//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

strtok with char example

char fname**[**32**],** lname**[**32**],** city**[**32**];**

char zip**[**32**],** zip**[**32**],** country **[**32**];**

strcpy**(**fname**,** strtok**(**str**,** "|"**));**

//notice subsequent calls are NULL meaning

//tokenize the SAME string further

strcpy**(**lname**,** strtok**(NULL,** "|"**));**

strcpy**(**city**,** strtok**(NULL,** "|"**));**

strcpy**(**zip**,** strtok**(NULL,** "|"**));**

strcpy**(**country**,** strtok**(NULL,** "|"**));**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

GET LENGTH OF CHARACTER ARRAY

char input**[**80**];**

cout **<<** strlen**(**input**)** **<<** endl**;** //outputs 80

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Binary Monkey Solution

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include <iostream>

#include <iomanip>

#include <fstream>

#include <cstdlib>

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

struct binary**{**

int age**;**

binary**\*** younger**;**

binary**\*** older**;**

**};**

int leftCount**(**binary tree**,** int count**)**

**{**

**if(**tree**.**younger **==** **NULL)**

**return** count**;**

**else**

**{**

count**++;**

**return** leftCount**(\***tree**.**younger**,** count**);**

**}**

**}**

void addTree**(**binary **&**tree**,** int age**){**

binary**\*** nMonk **=** **new** binary**;**

nMonk**->**age **=** age**;**

nMonk**->**younger **=** **NULL;**

nMonk**->**older **=** **NULL;**

**if(**nMonk**->**age **<** tree**.**age**){**

**if(**tree**.**younger **==** **NULL){**

tree**.**younger **=** nMonk**;**

**}**

**else**

addTree**(\***tree**.**younger**,** age**);**

**}**

**else** **if(**nMonk**->**age **>=** tree**.**age**){**

**if(**tree**.**older **==** **NULL)**

tree**.**older **=** nMonk**;**

**else**

addTree**(\***tree**.**older**,** age**);**

**}**

**}**

int main**()**

**{**

int count **=** 0**;**

int monkey**;**

cin **>>** monkey**;**

binary tree**;**

tree**.**age **=** monkey**;**

tree**.**younger **=** **NULL;**

tree**.**older **=** **NULL;**

**while((**cin **>>** monkey**)** **!=** **false)**

**{**

addTree**(**tree**,** monkey**);**

**}**

cout **<<** leftCount**(**tree**,** count**);**

**return** 0**;**

**}**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// End Solution

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Eclipse Shortcuts

Ctl + F11 – Run Regular

F11 – DEBUG the current perspective

* F6 – Step through code line by line
* F8 – Resume program (until a breakpoint is encountered)

When debugging, make sure to stop the current instance if a new debug instance is needed.

Remember to change the intellisense in Eclipse to accept all letters and symbols.

* Eclipse > Preferences > Java > Editor > Content Assist
* Auto Activation > Auto activation triggers for Java
* Enter all the characters you want to trigger autocomplete, such as the following:

**abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ.\_**

# Sorting Algorithm Template

|  |  |
| --- | --- |
| *default (1)* | template <class RandomAccessIterator>  void sort (RandomAccessIterator first, RandomAccessIterator last); |
| *custom (2)* | template <class RandomAccessIterator, class Compare>  void sort (RandomAccessIterator first, RandomAccessIterator last, Compare comp); |

// sort algorithm example

#include <iostream> // std::cout

#include <algorithm> // std::sort

#include <vector> // std::vector

bool myfunction (int i,int j) { return (i<j); }

struct myclass {

bool operator() (int i,int j) { return (i<j);} //binary function to compare

} myobject;

int main () {

int myints[] = {32,71,12,45,26,80,53,33};

std::vector<int> myvector (myints, myints+8); // 32 71 12 45 26 80 53 33

// using DEFAULT comparison (operator <):

std::sort (myvector.begin(), myvector.begin()+4); //(**12 32 45 71**)26 80 53 33

// using function as comp

std::sort (myvector.begin()+4, myvector.end(), myfunction);//12 32 45 71(**26 33 53 80**)

// using object as comp

std::sort (myvector.begin(), myvector.end(), myobject); //(12 26 32 33 45 53 71 80)

// print out content:

std::cout << "myvector contains:";

for (std::vector<int>::iterator it=myvector.begin(); it!=myvector.end(); ++it)

std::cout << ' ' << \*it;

std::cout << '\n';

return 0;

}