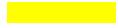


Lecture 14

Lecture contents:

- 1-Recap on binary trees
- 2-Client code example
- 3- 



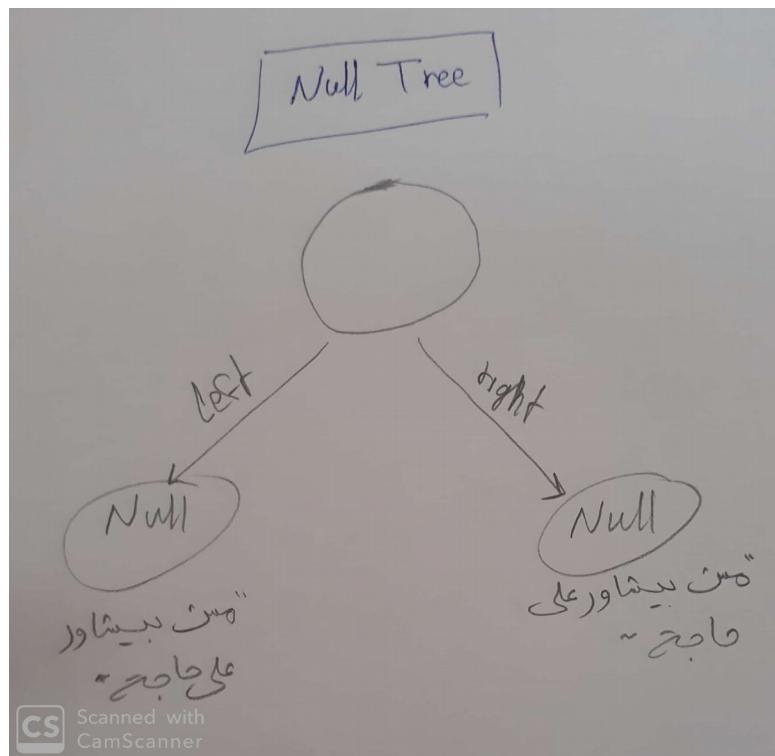
فلنكلم الهراء الذي بدأناه المحاضرة اللي فانت

كنا بنتكلم عن ال trees وعملنا ال constructor بتاعها وبعدها عملنا ال insert وكان عندنا ليها 2 cases :

Case 1:

عمل insert على null tree .. نفتكر ايه هي ال null tree :

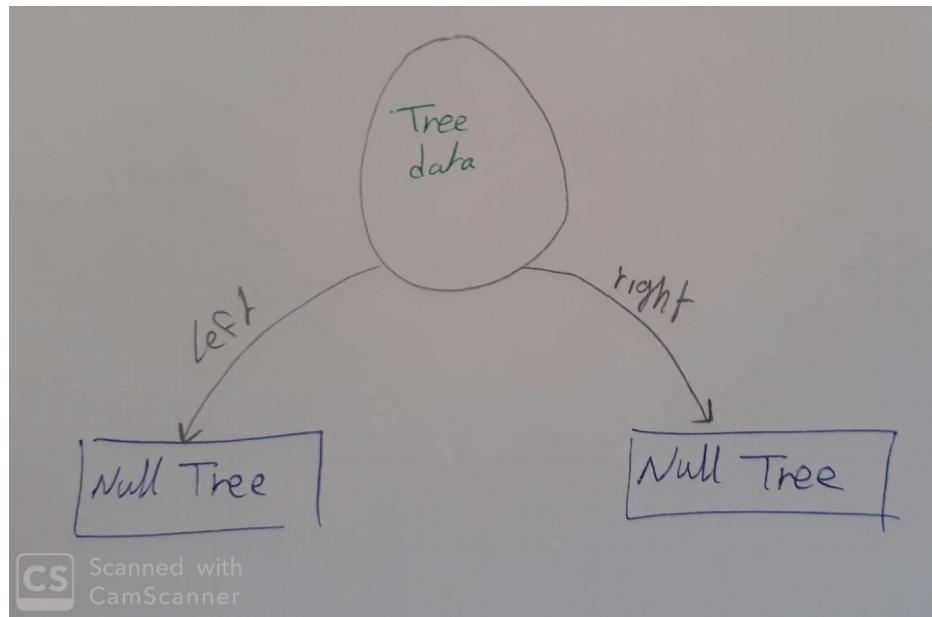
دي root ال sub tree بتاعها مفيهوش اي data ومش بتتشارو على حاجة خالص (NULL)



في ال case دي هنعمل ايه؟؟؟

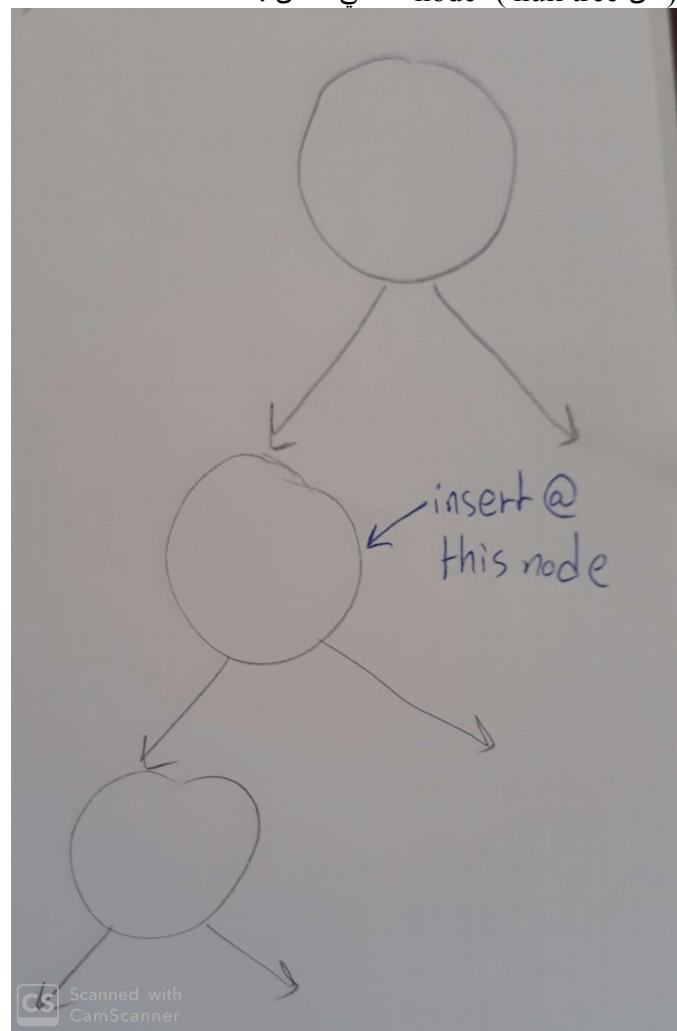
المفروض هن insert الداتا في ال null tree او ال node دي .. بس معنى اني حطيت فيها داتا ان هي خالص ما بقىتش null يعني ملينفعش ال left & right بتوعها يشاورو على ولا حاجة .. طب امال هخليهم يشاورو على ايه؟؟؟

هخلي كل branch يشاور على null tree لوحده (بدل ما كان يشاور على null) .. وبكرة ال null tree القديمة دي لما عملت insertion عليها حطيت فيها داتا وخليت ال branches بتوعها يشاورو على null trees (أينعم هم بس متتساش ان ال (sub tree هي اصلا null tree)



Case 2:

انا واقف عند node .. (مش null tree ، كدة في النص :



ساعتئها اننا كل اللي هعمله اني هـ update الداتا اللي جوة ال node دي بالداتا الجديدة اللي عايزة احطها وخلاص كده شكرأً راجع بقى الكود بتاعها في او اخر المحاضرة اللي فانت

وبعد كدة عملنا حاجة زي make left وقلنا ان الفرق بينها وبين ال insert ان make left بتعمل في حالة ان لو كان عندنا leaf tree او عندنا left tree بتعالها عباره عن null tree .. فهنا برضه في ال make left cases عددي 2 :
 اول واحدة ان ال node اللي عايزة اعمل لها left tree دي كان ليها left tree اصلا .. عشان كدة في الاول بنشفوف لو ال case دى متحقق .. لو آه بيقى مينفعش نعمل make left ونخرج من ال function لأن فيه pre condition مش متحقق .. طب ولو لا؟؟ هـ ال left اللي موجود وبعد كدة احط ال left الجديد اللي عايزة احطه ((ممكن اعمل كدة بإنني delete اللي هيرجلي من ال left اللي اسمها left بما انها كدة كدة بترجع ال pointer اللي بتشاور عليه . او بإنني اعمل delete leftTree وخلاص .. الاثنين ينفعوا بس في السلايدز احنا مستخدمين الطريقة الاولانية)) :

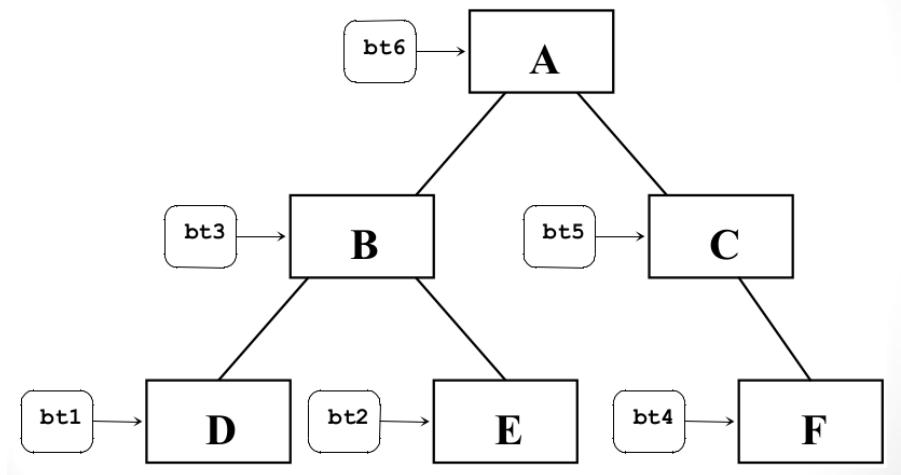
makeLeft()

```
template < class btElementType >
void BinaryTree < btElementType >
:: makeLeft(BinaryTree * T1)
{
    assert(!isEmpty());
    assert(left()->isEmpty());
    delete left(); // could be nullTree true, w/data
    leftTree = T1;
}
```

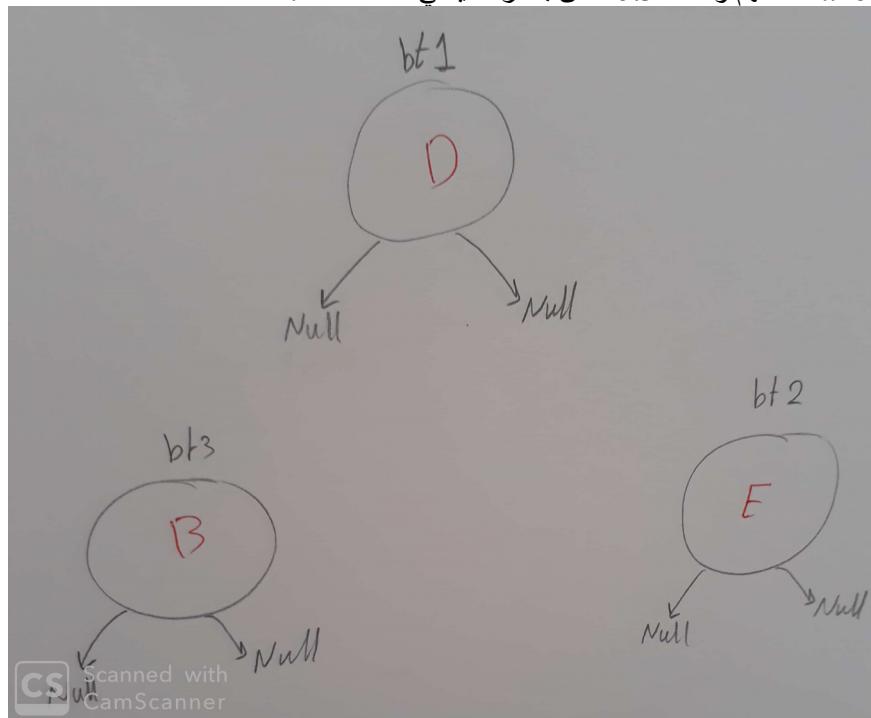
وبعدها اتكلمنا عن ال traversal وكتبنا ال ADT بتو عال file.h & file.cpp بتاعنا ولوقتي عايزيين بقى نستخدم ال ADT او نديها لل user يستخدمها في الابلكيشن بتاعه

اعتبر معايا اننا عايزيين نعمل ال tree دي :

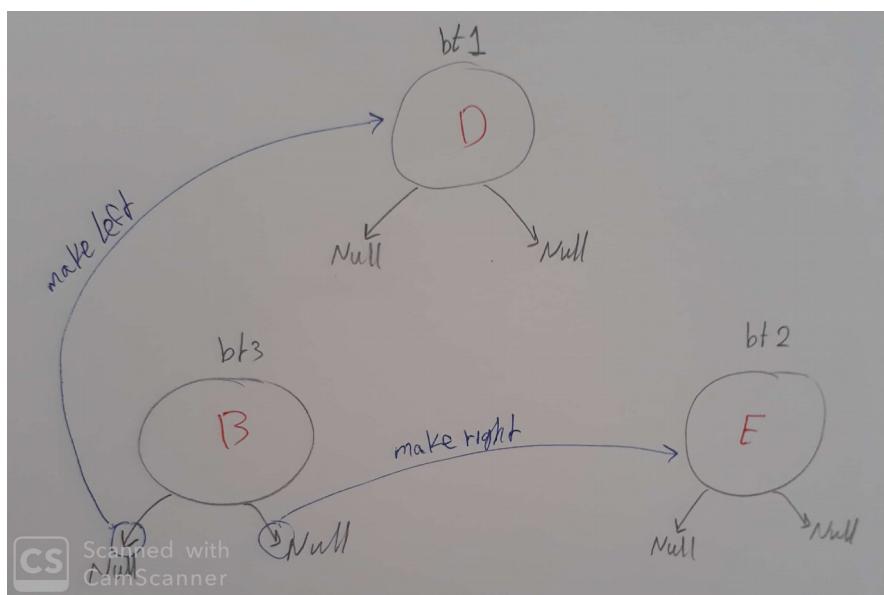
The Operation of Client Code Example



الموضوع بسيط .. احنا هنعمل new nodes وكل واحدة ن insert فيها الحرف اللي احنا عايزينه ونربط ال nodes دي ببعضها تعالى نبدأ ب bt1 & bt2 .. هنعملهم ونعمل bt3 كمان بالمرة، يعني هنعمل كدة :

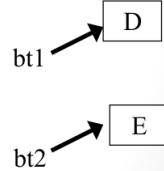


بعد كدة نقوم مخلين bt1 هو ال left بتاع bt3 و bt2 هو ال right بتاع نفس ذات ال : bt3



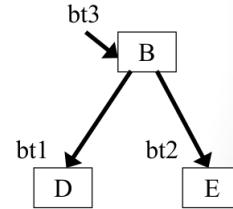
Simple Client for Binary Tree

```
int main()
{ typedef BinaryTree < char > charTree;
  typedef charTree * charTreePtr;
  // Create left subtree (rooted at B)
  // Create B's left subtree
  charTreePtr bt1=new charTree;
  bt1->insert('D');
  // Create B's right subtree
  charTreePtr bt2=new charTree;
  bt2->insert('E');
```



Create Tree

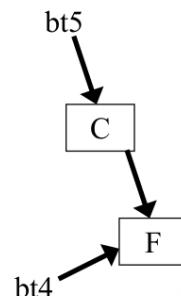
```
// Create node containing B, and link
// up to subtrees
charTreePtr bt3=new charTree;
bt3->insert('B');
bt3->makeLeft(bt1);
bt3->makeRight(bt2);
// ** done creating left subtree
```



: bt4 & bt5 ونعمل نفس الحوار عشان نعمل

Create Right Subtree

```
// Create C's right subtree
charTreePtr bt4=new charTree;
bt4->insert('F');
// Create node containing C, and link
// up its right subtree
charTreePtr bt5=new charTree;
bt5->insert('C');
bt5->makeRight(bt4);
// ** done creating right subtree
```



نلم العيال دي كلها بقى ونربطهم بأيوهم عشان كفاية قلة ادب بقى :

Create the Root of the Tree

```
charTreePtr bt6(new charTree);
bt6->insert('A');
bt6->makeLeft(bt3);
bt6->makeRight(bt5);

// print out the root
cout << "Root contains: " << bt6->getData() << endl;
```

هنا بقى نقف وقفه صغيرة .. لو بصيت على تعريف `bt6` اللي فوق هتلacihe مكتوب بطريقة مختلفة عن اللي احنا نعرفها .. بس دة عشان الدكتور يفهمنا ان الطريقتين ينفعوا وزي بعض بالطبع :

```
charTreePtr bt6 (new charTree); == charTreePtr bt6 = new charTree;
```

بعد كدة بقى بنعمل شوية حاجات مالهاش لازمة بس هتخلينا عارفين ازاي نـ `node access` كل

Print Left and Right Subtrees

```
// print out root of left subtree
cout << "Left subtree root: " <<
bt6->left()->getData() << endl;

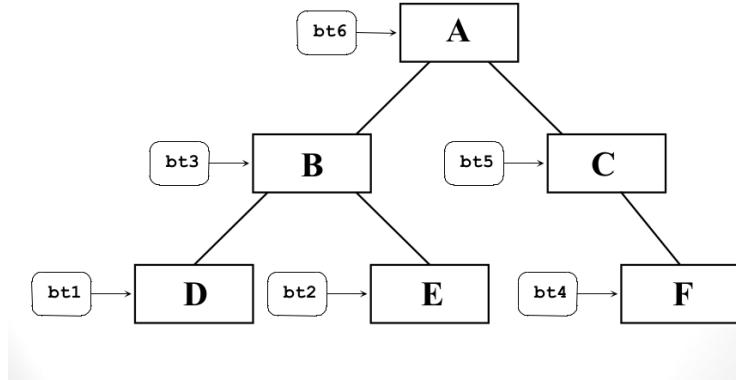
// print out root of right subtree
cout << "Right subtree root: " <<
bt6->right()->getData() << endl;
```

Print Extreme Child Nodes

```
cout << "Leftmost child is: " <<  
bt6->left()->left()->getData() << endl;  
  
cout << "Rightmost child is: " <<  
bt6->right()->right()->getData() << endl;  
  
return 0;  
}
```

وبكدة احنا خلاص جمعنا الشجرة بتاعتنا :

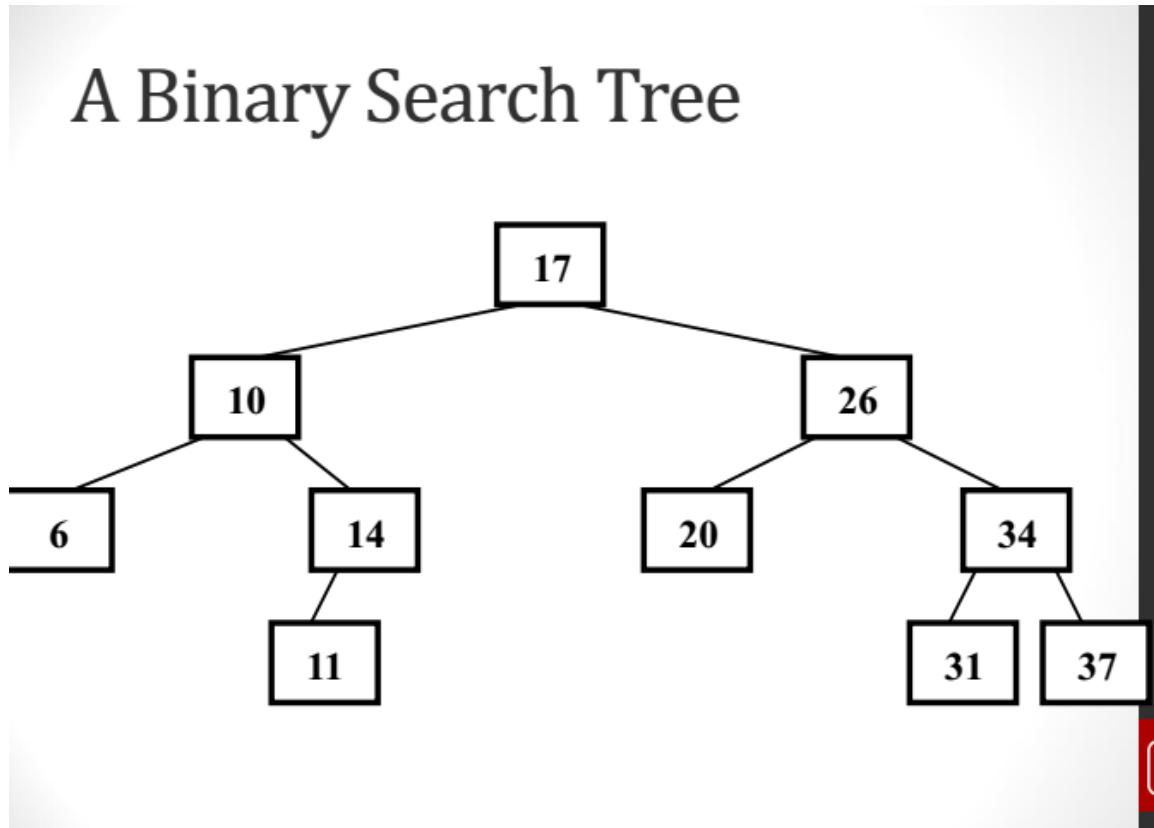
Final Product



كدة احنا خلصنا ال binary trees فقوم نحرق هالشجرة بقى ونزرع واحدة أشرف :

دلوقتى هنبدأ في حاجة جديدة وهي الـ BST(Binary Search Tree) .. هي زى الـ tree العادية بس من اسمها بتسهل الـ search يعني ايه؟ يعني لو انت بتدخل الـ data بتاعتك .. استخدمت الـ data structure دى .. لما تيجى تعمل serch هيتعمل بسرعة.

ده مثل لـ BST



افرض عايز تجيب الـ element اللي الـ key بتاع 11؟

عاده في الـ BST مش بنخزن data بس لا بنخزن pair of key and data

ايهد في كده .. ان الـ key ده بيحدد الداتا فين في الـ tree .

يعنى مثلا انا الـ data بتاعتي هي أسماء طلبة في الـ section فهخلی الـ bn هو الـ key فلما أقوله هات الـ data بتاعت الـ key 12 مثلا .. هيروح يجيب اسم الطالب اللي الـ bn بتاعه رقم 12.

نرجع للسؤال دور على الـ element اللي الـ key بتاعه 11

لو انا في الـ tree القديمة .. هعمل ايه؟

هعمل traverse فهمعل visit للـ nodes كلها لحد ما الاقي الـ 11 ... هنا بقى الوضع مختلف..

في الـ BST اللي فوق هتلaci الـ root 17 .. لو الرقم اللي بدور عليه اصغر من الـ node اللي انا واقف عندها بيقى اكيد على الشمال ولو اكتر بيقى على اليمين

بيقى في السؤال ده انا واقف عند 17 وبقوللك دور على 11 هتدور فين .. هدور على الشمال اكيد ... هروح شمال هلاقى 10 .. بيقى لو الـ 11 موجودة هتبقى فين بالنسبة للعشرة .. هتبقى على اليمين عشان الـ 11 اكتر من الـ 10

.. لما رحت يمين لقيت 14 بيقى لو موجودة هتبقى على شمال الـ 14 فهمنا اللعبة بقى خلاص .. والحمد لله لا قيناها فعلا.

طب لو بدور على 12 .. همشى بنفس الطريقة لحد ما هوصل للـ 11 المفروض انى اروح يمين بس هلاقى ان ده اخر الـ tree اذا الـ 12 مش موجودة.

خد بالك ان لازم الـ tree تبقى balanced tree يعني ايه ؟

يعنى الـ 17 في المثال اللي فوق لازم تبقى في النص عشان تسرع الـ search لكن تخيل ان الـ 6 هي اللي فوق ..
عمل insert كله هيفي يمين هتلافق انها بقت list عادية.

فلو عندك unbalanced tree في طرق تخليها ... مش هنتكلم عنها بس في طرق كده.

An Ordered Tree ADT

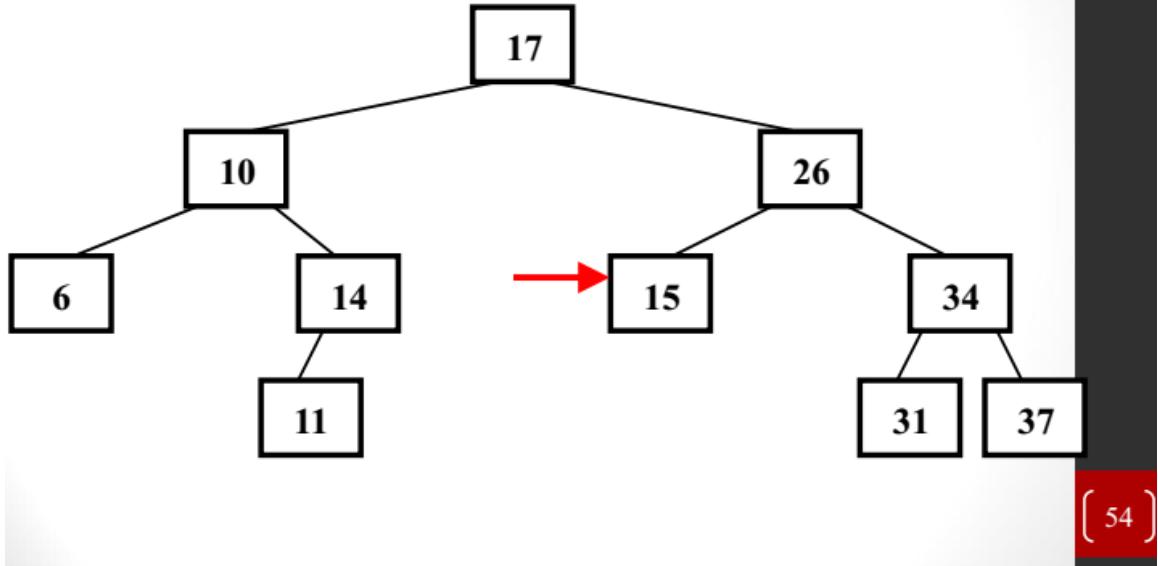
- BSTs are ordered
- BSTs provide for fast retrieval and insertion
- BSTs also support sequential processing of elements

Definition of BST

- A Binary Search Tree (BST) is either
 1. An empty tree, or
 2. A tree consisting of a node, called the root, and two children called left and right, each of which is also a BST. Each node contains a value such **that the root is greater than all node values stored in its left subtree and less than all values stored in the right subtree.**
- The BST **invariant**
 - The invariant is the ordering property
 - “less than goes left, greater than goes right.”

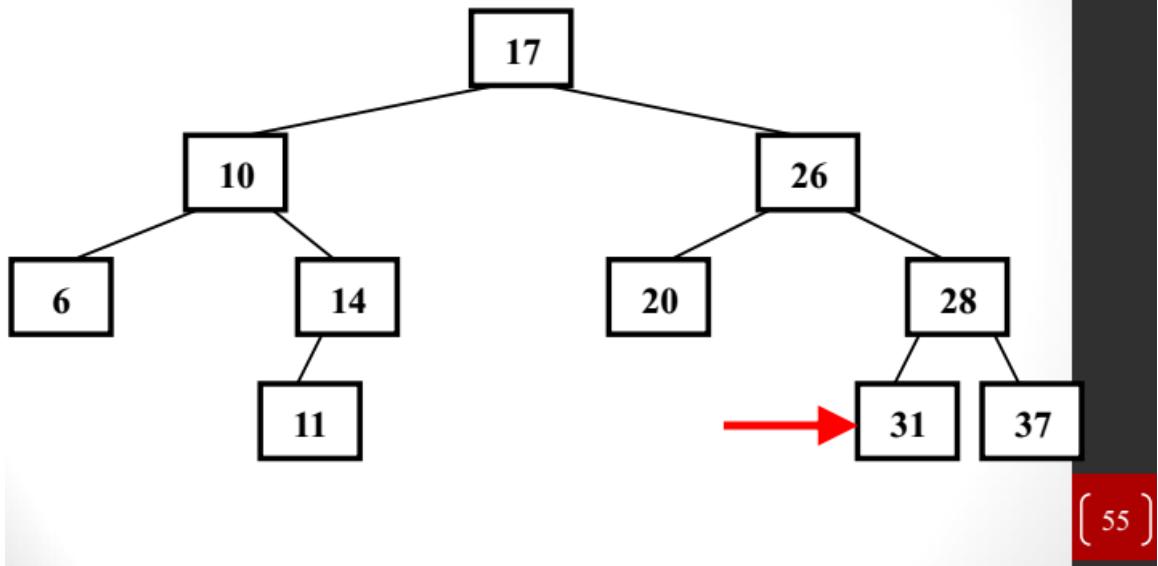
تعالوا نشوف تطبيق على الكلام ده.

Not a BST: invariant is violated



هنا ال 15 يمين ال 17 فدى متنفعش . BST

Not a BST: subtree is not a BST



هنا في ال subtree اللي على اليمين .. هلاقى ال 31 اكتر من ال 28 ومع ذلك شماليها .. بيقى دى مش .BST

Binary Search Tree ADT

- **Characteristics**

A Binary Search Tree ADT T
stores data of some type (btElementType)
Obeys definition of BST (see earlier slide)

- **Prerequisites**

The data type btElementType must
implement the < and == operators. (operator overloading)

[56]

الجملة اللي في الآخر بتقول اننا هنحتاج نقارن بين الـ nodes .. هنشوف ازاي قدام.

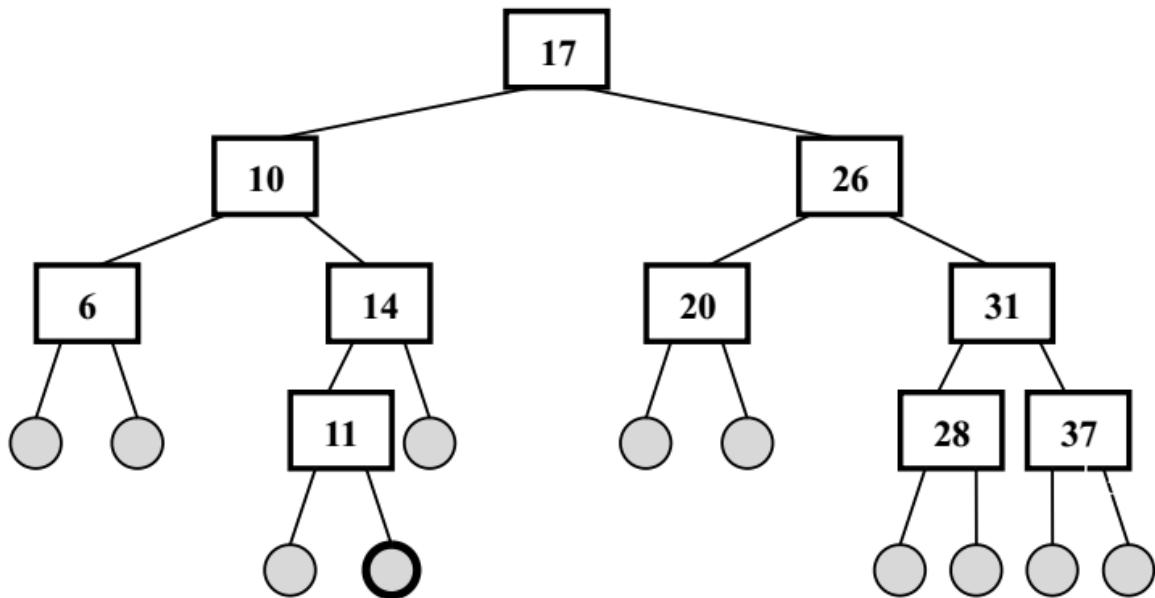
BST ADT Operations

- **isEmpty()** // check for empty BST
- **getData()** // accessor
- **insert()** // inserts new node
- **retrieve()** // returns pointer to a BST
- **left()** // returns left child
- **right()** // returns right child

[57]

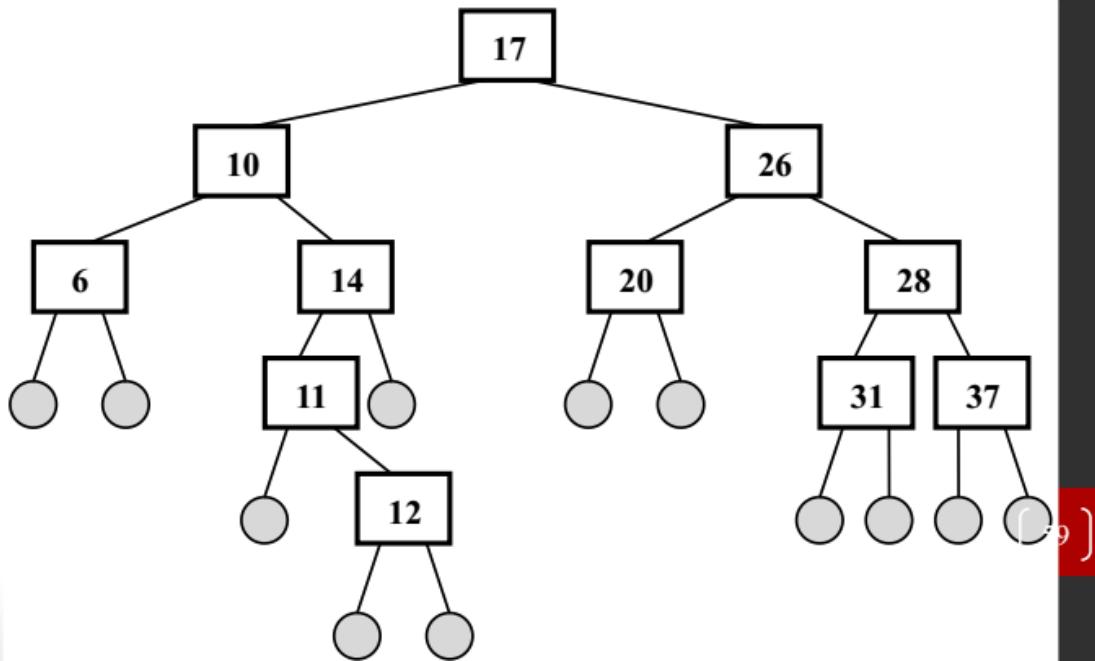
دي ال member functions بتاعت ال BST .. هنلاحظ مفيش makeright ولا makeleft علشان انا هنا مش ب insert بيمزاجي .. انا عايز insert في المكان الصح.

Where 12 would be inserted



دي قلناها قبل كده لما ال 12 مكنتش موجودة المره دي انا مش بدور عليها انا هعملها insert فهتبقى في المكان bolded ده.

After 12 inserted

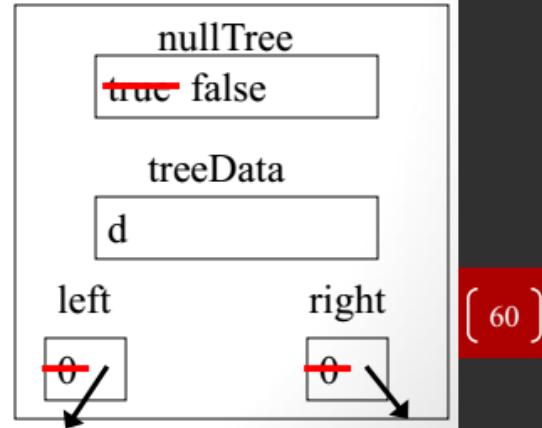


ده بعد الـ insert زى زمان بردو بنحط الـ nulltree عادى.

نشوف بقى الكود بناء الـ insert بس خد بالك اننا هنا مش هن insert في نص الـ tree يعني لازم نبقى بن insert في اخر .children

insert()

```
template < class btElementType >
void BST < btElementType >
:: insert(const btElementType & d)
{
    if (nullTree) {
        nullTree = false;
        leftTree = new BST;
        rightTree = new BST;
        treeData = d;
    }
}
```



insert (if not empty)

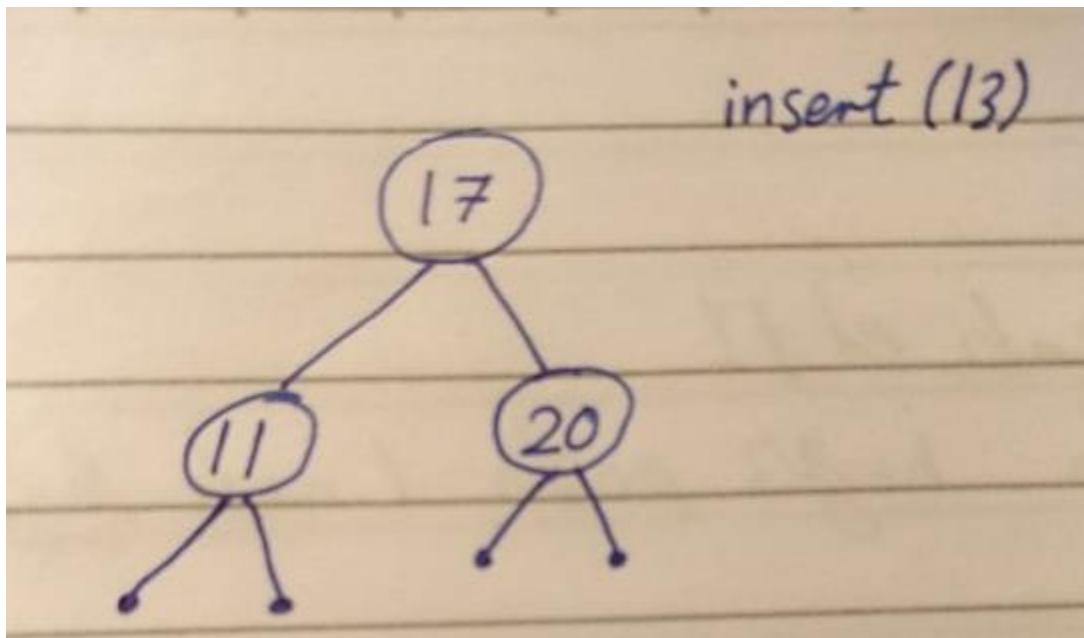
```
else if (d == treeData); // do nothing -- it's already here!
else if (d < treeData)
    leftTree->insert(d); // insert in left subtree
else
    rightTree->insert(d); // insert in right subtree
}
```

[61]

كل الحكاية ان لو ال tree مفهاش حاجة .. هبدأ احط ال root واعمله ال left وال right واخلى ال bool اللي بيقوللى ال tree فاضية وللا لا ب false

ولو مش هبأها وهي already فيها insert هشوف هل الـ data عندى في الـ tree لو عندي خلاص مش حطها تانى ولو مش عندي هقارن بقى عشان احطها في المكان الصح. لو الـ data اللي عملها اصغر من الـ node اللي واقف عندها بيقى حطها في الـ left لو اكتر حطها في الـ right.

ناخد مثال نفهم بيه الـ function دى ماشية ازاي؟



مثلاً عندنا الـ tree دى وعايزين نعمل insert(13)

في الأول الـ root هو اللي هيئنا على الـ insert دى ... ندخل بقى جوة الـ function هل الـ 17 دى null tree؟ لا .. طيب هل الـ $13 == 17$ لا ... طب هل الـ 13 اقل من 17 .. اه بيقى هننفذ الجملة دى

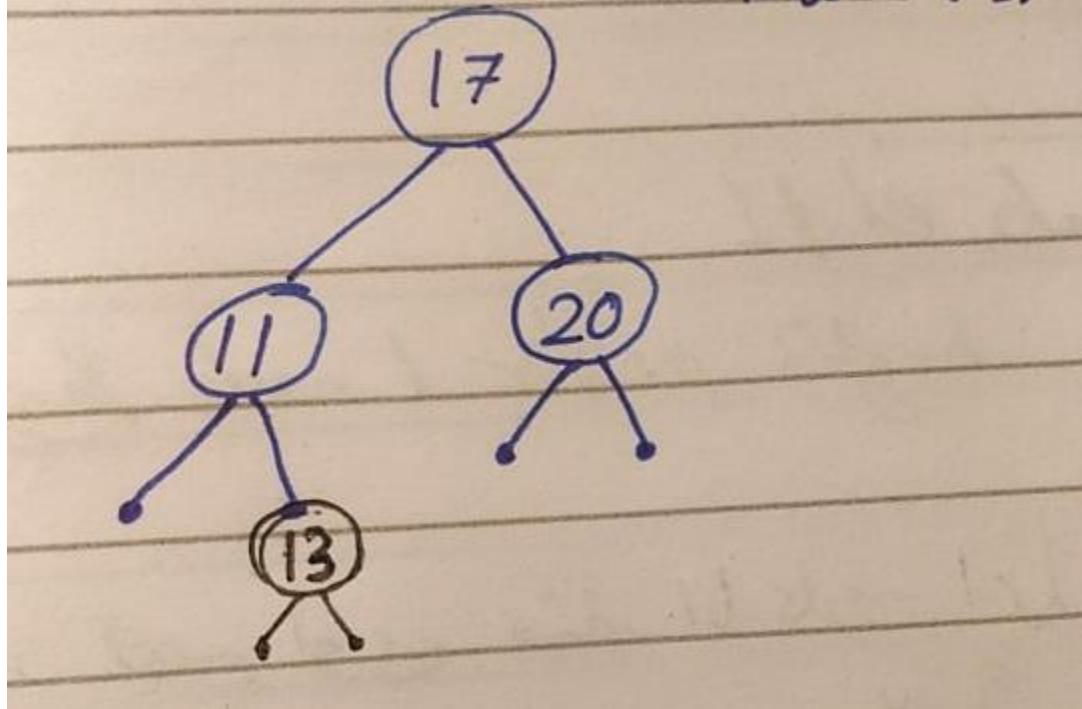
`leftTree->insert(13);`

باقى دلوقتى الـ pointer اللي بيشاور على 11 هو اللي هيئنا على الـ insert
هل الـ 11 null tree؟ لا .. طب هل الـ 11 بتساوى 13 لا .. طب هل الـ 13 اصغر من الـ 11؟ لا .. طب هل الـ 13 اكتر من الـ 11؟ اه بيقى هننفذ السطر ده

`rightTree->insert(13);`

يعنى الـ pointer اللي بيشاور على يمين الـ 11 هو اللي هيئنا على الـ insert
ندخل تالت جوة الـ function هل دى null tree؟ اه .. بيقى هخليها مش nulltree وهحط الـ 13 أخيراً.

insert (13)



استنى رايح فين .. لسة مخلصناش ..

دلوقتى احنا خلينا ال nulltree مش nulltree والكلام ده كان ضمن لما ال rightTree ندهت فخلصناها وخرجنا
للى نده على insert قبلها اللي هو كان ال leftTree اللي هو ال pointer اللي بيشاور على ال 11 هنطلع برة
ال if else ون return فرجع للى نده قبلها وهو ال pointer اللي بيشاور على ال root فن وكمه خلاص
خلصنا .. شوف كنت عايز تزوح فين بقى.

عشان هنبدأ ال retrieve

retrieve()

- **BinaryTree T.retrieve(btElementType d)**
- *Precondition:* T meets the BST invariant.
- *Postcondition:* T meets the BST invariant.
- *Returns:* if T contains a node with data d,
then T.retrieve(d).getData() == d;
otherwise, T.retrieve(d).isEmpty().

[62]

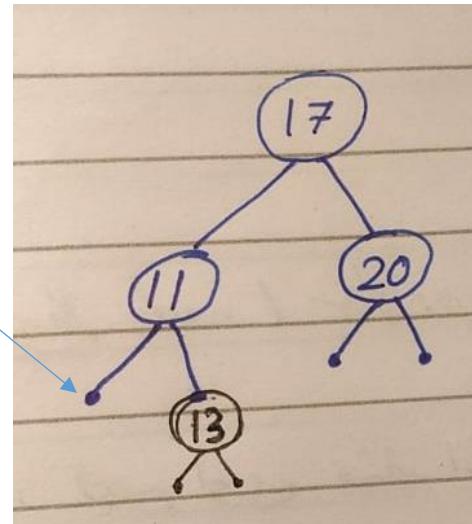
دی بندور بیها علی data معینه.

retrieve()

```
template < class btElementType > BST < btElementType > *  
BST < btElementType > :: retrieve(const btElementType & d)  
{  
    if (nullTree || d == treeData)  
        // return pointer to tree for which retrieve was called  
        return this;  
    else if (d < treeData)  
        return leftTree->retrieve(d); // recurse left  
    else  
        return rightTree->retrieve(d); // recurse right  
}
```

[63]

لو كنت واقف عند nulltree يعني في اخر حاجة في ال tree زى اللي بيشاور عليه السهم تحت مثلا



او كنت واقف عند `node` اللى بتاعتھا بتساوی الـ `data` اللى بدور علیها.
هرجع `pointer` بیشاور على الـ `node` اللى واقف عندها.

طب افرض انا كنت واقف عند الـ 11 وبدور على الـ 13 .. فلا انا وقف على `null tree` ولا نفس الـ `data` كمل
الـ `function` وھسوف هل الـ 13 اقل من الـ 11؟ لا بيقى

`Return rightTree->retrieve(13);`

هدخل `retrieve` تانى بس المرة دى انا وقف عند الـ 13 هل دى `nulltree` او نفس الـ `data` اللى بدور علیها .. اه
هرجع الـ `pointer` اللى مشاور على الـ 13.

مش محتاجة اقوللك انك لسة هت `return` من كل `retrieve` دخلتها عشان ده `recursion` زى ما عملنا فى
`insert`

نشوف مثال للـ `main`

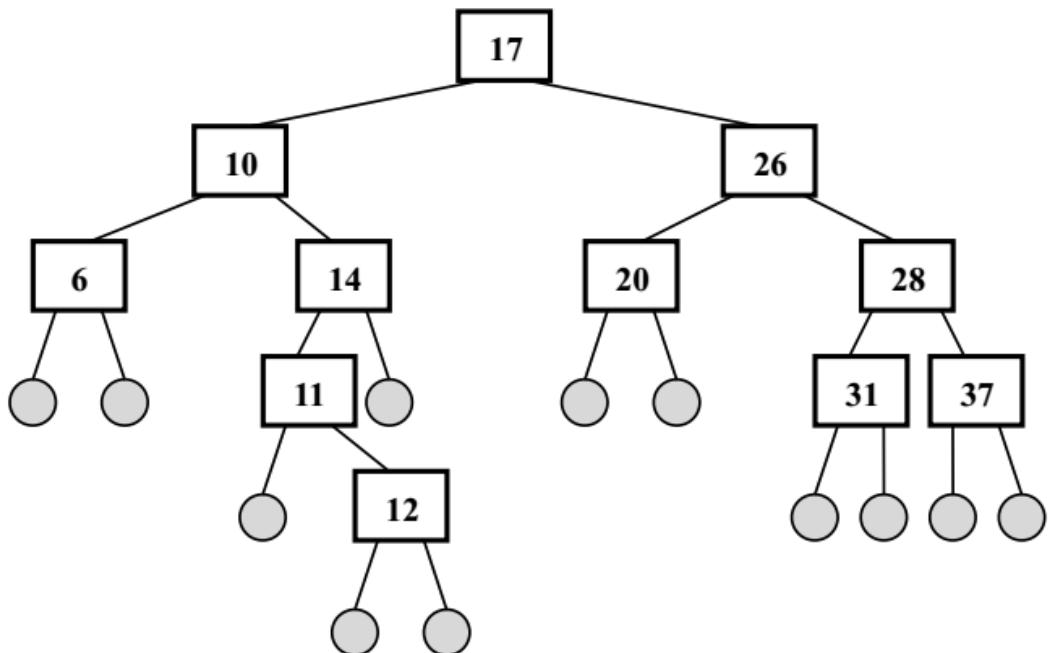
Client of BST

```
int main()
{
    typedef BST< int > intBST;
    typedef intBST * intBSTPtr;

    intBSTPtr b(new intBST);
    b->insert(17); b->insert(10); b->insert(26);
    b->insert(6); b->insert(14); b->insert(20);
    b->insert(28); b->insert(11); b->insert(31);
    b->insert(37); b->insert(12);
```

[64]

BST Result



[65]

دى result main جرب تعملها هو الدكتور عادها .. متكلمش عنها غير العنوان

ندخل في حاجة تانية

ما علاقة الـ BST بالـ binary tree ؟

فهي ممكن نقول انها binary tree من الـ special case او

BST is a binary tree

وعليها شوية حاجات

دي اسمها

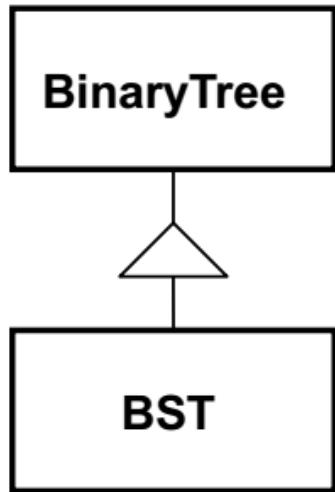
“is-a” Relations

- Defining one abstraction in terms of another.
- The Binary Tree ADT is a general class of which binary search trees are one type.
- Therefore, if we define a Binary Tree ADT we should be able to define a BST as a special case which inherits the characteristics of the Binary Tree ADT.
- A BST “is a” Binary Tree, with special features
- So, the BST is a derived class of the Binary Tree base class

(69)

بيمثلوا ها بالشكل ده

Inheritance Diagram



[71]

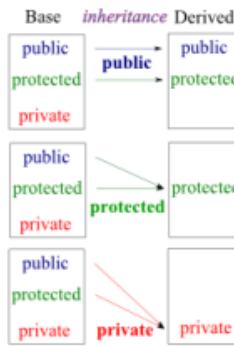
يعنى الـ BST inherits binary tree

بنسمى الـ child class هي الـ class او الـ original class والـ parent class هي الـ BST والـ binary tree والـ derived class

في تلت أنواع من الميراث

Inheritance Terminology

- Base class - the one from which others inherit
- Derived class - one which inherits from others
- So, the BST is a derived class of the Binary Tree base class.



{ 70 }

تعالوا نحاول نفهم من الصورة اللي تحت الـ **base** هو الـ **original** يعني طب هو الـ **original** عنده ايه يورثه؟
الـ **member functions** والـ **member variables** وال حاجات دى متوزعة ل حاجات **public** و حاجات **private** في حاجات للعامة **public** وفي حاجات لعيالى بس يشوفوها **protected** وفي حاجات محدث يشوفها **private**.
افرض انا عندي أولاد مختلفين وعايز اورثهم باساليب مختلفة.
فهيبيقى عندي التالت أنواع دول حاجة اسمها **public inheritance** يعني الـ **public** في الـ **base** يروح **public** في الـ **derived** والـ **protected** في الـ **base** يروح **protected** في الـ **derived** في الـ **private** محدث ليه دعوة بيه.
الكلام ده معناه ان الـ **interface** زى الـ **derived** بتاعك **interface** **derived** لـ **interface**.
ثانى نوع هو الـ **protected inheritance** ان الـ **public** والـ **protected** يروحوا

ده بنسخدمها لما ابقي عامل ال base مش عشان بيقى public والناس كلها تشووفه لا ده خطوة بس عشان
derive منه classe تانية تبقى حاجتها protected. وهكتب انا بنفسى ال public interface بتاع
ال derived class ده اللي حاجته protected.

ثالث نوع هو ال private وهو ان ال public وال protected base في ال private وال protected class

يعنى هنا لو عملت inheritance تانى ال derived الأخير ده مش هيشوف خالص ال public وال protected
بتاع الاولاني.

Public Inheritance

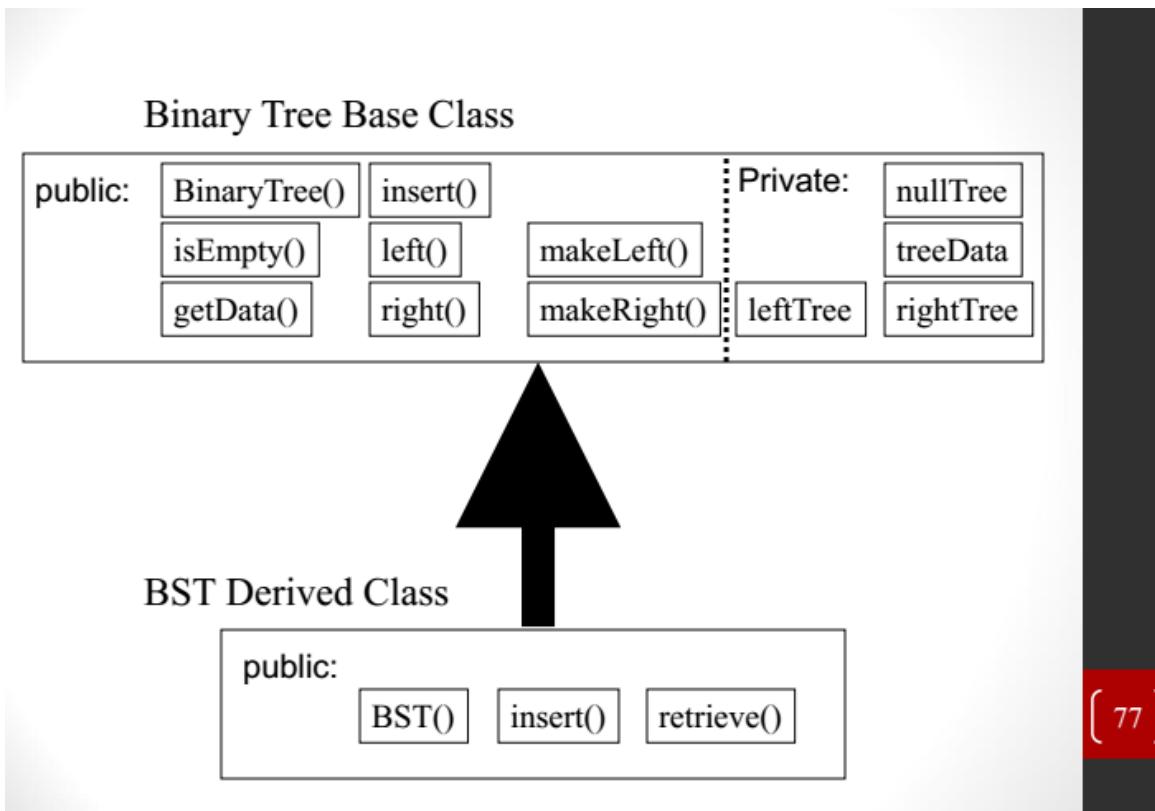
- Inheritance notation is done like this

```
class BST : public BinaryTree
```

- public means that BST client code must be able to access the public members of the base class. (public inheritance)
- It means that derived functions will only redefine member functions that they must (like insert() for BST). All other functions will be the public ones in the base class.

[74]

اول جملة يعني كده ال BST هتوريت ال public وال private بتاع binary tree يعني هتاخذ كل
ال functions



يعنى كل الـ `functions` اللي عند الـ `binary tree` هاخدتها بس في حاجة... الـ `insert` بناعت الـ `child` مختلفه عن الـ `base` وكمان معايا زياده `.retrieve`.

هنجعل ايه في حوار `insert` ده ... هنجعل حاجة اسمها `polymorphism` .. يعني هييفي عندي الـ `insert` الثانية دى بنفس الاسم .. بس اللي هيفرق بينهم مين بيناديها وايه الـ `parameters` اللي بتروح لها.

Polymorphism: statements that use more than one

abstraction, with the language choosing the right one while
the program is running.

Constructor (inheritance)

```
template < class btElementType >
BST < btElementType >
:: BST() : BinaryTree < btElementType >()
{
}
```

[79]

يعنى مثلا ده ال constructor بناء ال derived BST .. فاضى!!

عشان هو مفهوش حاجة زيادة عن ال constructor بناء ال binary tree فهيعمل نفس اللي في ال constructor بناء ال binary tree وخلاص

لكن لما احب أقول ان في حاجة هتبقى مختلفة بين دى ودى لازم هحطها في ال class زى كده

BST Inherited Class

```
template < class btElementType >
class BST : public BinaryTree < btElementType > {
public:
    BST();
    void insert(const btElementType & d);
    BinaryTree < btElementType > *
    retrieve(const btElementType & d);
};
```

[75]

هنا هنلاقى تعريف للinsert وللretrieve ولازم احط constructor بردو .. بس طالما حطيت دول هنا يبقى
انا بقوله هيبقوا مختلفين عن بتوع ال .base

ابقوا بوصوا في باقى السلايدز بردو اللي الدكتور مجابهاش.

فليكن همك السعي، لا الوصول.

والسير على طريق الوصول .. وصول.