Standings

Programmer Manual
Standings

1. Problem Description

The Standings class consists of an stl::list and functions to add to, delete from, and edit teams in the list, as well as simulate games between teams in the list. The specifications for the teams added to this list are detailed in the programmer manual for Teams.

2. Class Standings

Private data members:

list<Team> li	the list which holds all of the teams
bool isEmpty	determines if Standings is empty or not

Private member functions:

initDivision	parses the division number of a team in order to put the team in the correct division and conference
isConfGame	determines whether a game between two teams is a conference game
isDivGame	determines whether a game between two teams is a division game
updateWinPercent	calculates the win percent of a team
standingsSort	sorts the list by division number, then by win percentage, and finally alphabetically by team name
deleteList	deletes every element in the list
validatedInput	checks whether the user's input is valid or not

Public member functions:

Standings	constructor for a Standings object
init	initializes the list by parsing the data from the input file
print	prints out the list in order
addTeam	adds a team to the list
deleteTeam	removes a team from the list
playGame	simulates a game between two teams
output	writes the new standings into a new file which can be read back in
editTeam	allows the user to edit a specific team in the list

3. High Level Program Solution

Standings

sets isEmpty to true

init
    if the list is not empty, delete the list
    while the input file is not empty:
        read the data from the input file and parse it into a Team struct
        calculate that Team's win percentage
        push the created Team onto the list
    sort the created list

print
    if the list is empty, tell the user to input data into the list
    set up formatting for printing, including whether or not to print the conference or the division
    print all of the formatted data for each Team in the list

addTeam
    create a new empty Team struct
    check if the user entered no name or a name already in the list
    ask the user if they would like to input the information about the Team
    if yes:
        get the information for each data member of the Team from the user
    if no:
        initialize all of the Team data to 0
    push the new Team onto the list
    sort the list

deleteTeam
    check if the user entered no name
    search the list for the name of the Team to delete
    if it is found:
        remove the Team
    if it is not found:
        inform the user it was not in the list

playGame
    check if either team was not found in the list
    get the score of the game from the user
    determine a winner if there is one, and update the two teams stats appropriately
    sort the list

output
    open the output file
    writes the information for each Team into the file according to how it will be read back into the program
    close the output file

editTeam
    check if the user entered no name
    search for the team to edit
    if the team is not in the list inform the user
    if the team is in the list, get information for each data member of the Team from the user
    sort the list

initDivision
    check the division number for a given team
    copy the correct division and conference name into the Team according to the division number

isConfGame
    check if the difference between two Team's division number is less than 4
    if yes:
        return true, the game is a conference game
    if no:
        return false, the game is not a conference game

isDivGame
    check if the division number between two Teams is the same
    if yes:
        return true, the game is a division game
    if no:
        return false, the game is not a division game

updateWinPercent
    calculate the win percentage of a Team
    the win percentage is the number of wins divided by the total number of games
    if the total number of games is 0, set the win percentage to 0

standingsSort
    call the stl:list.sort function to sort the list based on the overloaded operator < in the Team struct

deleteList
    iterate through the list, deleting each element using stl:list.erase

validatedInput
    if the user's input is outside of the bounds, or causes cin to fail, turn on cin, clear the buffer, and reprompt for the input