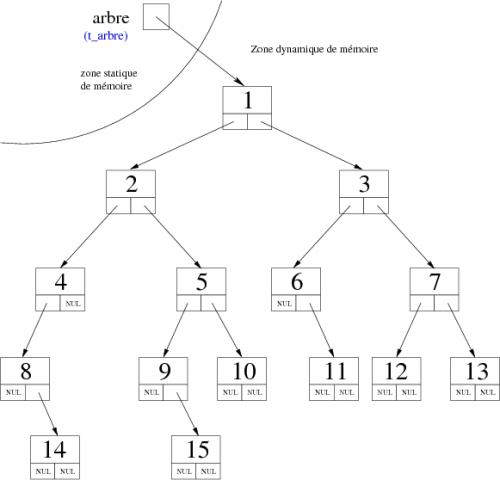
# TP Arbre de compétence



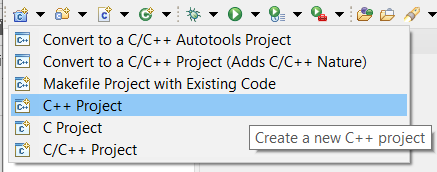
# Création du projet Eclipse CDT et mise en place de l’environnement

Dézipper la plateforme Eclipse CDT du server chez vous.

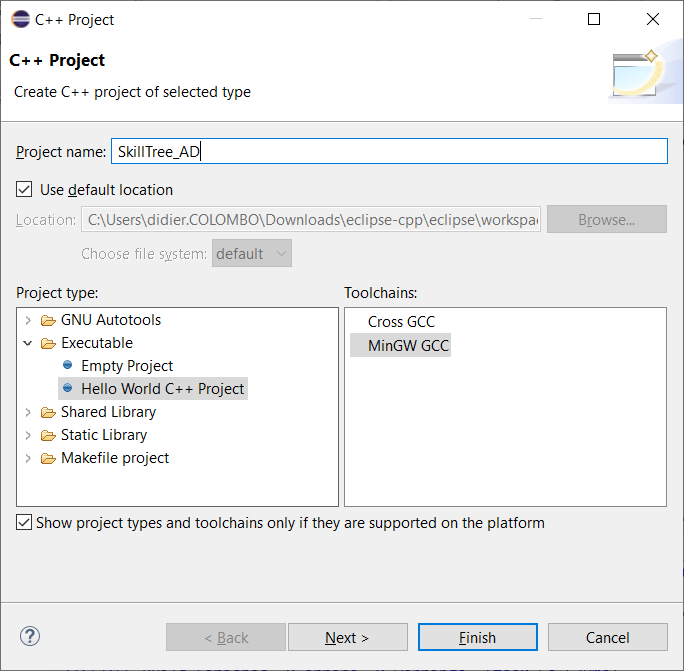
Préciser le chemin de votre compilateur.

# Création du projet

Depuis le bouton en haut à gauche  choisir C++ Project.



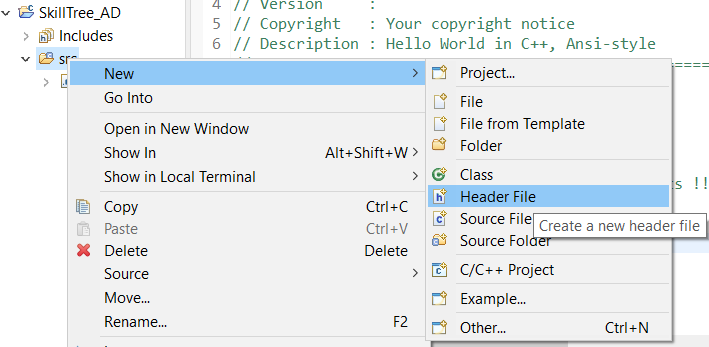
Nommer le projet SkillTree\_VosInitiales et choisir les options suivantes :



Puis Finish.

# Création et définition du header

1. Création du fichier header



Nommer le SkillTree\_VosInitiales.hpp et ajouter le code suivant :

**#ifndef** SkillTree\_ VosInitiales \_

**#define** SkillTree\_ VosInitiales \_

**#include** <iostream>

**using** **namespace** std;

//Code ici

**#endif** /\* SkillTree\_VosInitiales\_HPP\_ \*/

1. Méthode publique et attribut privé
2. Ajouter la définition du type Arbre en utilisant les attributs suivant (en private) :

* Un attribut skillName de type string.
* Un attribut skillPoint de type int.
* Un attribut rightChild.
* Un attribut leftChild.

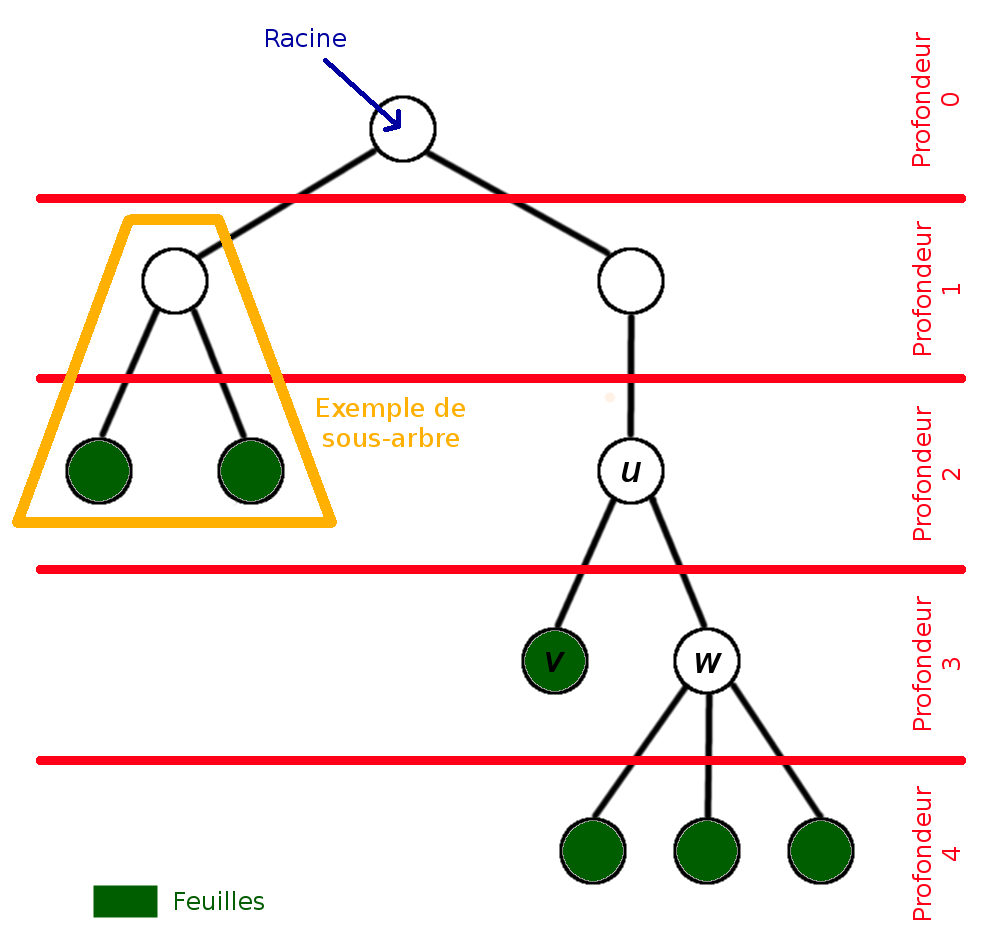
1. Ecrire la signature du constructeur par défaut et du destructeur.
2. Ecrire les signatures des méthodes publiques suivantes :

* setName (pour changer le nom)
* setLeftChild (pour changer le sous arbre de gauche)
* setRightChild (pour changer le sous arbre droit)
* IevelUpSkill (qui fait plus 1 sur l’attribut skillPoint)
* reset qui remet à 0 les points de compétence de l’arbre et de ses enfants.

1. Ecrire la signature des méthodes getDepth et display :

Display affiche l’arbre en console avec des cout).

GetDepth calcul la profondeur de l’arbre. (voir image ci-dessous)



# Implémentation des méthodes en C++

Complétez les méthodes dans le SkillTree\_AD.cpp (dans l’ordre que vous préférez) en partant de la base suivante :

**#include** <iostream>

**#include** "SkillTree.hpp"

**using** **namespace** std;

**inline** **SkillTree::SkillTree**(//Add parameters if needed) {

//Add method body

}

**inline** **SkillTree::~SkillTree**(//Add parameters if needed) {

//Add method body

}

**inline** **void** **SkillTree::setLeftSkillTree**(//Add parameters if needed) {

//Add method body

}

**inline** **void** **SkillTree::setRightSkillTree**(//Add parameters if needed) {

//Add method body

}

**inline** **void** **SkillTree::addSkillPoint**(//Add parameters if needed) {

//Add method body

}

**inline** **void** **SkillTree::setName**(//Add parameters if needed) {

//Add method body

}

**inline** **void** **SkillTree::resetSkillPoint**(//Add parameters if needed) {

//Add method body

}

**inline** **int** **SkillTree::getDepth**(//Add parameters if needed) {

//Add method body

}

**inline** **void** **SkillTree::display**(//Add parameters if needed) {

//Add method body

}

# Tester les différentes fonctions dans un fichier TestArbre

**#include** <iostream>

**#include** "SkillTree.cpp"

**using** **namespace** std;

**int** **main**(**void**)

{

SkillTree A;

SkillTree A1;

SkillTree A2;

A.setLeftSkillTree(&A1);

A.setRightSkillTree(&A2);

...

A.display();

cout << A.getDepth() << **endl**;

# Compteur

Ajouter deux attributs (nodeCounter et treeCounter) à la classe arbre pour compter le nombre de nœud créé au total et le nombre d’arbre créé au total.

# Optionnel

Ajouter un pointeur vers l’arbre parent et adapté le code existant.

# Bonus

Rendre cet arbre générique (pour que cet arbre puisse avoir autre chose que des entiers).

# Questions subsidiaires :

De quel jeu sont tirés les noms des compétences de l’exemple donné l’exemple d’après vous ?