COMS W4115 Programming Languages and Translators

(Professor: Stephen A. Edwards)

Fantasy Football Stat Tracker Compiler: Language Reference Manual

Michael Lam
Email: michael.lam@lmco.com
Phone: 215-815-5629
Due Date: 10/18/2007

Table of Contents

1 Introduction

My inspiration for designing a Fantasy Football Stat Tracker compiler transpired while participating in a fantasy football league with some friends. While the cost for participating in the fantasy football league is free of charge, most of the online fantasy football services charge a bundle for using their online stat tracker. The FFSTC will provide features for gathering player statistics, sorting based on category type, filtering by position type, stats comparison, and search by player name. In this version of the FFSTC, statistics will only be available for the 2006 Fantasy Football Season.

2 Lexical Conventions

2.1 Tokens

There are six types of tokens in this language: identifiers, keywords, constants, strings, expression operators, and separators. A whitespace must be used to separate tokens.

2.2 Comments

FFSTC supports single and multi-line comments. Single line comments are denoted using the notations @@. Multi line comments begin with @% and ends with %@

Example:

@@ This is a valid comment

@%This is also a
valid comment%@

2.3 Identifiers

An identifier is a sequence of letters and digits with the first character beginning with an alphabet. Identifiers are case sensitive; the "_" is allowable

2.4 Keywords

The following are identifiers reserved as keywords and may not be used otherwise.

while	endwhile	QB	TeamName
if	else	RB	bool
endif	bool	WR	int
FALSE	TRUE	TE	
for	endfor	KKR	
continue	break	DEF	

The following are identifiers reserved for designating team names.

BUF	BAL	HOU	DEN
MIA	CIN	IND	KAN
NE	CLE	JAC	OAK
NYJ	PIT	TEN	SD
DAL	CHI	ATL	ARI
NYG	DET	CAR	SF
PHI	GB	NO	SEA
WAS	MIN	TB	STL

2.5 Constants

Three types of constants are allowed in FFSTC: integer, double, and string. All constants are capitalized.

Example:

const MAXYARDS = 100;

2.6 Operator

The following operators can be used in FFSTC:

=	*=
+=	-=
/=	==
>	<
>=	<=
&&	||
!=	!
+	-
*	/

3 Data Types

The following table shows the data types in FFSTC:

QB	int
RB	bool
KKR	
TE	
DEF	
TeamName	

4 Declarations

Declaring variables in FFSTC can take the following form:

```
QB myQuarterBack;
RB myRunningBack;

QB myQBRankings[10];
RB myRBRankings[10];
```

4.1 QB Type

The QB type will be associated with the statistical categories passYards, passTD, and passINT.

4.2 RB Type

The RB type will be associated with the statistical categories rushYards, rushTD.

4.3 WR Type

The WR type will be associated with the statistical categories rcvYards, rcvTD.

4.4 TE Type

The RE type will be associated with the statistical categories rcvYards, rcvTD.

4.5 KKR Type

The KKR type will be associated with the statistical categories FGMade, FGMiss.

4.6 DEF Type

The DEF type will be associated with the statistical categories defSACK, defINT.

5 Statements

All statements in FFSTC end in a semi-colon.

5.1 Conditional Statements

The following syntax denotes the *if* statement:

```
if (expression)
        statement
endif
```

5.2 Looping Statements

The following syntax denotes the while statement:

```
while (expression)
        statement
endwhile
```

5.3 Print Statements

The printing for scoring statistics are handled through the build-in function *FFLPrint(x,y).* Users may only use print statements to echo messages.

For Example:

printf(“Hello World.”);

6 Scope

In FFSTC, all variables declared will be represented as global variables. All declarations should be declared in the beginning of the file.

7 Built-in Functions

All Built-in functions in FFSTC begin with FFL.

FFLGetDatabase();	Required in the beginning of program. Without this declaration, FFSTC will not be able to obtain records.
bool FFLSearch(*Para1*,*Para2*,*Para3*,*Para4*);	Search for the statistics for a specified position type by name and team and stores the value for position type. Returns true or false. *Para1* ➔ *Position Type* *Para2* ➔ *Lastname* *Para3* ➔ *Firstname* *Para4* ➔ *Team Identifier* For Example: QB myQB; bool isFound (false); isFound = FFLSearch(myQB, "Favre", "Bret", GB);
int FFLRetrieve(*Para1,Para2,Para3,Para4);*	Retrieve the statistics for a specified position type ranked highest by the statistic category type. The record will be stored into an array and the record with the highest ranking will be returned. *Para1* ➔ *Position Type* *Para2* ➔ *Position Identifier* *Para2* ➔ *statistic category* *Para3* ➔ *number of records* *Para4* ➔ *Team Identifier* Example: QB myQBRankings[11]; FFLRetrieve(myQBRankings, QB, passYards, 10);
FFLPrint(*Para1, Para2*);	Prints the statistics for a specified position type by the entire 2006 season or weekly. *Para1* ➔ *Position Type* *Para2* ➔ *0 denotes entire season, 1-17 denotes weeks 1 through 17.* Example: @@ Prints my quarterback statistics for the entire 2006 season FFLPrint(MyQB, 0); @@ Prints my quarterback statistics for week 6 FFLPrint(MyQB, 6); Options: Prints the statistics for a specified TeamName type by weekly only. Example:

	@@ Prints team scoring totals for week 11 FFLPrint(myCustomTeam, 11);
TeamName Type = FFLFormulateTeam (*Para1*, *Para2*, *Para3*, *Para4*, *Para5*, *Para6*, *Para7*);	In a fantasy football league, a team consists of seven position types: QB, RB, WR, WR, TE, KKR, and DEF. This build-in function will create a team using specified position types and assign it to TeamName Type. *Para1* ➔ *QB Type* *Para2* ➔ *RB Type* *Para3* ➔ *WR Type 1* *Para4* ➔ *WR Type 2* *Para5* ➔ *TE Type* *Para6* ➔ *KKR Type* *Para7* ➔ *DEF Type* Example: TeamName myCustomTeam; myCustomTeam = FFLFormulateTeam(myQB, myRB, myWR1,myWR2, myTE, myKKR, myDEF);

8 Functions

User defined functions are not allowed in FFSTC.

9 Sample Program

```
@@ #FFSTC indicates the beginning of the program
#FFSTC

@@ Required declaration to obtain scoring statistics for all position types
FFLGetDatabase();

QB myQB;
RB myRBRankings[11];
WR myWRRankings[11];
RB myRB;
WR myWR1;
WR myWR2;
TE myTE;
KKR myKicker;
DEF myDefense;
bool isFound(false);
int RBHigh, WRHigh;

@@ Search for the top ten rankings amongst position type based on statistical category
RBHigh = FFLRetrieve(myRBRankings, RB, rushYards, 10);
WRHigh = FFLRetrieve(myWRRankings, WR, rcvYards, 10);

@@ Print the top ten rankings
for (int i=0;i<10;i++)
        FFLPrint(myQBRankings[i], 0);
endfor

@@ Assign my players to the highest ranked position player
myRB = myRBRankings[RBHigh];
myWR = myWRRankings[WRHigh];

@@ Print the seasonal stats for my players.
FFLPrint(myRB, 0);
FFLPrint(myWR, 0);

@@ Search for my starting quarterback
isFound = FFLSearch(myQB, “McNabb”, “Donovan”, PHI);

if (isFound)
        FFLPrint(myQB, 0);
endif

@@ Denotes the end of the program
#ENDFFSTC
```