداً ☺!!

هيظهر الكلام ده معاك في الـ (Tracing)
لما تشوف الـ Address ← (this) ☺!!

* لاحظ لما تعمل (Compare) --
لازم تحط الـ (Result) جوه (Return) وإلا كأنك
معملتش حاجة !!

هام جداً جداً !!

[3] stack

① لاحظ دائماً قيمة الـ (TOS) --
عشان بناء عليها بيتم إضافة العناصر أو الطباعة !!

② عشان تحقيق أو تشيل لازم تستخدم --
Ex push(5) , pop()

عشان الـ (Arrays) ← (private) !!

هام جداً جداً !!

دى ميزة operator [] ← يعتبر (Fn.)
ولكن (implicit) وبيساعدنا فى الإضافة
وتعديل القيم زى ما كنت بعمل مع أى (Array)
عادية خالص !!

بكرة الـ (Dynamic Binding)

2/11/2022

أول وأهم حاجة فيها هي أن الـ (inheritance) يكون (public) !!

OOP - 6

inheritance ⟶ "is a"

غير علاقات اصبارح اللي كانت جزء من حاجة !!

Class Child : [ ] Parent

(public) by Default

هش بيحصل (copy) .. الـ (Fn) موجودة مرة واحدة .. ولكن بتشاور عليهم بالـ (pointers) !!

* بنستخدم الـ (Constructor Chaining) زي (مسألة الإجرة)

لما تستدعي (Fn.) ← افتكر تحطها الأقواس بتاعتها ☺!!

~~Get X~~ → Get X () ✓✓

هام جداً جداً ☺!!

parameterless ctor

* لو معملتش (Chaining) .. هيروح لـ

بعدها ممكن أعرف (X , Y) من جوه الـ (ctor) بتاع الـ (Child) ☺!!

Delegat → behave as I can do a Fn. while in Reality I am only sending orders to something Else to do it For me ☺!!

C#

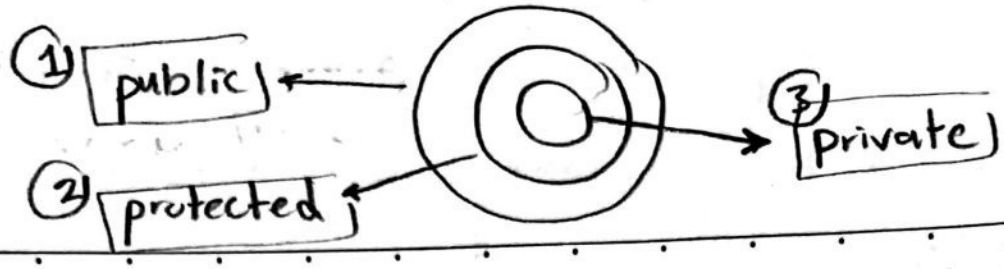
* عاوز أخلي الـ (Attribute) متاحة لـ (child)
(Direct Access) ؟ :)

* الحاجة اللي كانت (private) في الـ (parent)
لو عملت إيه في الـ (child) هتبقى
هتعرف تخليها (Accessable) :)

```
C.X = 5;  X
C.Y = 7;
```

عشان أحل الكلام ده بستخدم (protected)
بين بره الـ (classes) خالص ..
في class جديد أو داخل الـ (main)
الـ (protected) هتبقى زي الـ (private) :)

هام جداً جداً :)



Fn. overriding

بيستفاد ال (Developer)
أن ممكن كل مستوية افتكر
اسم ال (Fn.) بتاعت
ال child وبتاعت ال (parent) ؛؛

```
int  Sum ()
{
    return parent :: Sum () + Z ;
```

scape operator

لو كتبت Sum() بس كان
هيحصل (Recursion)
و (overFlow) ؛؛

لازم أعرفه أنا عاوز
أنهي Sum()

C. parent :: sum()

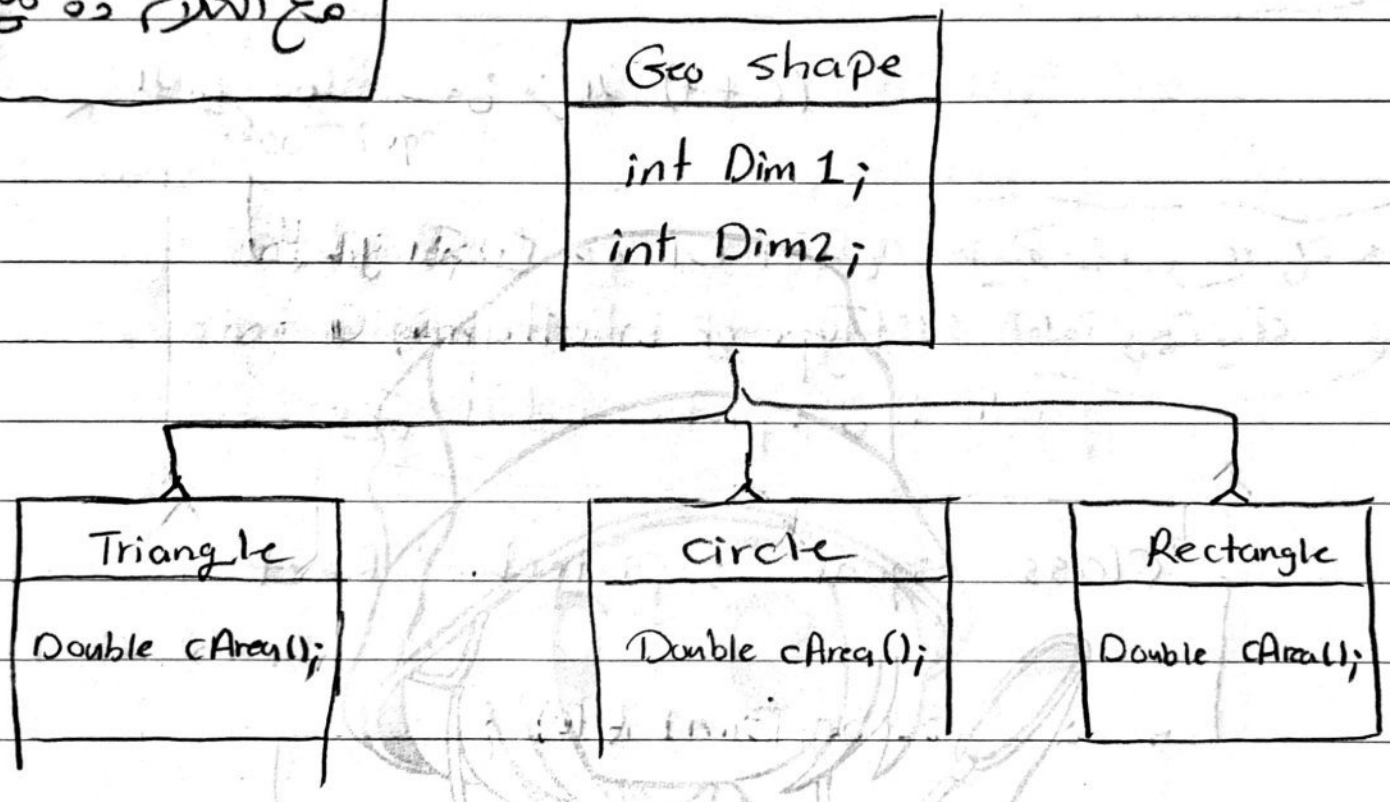
Fn. عاوز بعضها وتشتغل
في ال (main) بس ممكن مغفلة ؛؛

غير مسموح بيه في
ال C# ؛؛

في كل مرة بتعمل (ctor) من (child) --
بيعمل (ctor) لل (parent) الأول ...
هو ده هيستهلك (Memory) ولكن ..

ميزة ال (inheritance)
في إضافة أكواد جديدة
بدون الحاجة للتعديل على
الأكواد القديمة وخلينا
هنتناش مشاكل
مع الكلام ده ؟!!

* هتتكلم عن أمثلة ال (shapes)



oop 6-2

* : C. GeoShape :: Set Dim1 (70) ;

الحوار ده سببه في ال (C++)

حل الكلام ده أن أغلى
ال (Type of inheritance)
بيبقى (protected) :)

Class square : protected Rect

S. SetDim1 (40) ;

كده منعت ال (Developer) يوصل لبتاعت ال (Rect.)
وهتحتاج أعمل (SetDim1()) خاصة بال (square) :)

so → S. Rect :: Set Dim 1 (40) ;

invalid .. because it's in-Accessable

Rect :: setDim() → موجودة في اللغات التانية

S - Rect :: Set Dim() → موجودة في ال (C++) بس !!

هنستوبه في ال (C#)

لغت ال (Type of inheritance)

خليت كل حاجة (public) !!

* وبالتالي لازم أعمل (Area() جديدة جوه (square) !!

* كل اللي بتحكيه النوارده (Static Binding) هينتهي تماماً أول ما نعمل (Dynamic Allocation)

```
class Base
{
    int x;
  protected:
    int y;

  public:
    int z;

    Base (int x, int y, int z)
        { --- }

    int Fun ()
    { return x + y + z; }
```

Regardless of (inheritance Type)

تنسه مهم

مهمتهش أوهل حاجة كانت (private) جوه ال (Base)

هسكا جهاً جهاً مهم

```
Class Derived : [ ] Base
{
    int A;

  protected:
    int B;
  public:
    int C;
    Derived (int a, int b, int c
            , int a, int b, int c);
      { —— }

  int Fun ()
  {
    return A +
           B +
           C +
           X +
           Y +
           Z;
  }
```

Class Derived02 : [ ] Derived

{

int K;

protected:

int L;

public:

int m;

Derived2 (---) { }

int Fun ()

{

Return K +

L +

m +

A +

B +

C +

X +

y + → [ ]

(60) Z + ~~y~~; → [ ]

```
int main ()
{
    Base o1;          | Derived o2
                      |
    o1.X =            |
    o1.Y =            | o2.Z = [      ]
    o1.Z =            |
      1;              |
```

11:00 [type of inheritance]

* لو ال (inheritance Type) ال عندى (protected)

في أى حاجة كانت (public) في ال (base)
هتنزلي (protected) في ال (child) ☺!!

فكر فيها كأنها معناة ☺!!

1. public , public
2. ⟶
3. protected , protected
4. public , private
5. protected , public

هينفعش توسع الى جايلك
(private) أو (protected)
هيفضلوا زى ما هم ☺!!

الإمتحان صح وغلط وبعدين (MCQ)
(ساعة) أو (ساعة وربع)

الإثنين ← 11/(7) ← C

الإثنين ← 11/(14) ← oop

9 choices → عدد الإحتمالات اللي عاوزين نعملها

اتعمل (5) منهم في المحاضرة
هنعمل الـ (4) اللي قيين في الـ (Lab) :)

Tasks

① Type of inheritance → Excel sheets or comment or powe

② Geo shap → Sum Areas

هنكتب الكود الى البلاستيشن كتبه

stand alone fn.

Rect, Triangle, circle
↳ square

(Return) sum of All the Areas

pointer From shape + No. of shapes

عندي (For Loop)
هتمشي على كل واحدة :)

int SumArea (8 parameters)

Return Type

4 shapes

4 sizes

يقصد عدد ال (shapes)

[3] new class (color)

every shape inherites From it

تكملة لـ مسألة الرباعجورة :)

~~Square~~ ~~Triangle~~ → دول مش موجودين في الـ (Fn.) !!

3 Lines

· Rect

Tasks oop-5

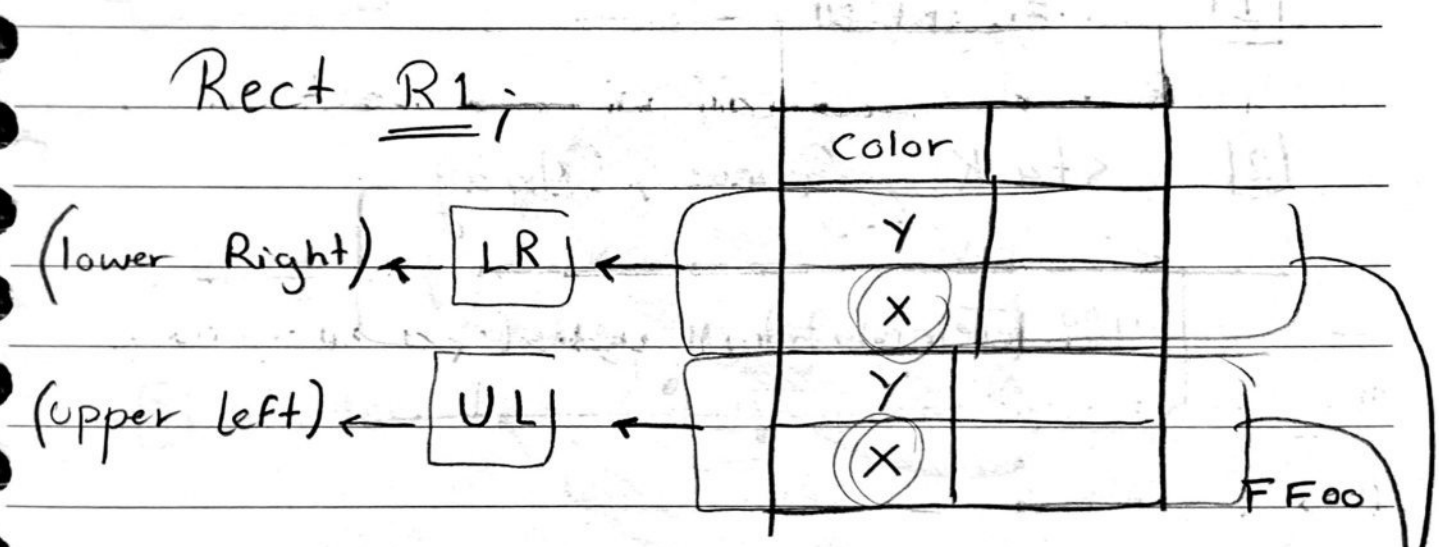
1. الأبامبورة

2. stack , Queue , Array

أعملهم بالـ (Template) !!

1 Drawing

Rect R1;



حصلوم عملية احتواء كامل (2x objects)
داخل (object) أكبر الى هو (R1)

دى كده علاقة (composition) ---
عدم وجود ال (2x objects) يعني
انتهاء ال (R1)

* لاحظ لازم نكتب ال (class) بتاع ال (point) الأول .. وبعدين ال (Rect) !!

* ترتيب تشغيل ال (ctor)

Rect R1 → point LR → point UL

ده بيحصل بحاجة اسمها (Constructor chaining) !!

Ex

Rect (int X1 , int Y1), int X2 , int Y2), int C)

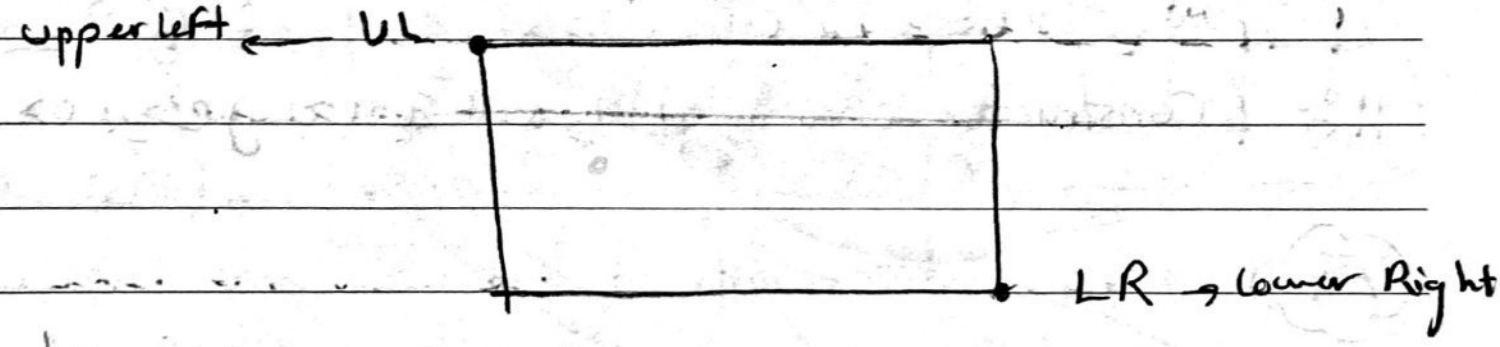
: UL (X1 , Y1) , LR (X2 , Y2)

ممكن يتحطوا في نفس السطر بس عشان المساحة !!

* كده ال (compiler) هيستخدم ال (parameterized)
بدل ال (parameterless) لـ point !!

* هنعمل (class) لكل (shape)

وهنبدأ نحدد كل واحد منهم محتاج كام نقطة ؟!



1. بعد ما بتسجل (2 points) عندك في ال (Class)
بتظبط ال (Constructor chaining)
بحيث تدخل كل القيم مرة واحدة لما تعمل (object)
من Rect()

2. بتعمل في نفس ال (Class) ← (Draw Fn.)
كل اللي بتعمله أنها بتأخذ القيم اللي انت سجلتها
لكل (point) وتدخلها لـ (input).

أبعاد الرسمة

عشان تظبطها افتح ال console وافتح صفحة (paint)
على مقاسه وهتلاقي الإحداثيات تحت على الشمال ☺ !!

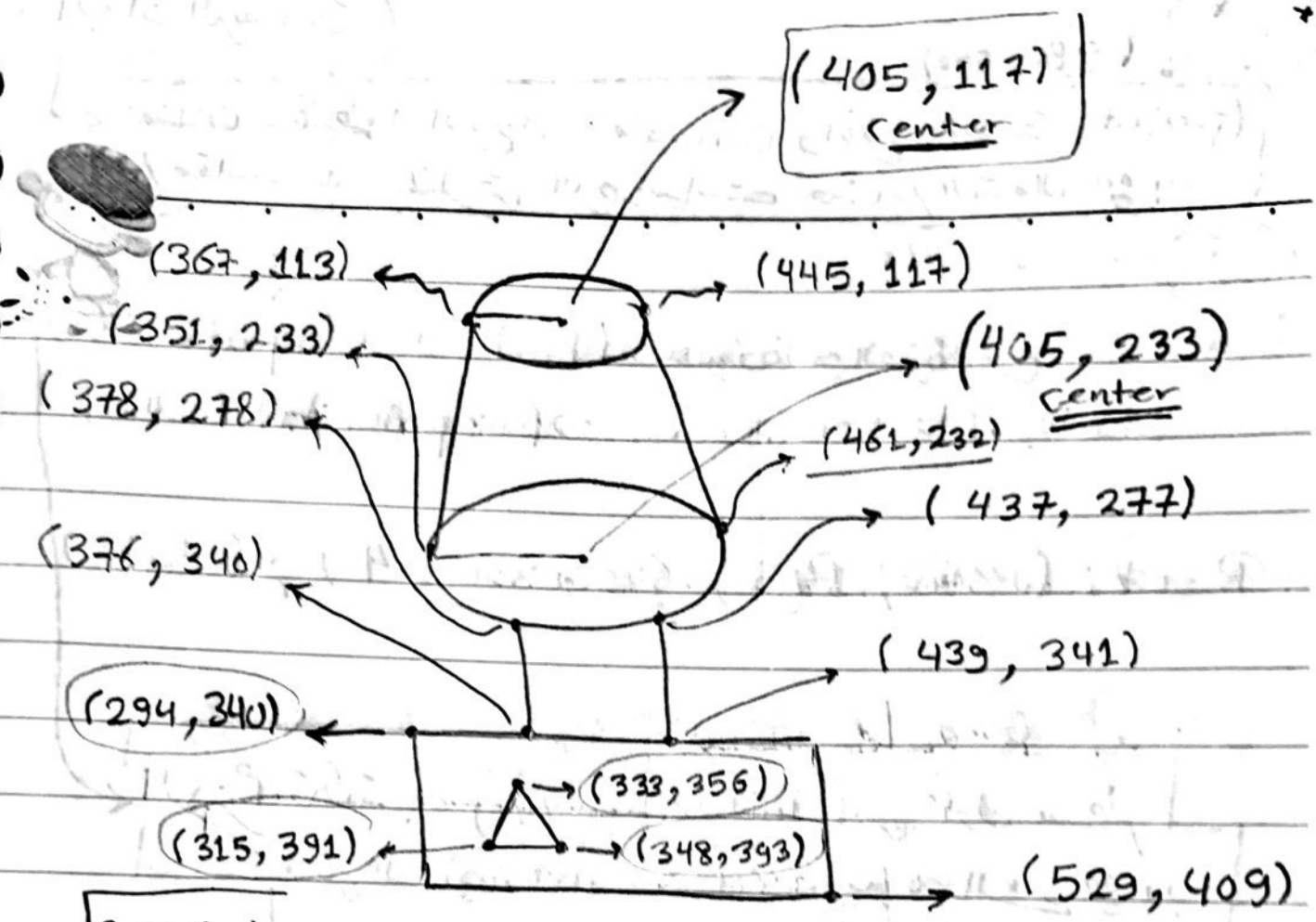
③ (input) لـ (Fn.) موجودة جاهزة من
ال (graphics.h)

Rect (300, 145, 620, 300, 4);

لازم تكتب (initgraph();) في أول سطر
في ال (main) عشان يرسم معاك ☺ !!

المسئول عن اللون هو (setcolor(4);) ودي
بحطها كجزء من ال (draw()

دي هتحدد لون الخط اللي بترسم بيه مش ال fill ☺ !!



2 points

(line , circle , Rect) الي عندنا

اعمل بيه أي شكل
زي مثلث مثلاً !!

* هعمل ال (Triangle)
ب (line) ـــــ
يبقى هحتاج 6 points

* الدائرة بتأخذ ال (x , y , Radius)

center point

Pro Tip

∴ ① Circle → one point + int Radius

② Rect → 2 points

③ Triangle → 3 points

و تستخدمهم ال (Line Fn.)
من ال Draw()

أنا كنت هعمل بدل point ← (line)
و هعرف (3x lines) ..
تكرار كتير بلا هدف !!

2 stack

1) مرة هتدخل (int)

2) مرة هتدخل (char)

3) وتتابع عدد ال Constructors بال (static)

int MyStack <MyType> ;; view ()

بحطوها قبل ال object
ويكون محدد <char> مثلاً

في الكلام ده يسمع من ال (ctor) جوه
وأي حاجة إنا كنت سايبها <MyType>

```
static int GetSNum()
{ Return SNUM; }
```

بستخدمها عشان أوصل لـ (static Attribute)
عشان بيكون (private) :)

الـ (static counter) بيعد الـ (int) لوحدهم
والـ (char) لوحدهم :)

Tasks oop-6

1 color class

setters and gettors

and use ctor chaining then use Get Color() to use the value inside a (child class) :)!!

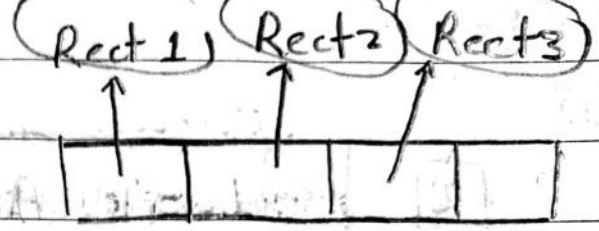
2 Sum All Area shapes

*جرب الأول أن كل حاجة شغالة طبيعي ~

الفكرة ← 1 عاوزين نعمل (arrays of objects)
هندخل لـ (Fn) ① اسم الـ (Array) ..
الي هو يعتبر (pointer) ؟؟ !!
② بالإضافة لـ (size) ..
الي هو يعتبر عدد الـ (objects)
الي هنعمله (sum) ؟؟ !!

2 والـ (Return) هيكون (Double)
عشان الـ (percision) ؟؟ !!

in C++ → array names are passed by Reference Automatically by the compiler !!



نفس الكلام مع بقية ال (shapes) !!

Array of class objects

```
Rect RectArr[1] = { Rect(20,30) }
```

- Rect: اسم ال (Class)
- [1]: لازم أحط الرقم هنا مينفعش (variable) عشان دى على ال (stack)
- Rect(20,30): كل (object) متعرف بال (Array index) فمش لازم تروح تديه Name معين زى Ex: Rect R1(—,—);

هام جداً جداً !!

كل اللى بعمله أن بنادى على ال (constructor) باسم ال (Class) الأصلى بحيث يعمل (intilization) لل (object) ده !!

* ممكن أعرف (objects) بره الـ (Array)
وأدخل لـ (Array) اسم الـ (object) !!

Ex)

```
student s1( ___, ___ )

student arr[2] = { s1 } ;
```

كده العنصر التاني هيستخدم ضمنياً
الـ (parameterless constructor) ..
فلو انت مش حاطل واحد هيطلع (Error) !!

Class ال بيره

لسا جاية جاية :)

بمجرد ال (variable) ده هيكون (Accessible)
في الوضع الحالي ولا لأ :)

3 Type Tracing For inheritance

* بنعمل (Elimination) الأول للحاجات اللي مش
هتتأثر بال (Type) اللي هـ (inherite) بيه

Ex → لو حاجة (private) في ال (Base)
عمرها ما هتكون (Accessible) حتى
لو عملت إيه في ال (child) :)

* أي حاجة جايية من ال (base)
وكانت (public or protected)
بيكون لينا كلام معاها :)

زي ال (Z, Q, Y)

Z → public

Y → protected

1

* في حالة الـ main()

(1) Derived 02

02.z

فرصته الوحيدة في (Accessible)
أن يكون (1st inherit) ← (public)
غير كده هيكون (in Acessible)

(2) Derived 03

03.z

لازم (1st, 2nd) يكونوا (public) ← احتمال ضعيف
محصلش غير مرة من (19 احتمالات) !!

03.c

كانت (public) في (02)
وفي مجرد محاجة (2nd) تكون (public) ← حصلت مرة واحدة !!

2

داخل الـ (class)

* هدا دخلنا جوه الـ (classes) فرصة الوصول لأى من القيم بتزيد ☺ !!

Derived02

* الـ (Y) والـ (Z) في Derived معلهمش كلام .. عشان حتى لو اتحولوا لـ (protected) أو (private) هقدر أوصلهم طالما جوه الـ (class) ..
* أو بالـ (getter) وأنا بره الـ (class) ☺ !!

هنلاحظ أن (Y) و (Z) هيتعاملوا نفس المعاملة عشان لو (Y) ← protected
(Z) ← public
ودخلوا على (protected) ..
يبقى هيتعاملوا على هذا الأساس وهينسوا وضعهم القديم ☺ !!

(Z, Y) ⟶ 1st → private في حالة
∴ (Z, Y) in Accessible

Set color() → Might want to name it differently

Summary

Constructor + get color → to get the input value

[1] int main() → Must be public

1. → Derived 02
   → Z → 1st public → Accessible (✓)

2. → Derived02 03
   → Z → 1st, 2nd public → Access (✓)
   → C → 2nd public → Accesible (✓)

2 inside class → Must not be privete

1 → Derived02

→ (Y, Z) → 1st private → inAccessable (X)

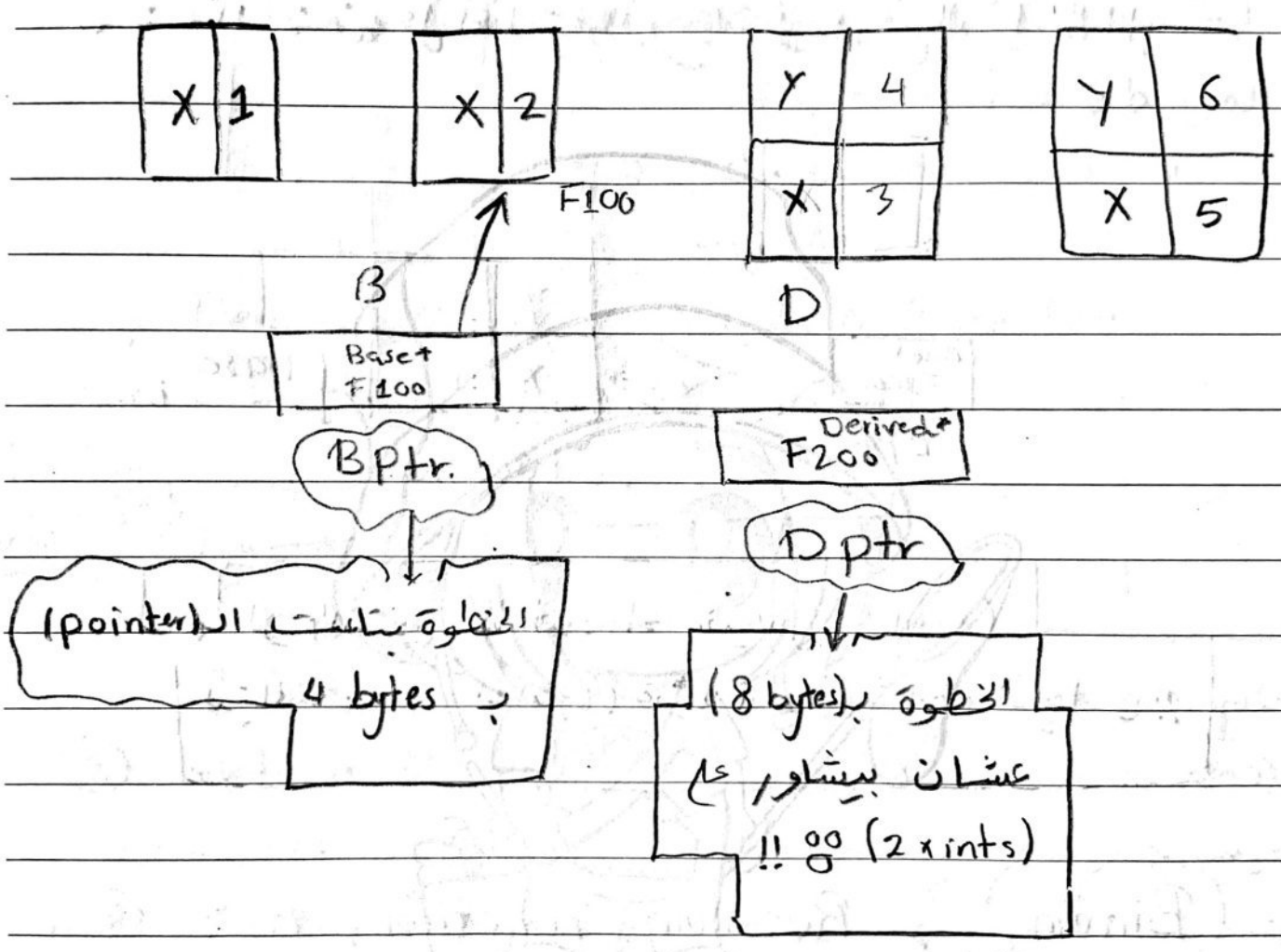
أي اختيار غير كده بالإضافة
لأي اختيار من ال (2nd)
هيخلي ال (Y, Z) ← Accessable(V) عادي جدا

محاضرة

3/11/2022

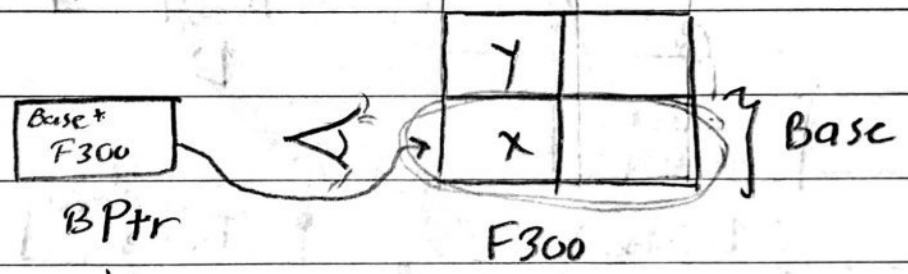
Dynamic Binding

oop-7



* طب ممكن (BPtr) يشاور على (Derived) ؟!

← أيوه نظرياً هيجيب الجزء الـ (Base) منه بس ؟!!
* شغالة في كل لغات البرمجة اللي في الدنيا ؟!!



← لو طلبت show() هيعمل (call)
للباعت الـ (Base) عشان مش شايف غيرها ؟!!

Binding → Resolving Function
to Function Body

← لما الـ (compiler) بيربط (call) الـ (Fn.)
بالـ (Body) اللي انت كاتبه ؟!

30 → shapes Fn. Edit

1 statically → compiled time

→ الـ (Compiler) هيوصل الـ (pointer) بالـ (Fn.) المرتبطة بالـ (Data Type) ..
الـ هو (Base) متاح ☺!!

* Derived can be casted as a Base ☺!!

34 2 Dynamic Binding , Late Binding

→ public in it's class ☺!!

① virtual Method

② Derived inherites (public)

③ Must override on it

الـ (Compiler) مش هيربط الـ (Fn. Call) بالـ (Body) وهيسيبها لـ (operating sytem) يحدد ..
بناء على نوع الـ (object) مش نوع الـ (pointer)

38 → ④ pointer to base = object From Derived

Base * B.Pt

لازم يكون محطوط (virtual) وباقي قواعد الـ Dynamic Binding الأربعة

في حالة محطوط (virtual) على الـ (Fn) هيروح لأحدث نسخة منها :)

طالما بيشاور (Derived) ← هيجيبلك الأحدث

Ex)

BPtr = new Derived (7, 6)

BPtr → Dynshow()

← هيعرضلي النسخة بتاعت الـ (Derived) :)

Develop against Base (Abstract) ←
not against Derived (concreate implementation)

44 → فقرة الأسئلة → نجم سأل عن خطوة الـ (pointer) في حالة الـ (Array) :)

Dynamically Binded Methods

48

أبطأ بسبب أن فيه (Fn-) بتتركن
طد از (Run Time) ولكن مميزتها كتير :)

* طب في حالة (DPtr) بيشاور على (Base object)

خطوته هتكون كبيرة وهيبقى فيه (gap) —
مش متأكد يا هيشتغل يا هيبلك (Error) :)

الباشمهندس قعد يجري
ويعدل في الشروط
عشان يطلعله (Error)
وارد (Dynamic Binding)
يبوظ :) (MCQ)

51 → work around

52 → حالة الـ (Reference)

هيروح لأحدد نسخة برده ؟؟!!

Ex

Derived 02 DD (4,5,6);

Base & BaseRef = DD, *BPtr = ⅅ

هما جبت آخرها !! ☺

* لازم ال (Key word) override

ملوش أى (Functionality) !! ☺ ولكن !!

```
void Dynshow() override
{ }
```

هتطلعلك (warning) لو غيرت اسم ال Fn. مثلاً .. فتقولك ان انت كنت ناوى تعمل (override) بس محصلش ☺!!

SumOfAreas() → Fails on open / closed principle ☺!!

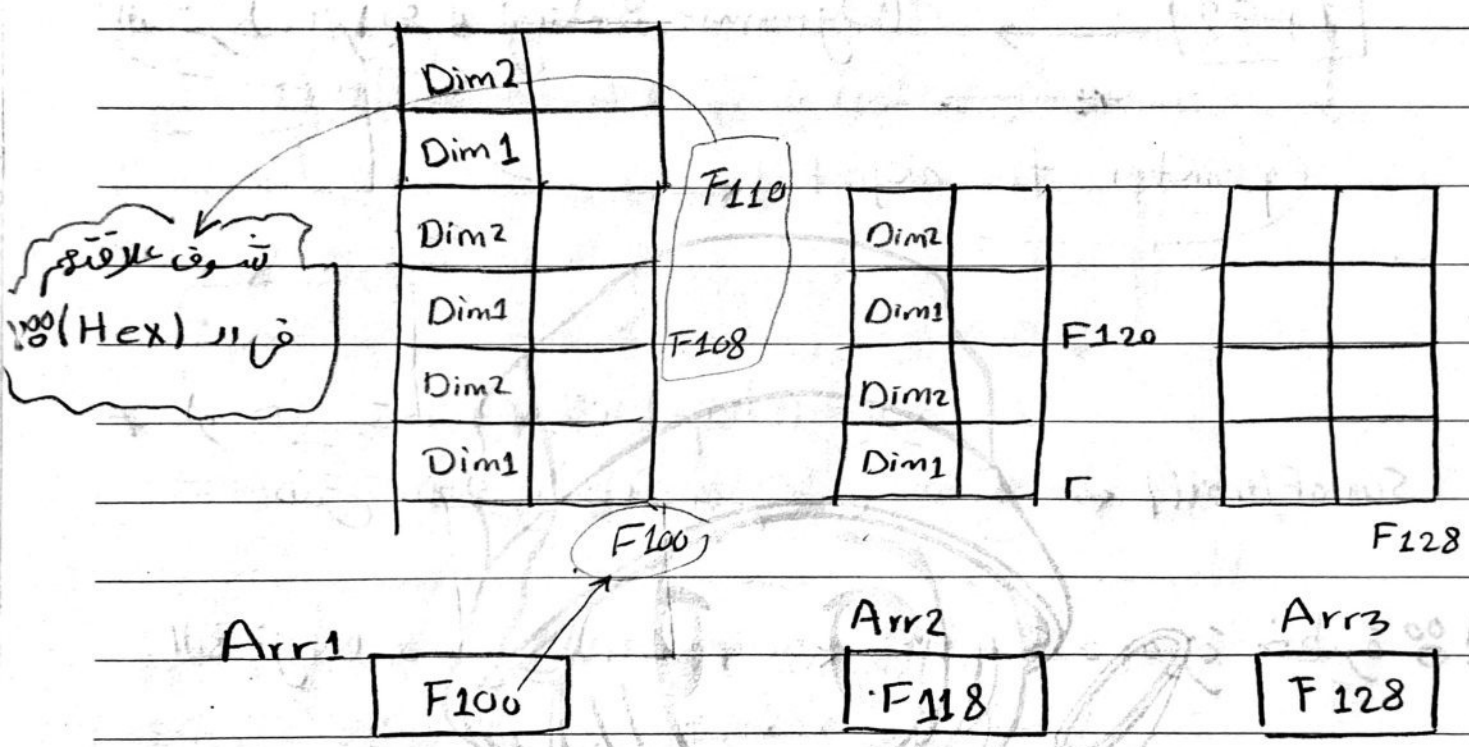
↓

الى اتعملت لـ (geoshape) !! ☺

GArr[i] → CArea()

*(GArr[i]).CArea() ← لاحظ استخدام البسوم

* دول أول 3 سطور في ال main() 100%!!



1:29 → بسبب عدم إضافة أشياء ـ لكن لتوفير سهولة إضافة أشياء لـ Fn الأصلية

1:31 → Pointers and polymorphism

1:32 → الشرط الرابع لـ (Dynamic Binding)

pointer to object

* لو حبيت أضيف (shape) جديد ـ
مش هغير أي سطر من Fn ← SumofArea()

الشغل كله على الـ main() ← وده كده ميزة خطيرة !!

خصوصاً أن غالباً مش بيكون معاك الـ (source code) بتاع الـ (Fn) ـ (1)

1:36 ولكن عاوز أضيف shape (2)

يبقى كده وصلت لـ

!! Closed For Modification (1)
open For Extension (2)

1:40 → Summary

الـ (polymorphism) في أبهى صوره :)

1:41 كان ممكن اكتبها أسهل بالـ (Reference) :)

2:04 ممنوع تستخدم الـ (Reference)
مع الـ (objects) في الـ (C++) أعتقد

ممنوع (Array of Reference)
في الـ (C++) (Search) :)

حل فيديو المحاضرة عشان الحته دي :)

oop 7-2 → 2:05

Pure virtual Method

① Not implementing CArea in Geoshape

② Delegate , CArea implementation

← بجبرار (Classes) اللي جديدة أنها تعمل
ال (implementation)

→ ال class بتاعها اسمه (Abstract class)

→ ينفع ~~ينفع~~ (implementation) لأحدى ال (Methods)
وبالتالي لا يمكن عمل (objects) منه !!

2:13 → Abstract Class Geoshape has ,
no implemntation to pure virtual Fn

→ وهيطلعلي (Error) تاني

2:18:00 → class Rect to be ..
Concrete class Must Fully implement
all Base pure virtual Methods

Abstract class notes

2:16 ① can inheritance

② Geoshape * GPtr;

→ 4 bytes in stack
(pointer to Geoshape)

معملتش (Create)
هجرد بشاور :: (✓✓)

③ Gptr = new Cirlcle (10) (✓✓)

ولكن

2:20 (XX) GPtr = new Geoshape (10,10)

عشان علـ (object)
من (Geoshape) :: ممنوع

2:22 → Quick Question on Abstract

2:27 → Concrete vs Abstract class

هام جداً جداً !!

2:24 → interview Question on Abstract class

ليه عندنا (Constructor) في ال Abstract class و (Destr.)
وأنا مش بعمل objects ؟!

ممنوع (Directly) ولكن بستخدمه
في ال (ctor chaining) داخل ال (Childs)
عشان أعرف ال (Data) بتاعت ال (parent)
ال هستخدموا في ال (child) !!

2:30 → Must implement which?! (Type classes)

2:40 → لحد الوقت ده كان فيه أسئلة عن ال {Abstract Class}

11:42 → Laptop 92%

* (point , Rect) → (Composition) علاقة

طب إيه علاقة الـ (Aggregation)

↳ Stronger Form of Asscosiation

(Class picture)

* علوزين نرسم الأيا جورة ← بس.

picture . paint() → يرسمل كل حاجة !!
لـ ← هم

ليه لازم ال (Setvalue()) Fn. ← عشان ال (ctor) بيشتغل مرة واحدة -- طب لو عاوز أعدل على ال object بعدين

```
class pic
{       Rect *pRect;
        int RNum;
public:
        pic () {RNUM=0;}

        pic (Rect* A)
```

picture → مرتين

sumArray

Where Does Teams store its store

4/11/2022

* pointers and variables can't see what they store until Runtime

Compile Time vs Run Time

Compile Time

عملية تحويل ال (Code) بتاعنا لـ (Machine code)

بيتم فيها مراجعة ال (syntax) ولو فيه حاجة غريبة عنه بيشوفها في ال (Libraries) اللي أنت عاملها (include) ..

بس كده .. هو ميعرفش علاقة ال (code) علاقته ببعض ما بتشتغل هتكون إيه أو نوع ال (Data) المتخزنة

Ex

$X = 5/0 \rightarrow$ No Error at compile Time

Run Time

هنا بقى بنشتغل ال (.exe) اللي عملها ال (compiler) ...

بيتقرأ فيه ال (Data) اللي خزنتها ويبدأ فعلاً يعمل عمليات حسابية

وقتها ممكن يظهر ..

1. Division by zero
2. DeReferencing NULL pointer
3. Running out of memory
4. Fails in opening a program :(!!

وممكن هيطلع (Runtime Error).

→ that's why you wont get an Error

For (int) X = "Hello" → storing char in an int

OR

int X = 5/0 → Division By zero

when compile Time..

but you will get it in Runtime.

but why pointer and not variable ?!

Because → ① you will use Functions and variables can be passed By values.

But using pointers will increase our scope.

② you can Refer to the same space in memory From Different Locations :)

لاحظ ال (Variable name) هش بيتخزن في ال (memory)
لما بتعمل (compile) بيتحول لـ (symbol table) يتدي (instructions)
ال (operating system) وبس ..

4 ints named int

5	4
y	x

Ex 1 variables

```
X = 5;
y = 5;

X = 4;
Cout << y << Endl;
```

هتطبعلي كام؟
(y = 5;) برده
متأثرتش بتغير (X = 4)
هي مش شايفة حاجة غير
البرنتم اللي متسجلها ☺ !!

يعني مساحة ال (variable name) بتكون جزء من
ال (.exe) المتخزن على ال (Hard drive) ...
لكن لما الكود يشتغل (Run-Time) و يبدأ البرنامج
يخزن بيانات البرنامج في ال (RAM) عشان يستخدموها
مش بيحدد مساحة لـ (Variable Name) ☺ !!

هشرح أكتر ☺ !!

غلط برده -- لازم الى جنب العلامة دى (&) يكون (variable) متعرف قبل كده عشان أخد الـ (Address) بتاعه !!

مينفعش الـ (variable) يكون (رقم) !!

Ex (2) → pointers

int *x = &5;

int *x = 5; → wrong

① لازم تعرف (variable) عادى يرتبط اسمه بالـ (Data) -- عشان بعد كده تشاور على الـ Location بتاعه !!

② أو تحدد الـ (memory location) بطريقة (Dynamic) بإستخدام malloc() !!

int x = 5;

int *y = &x;

كده خزنت الـ (Address) فى الـ pointer اللى عرفته على الستاك -- (*) هتبقى عشان الـ DeReferencing !!

cout << y << endl; → prints Address

*y → prints Value

```
int w = 5;

int * x = (&) w;

X = 5;  ⟶  هينفعش تسجل (int) رقم عادى فى (int*) اللى هو (pointer) مخزن (Address) !!

int *y = x;

(pointer) عرفت

خزنت فيه ال (Address) المخزن فى (X) عادى عشان الطرفين (int*) !!

cout << *y << Endl ⟶ (5)
```

لو (w) غيرت قيمتها لـ (10) .. (y *) هتبقى (10) و نفس الكلام (x*) هتبقى (10) ..
ودى ميزة رهيبة فى التعامل مع ال (Fn) وال (Classes) و بتساعد فى تحقيق مبادئ ال (solid) !!

باختصار، with pointers → no scope limitation!!

③ Pointers also used with Arrays

← حتى محتاج أعرف (index) مجرد بزود

ال ptr + (1)

خطوة بقيمة ال (size)

بتاع (one Element) من ال (Array) :)

← يعني في حالة ال (int)

كل مرة بيتحرك بـ (4 Bytes) :)

ولكن في حالة ال (Dynamic) .. أن أعمل ال (Main Method)
تبقى (virtual) كده هيقول لل (operating system) يأخر ال (Binding)
لل (Run time) بحيث يعرف هو بيشاور على إيه ...
وبناء عليه بيشتغل أحدد نسخة منه !!

Lecture 7 (Dynamic Binding) - Re-watch

في حالة ال (Compile time) ال (compiler) لسه
مشافش ال (Data) صح ؟!

فبيعمل (static binding) بينه وبين ال (Fn.)
الى من نفس ال (Data type) بتاعه !!

Ex (1)

```
Base * BPtr;
```

(بترجع (pointer) لل (Location))

```
BPtr = new Base(2);
```

كده عملته object بيشاور عليه !!

```
BPtr -> show()
BPtr -> Dynshow
```

في الحالتين (Base) .. (1) عشان ال (Type) بتاعه (Base)
في هش هيشوف غير ال (Data) الى من نفس ال (Type)
بتاعه ..

(2) وفي حالة (Dynshow) ← بيشاور على (object base)

عبارة عن (Fn) قبلها كلمة (virtual) يعني

Ex(2)

```
Derived *DPtr;

DPtr = new Derived (5,3);

DPtr --> show()
DPtr --> Dyn show()
```

نفس الي فات ← هيشاوروا على حاجات
ال (Derived) في الحالتين يعني

Dynamic Binding Rules

in its class

1. (virtual) and (public)
2. public inheritance to Derived
3. override the Fn. in Derived
4. pointer From Base points at Derived !!

مهمة وبيعمل فيها لعب في ال (...)

هام جداً جداً !!

Ex (3)

نوعه (Base) ...
بس بيشاور على (Derived) ...
هنا يحدث الإختلاف في
حالة وجود (virtual) !!

```
Base *BPtr;
BPtr = new Derived(5, 3);
```

Same as
BPtr = DPtr
OR (Bptr = &D;)

هام جداً جداً !!

```
BPtr -> show();
BPtr -> Dynshow;
```

هيشاور على بتاعت ال (Base) عشان
مفيش (virtual)
وبالتالي هيروح للي
تبع ال (Data type)
بتاعه ال هو (Base)

هيشاور على بتاعت ال (Derived)
عشان محطوط (virtual) على أول (Fn.)
وبالتالي (Dynamic Binding)
فهيستنى لحد ما يعرف نوع ال (object)
اللي بيشاور عليه ..
يلاقيه (Derived) فيشاور على ال (Fn.) بتاعته !!

هام جداً جداً !!

① خطوة صغيرة أفضل --

② طب ال (Data) ؟! كده كده بتكون (private)
ولو عايز توصلها هتستخدم (Fn)

③ طب ممكن العكس ؟!

ولو (virtual) هتنفذ نفس الكلام !!

* طيب أنا استفدت إيه من الكلام ده ؟!

① أن طالما أغلب ال (Fn) بتاعت ال (Base) الرئيسي
قبلها (virtual) --

② وأي (class) جديد بخليه يـ (inherit)
من (Class Base) وأخليه يـ (override) ال (Fn)
بال (implementation) الي تناسبه

③ كده هبقتش محتاج غير (pointer) واحد من (Base)
أوصل بيه لكل الخواص بتاعت ال (inherited classes)

④ بمجرد يشاور على (object) جديد --
هتظهر ال (pointer) هيكبر لآخر واحد
من ال (Fn) الي متسجلة عنده (virtual) !!

هسا جد أجد أ :)!!

هنستخدم أكتر لما نشرح مسألة ال Sum All Area() :)!!

Ex shapes

هيخليك لو عملت (Draw fn) مثلاً
وعملت فيها (pointer) واحد بيشاور على (Draw)

لو حبيت تضيف (shape) جديد زي (Circle) مثلاً

كل ال عليك تعمل (Object) من ال (shape) الجديد
وتخلي نفس ال pointer القديم يطلب Draw()
من نفس سطره القديم..

هيطلبك ال Draw() بتاعت ال (Circle)
الي أنت عاملها..

بكده حققت ال (O) في solid ← open / closed principle

Open For Extension

هسا جد أجد أ :)!!

Closed For Modification :)!!

تسوية تساؤلات وخواطر

1 اشمعنا (pointer) من ال (parent) بيتحكم في كل حاجة ..
ميشفعش (pointer) من ال (child) يتحكم في ال (parent) ؟!

ببساطة لأن السبب بسيط جداً !! ...
ال (ptr) لما بيتعمل لـ (child)
بيبقى هتوقع أن الخطوة بتاعته من المرة الواحدة
هتكون ب ال (size) بتاع Data (parent + child)
بسبب ال (inheritance) ...

وبالتالي لو بيشاور على (Base object) هيلاقي فيه (gap)
في ال (size) اللي بيشاور عليه ..
في ال (compiler) هيرمي (Error) !!

Ex)

غلط

~~Derived * DPtr = new Base (5);~~

في ال (Runtime) عشان ال (Compile)
معترفش ب (syntax) !!

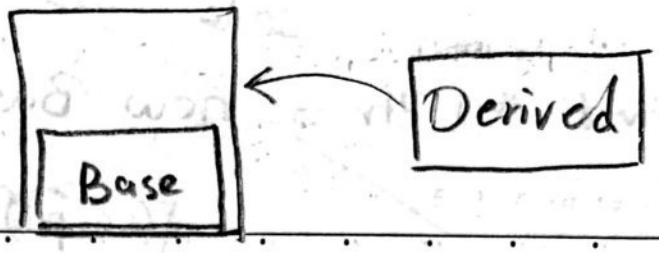
هيديك ال (Error) التالي

"Invalid conversion From "Base*" to "Derived*"

عشان (new) رجعت (pointer to Base object)
ومش عارف يحوله لنسخة (Derived*)
المطلوبة على الشمال ..

عشان حرفياً ال (Base class) مبيعرفش أي حاجة
عن عياله !!
هم بيورثوا منه .. لكن هو ب (class) مبيعرفش
غير نفسه !!

فمبيعرفش إيه (Derived) دي عشان يتحول ليه ..



* لكن في حالة العكس ..

Base *BPtr = new Derived (5,3);

ال (Derived) عارف نفسه ..
وعارف بيانات عن أبوه (Base) ..
وبالتالي

(Derived*) that the new gives From a (Derived object),

→ CAN Be converted to (Base*) ☺!!

Extra note

* ده بالإضافة لأن (DPtr) خطورته كبيرة
على أنه يشاور على (Base) عشان
يحتوي على (Data) ← (Base + Derived) مع بعض
و ال (Data gap) دي السبب من أسباب ال (Error) ☹!!

Every (Derived) is a (Base)

But (not Every) (Base) is a (Derived)

2 بالنسبة لـ (Attributes) عندنا
(constructor chaining)

علاقة الـ (inhentance)

في الـ (child) بيعرف كل أبواته :)

Every (car) is a (vichael)

But not every (vichael) is a (car)

(submarine, Motor cycle, ------)

note

* لاحظ لو فيه (Fn.) في الـ (Derived) بس
مش موجودة في الـ (Base)
← الـ (Base) مش هيشوفها خالص :)

* (static Binding) بيبقى مكتوبة في ال (.exe) من ال (compiler)
هيروح يربط ال (Fn. call) بالفنكشن (Fn. Body) !!

ابحث أكتر في استخدام ال (References)
والفرق بينها وبين ال (pointer) وال (Address) !!

Ex

```
Derived02 DD (4,5,6) ;

Base &, BaseRef = DD
```

→ same as `*BPtr = &DD ;`

ويكون أديت (DD) اسم مختلف و بستخدمه
بردو .. وبالتالي !!

→ `BaseRef.Dynshow () ;`

ويكون ده ال (object) نفسه !!

→ `BPtr -> Dynshow () ;`

`(*BPtr) . Dynshow () ;`

note

Both point at Derived02 Lastest Fn. !!

void DynShow () override

{ }

بتديلك (warnning) لو حطيتش
ال (Fn) عادية ال (override) ---
ممكن عدلت اسمها بالغلط ؟؟

Sum OF Areas

* الميزة الى معايا دلوقتي من ال (Dynamic Binding)
أن لما بشاور على (object) بقدر استخدم ال
(Overrided Fn) بتاعته ؟؟!!

وأنا عندي (Array of objects) لكل (Shape)

① هخلي (pointer) واحد يمسك كل (Arrays)
يعدي على واحدة ويـ (Reset) ويمسك اللي بعدها؟!

② ولا أعمل (pointer) واحد لكل (Array)
وأكون عارف ال (Size) بتاعها بسهولة
عشان أنا عاملها (Dynamic Allocation)
وهيبقى عندي (For Loop) لكل (shape)

لأ غلط -- ده هيخلني أحتاج أعدل على ال (Fn.)
في كل مرة أضيف (shape) عشان أعمله (For loop)
مخصوصة ليه -
ده يعتبر نفس الحل العادي اللي اتعمل قبل كده أصلاً ؟؟!!

هاكا جداً جداً ؟؟!!

الصح

أعمل (Array of pointers) وكل (pointer)
فيها بيشاور على (object) معين
في الـ (Array of objects) ـــ

يعني عدد الـ (pointers) بعدد الـ (shapes)
اللي هحسبلهم الـ (Area) ☺!!

بكده مش هحتاج غير (For Loop) واحدة
جوه الـ (Fn.) ـــ
وبدخلها عدد الـ (objects) والـ (Array of pointers)
وهي تتعامل ☺!!

وفي حالة زيادة (shape) ـــ التعديل هيكون في الـ (main()
بعد ما اتعملـه (class) وبيـ (inherite) من Geoshape
وبيـ (override) الـ (Fn.) اللي بتاعت (Draw()) ـــ
هزود لـه (object) في الـ (Array) وأزود الـ (Object count)
وبس كده ـــ هلمستش (Draw() نفسها ☺!!

الحوار ده هيبقى أسهل كمان لما نعمل
ال (Array of pointers) بتبقى ← (Dynamically Allocated) !!

note → Array of pointers Allocation

1 Static Allocation

Geo shape* Arr [7] ;

يعنى بعمل (Array) ال (Type) بتاع كل (Element) هيكون (Geoshape*)

وده الطبيعى هيكون (pointer) بيشاور على ال "Array "of pointers !!

فـ يعتبر ضمنياً كتبت (**Arr) الى هو (pointer to pointer) !!

implicitly !!

هام جداً جداً !!

ال (new) يعتبر بتأخذ نوع ال (Element)
وتزود عليه (*) ويبقى ده اسم ال (Array) !!

② Dynamic Allocation

Geoshape ** Arr = new Geoshape * [7]

نوع ال (Element) اللي هيشاور عليه !!

نوع ال (Elements) اللي في ال (Array)

في حالة لو (Array) عادية Dynamic برده

struct OR class

```
students *S = new Student [5];
```

OR

```
int * x = new int [5];
```

normal Data type

Fn. implementation

```
double SumOfAreas (GeoShape** GArr, int c)
{       double sum=0;
        For (int i=0; i<c; i++)
{ Sum += GArr[i] -> CArea(); }
        Return sum;
}
```

GArr[i] → int* → Address of an object

(*GArr[i]) → the object

(*GArr[i]).CArea() → Fn. inside of the object

```
int main()
{

    Rect Arr1 = { Rectangle (50,30) }

    Circle Arr2 = { Circle (20,40) }

Geoshape * Arr[2] = { Arr1 , Arr2 };
```

(Element) بتاع أول (Address) ال <o>
same as → &Arr1[0]

So technically it's an <u>Array of Addresses</u>

and the (Arr) is the main pointer that Reaches Everything ☺!!

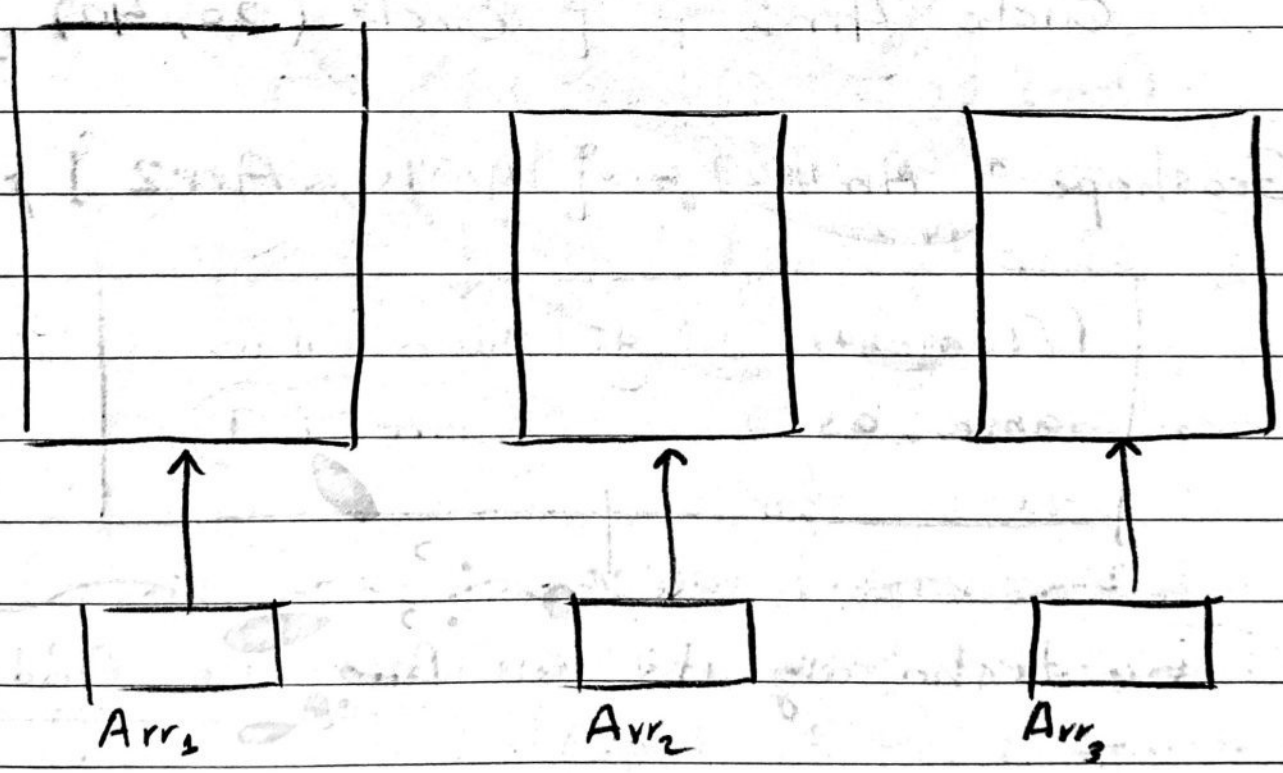
Variables استخدام *

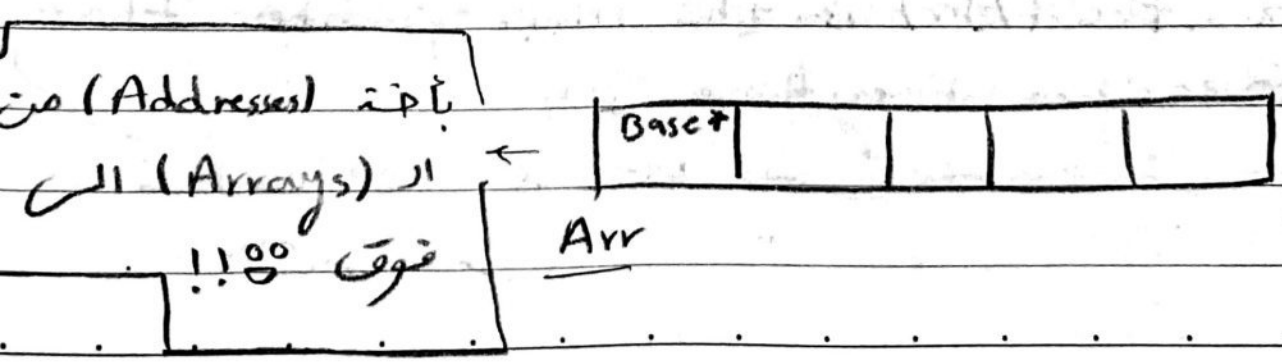
and At Last ..

no. of shapes For the For Loop counter

SumOFAreas (Arr , 2) ;

هكذا بالضبط !!





On using variables

Note

```
Geoshape G(10,10);

Square S (20);

G = S;

cout << G.CArea() << endL;
```

هتأخذ الجزء اللي تعرفه من (S) و هتنام على كده :)

عشان كده السطر ده هيطبع (Zero)
عشان دي الـ (Fn.) بتاعت (Geoshape)
اللي أنا حاطها الـ (implementation) بتاعها
بـ { Return 0; } وبس على كده :)

فـ الـ (variable) كل مرة هيتعامل (static Binding)
قدراته مش هتستحمله بغير كده :)

هتاخد جدا جدا :)

① Code once ③ input is Type (Base)

② Develop against parent ④ passing any number of child objects

* ملحوظة مينفعش تعمل (Array of Reference)

بدل ال (Array of pointers) ..

عشان ال (Reference) لازم يكون بيعمل (Alias)

لـ (variable) ..

بس معندوش (variables) أصلاً ..

عشان كده هيديك (Run time Error)