## Passing data objects from lexer to parser (using Flex and Bison)

Nikhil Hegde, Department of CSE, IIT Dharwad

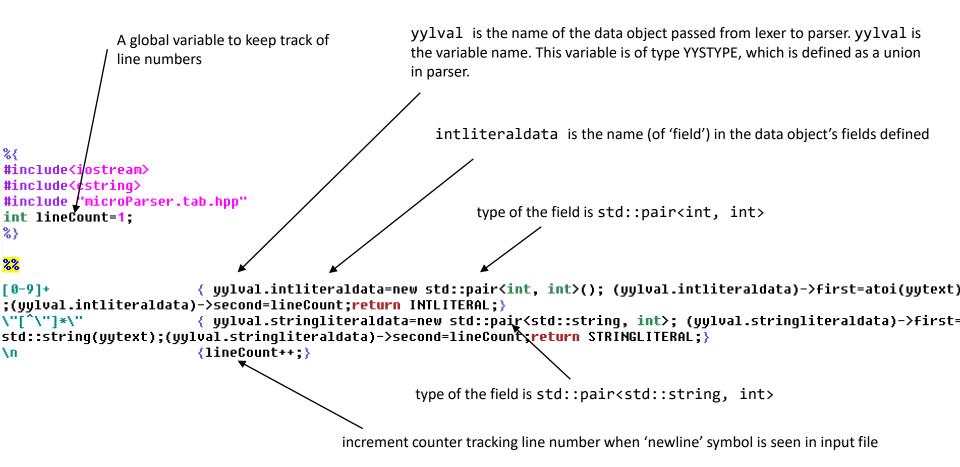
Nikhil Hegde, IIT Dharwad

• Goal: recognize an INTLITERAL and STRINGLITERAL in the program text, print the line number and the value of the INTLITERAL and STRINGLITERAL as a semantic action in the parser.

```
PROGRAM test
STRINGLITERAL val:"Hello" at line:6
                                                     2 BEGIN
                                                           INT a,b,c,x,y,z,h,j,k;
INTLITERAL val:1 at line:7
                                                        FUNCTION INT main()
                                                           BEGIN
INTLITERAL val:2 at line:7
                                                           STRING str:="Hello":
                                                           a := 1; b:=2;
                                                           IF(a = b)
INTLITERAL val:0 at line:9
                                                              i := 0;
                                                              WHILE (j <= 10)
INTLITERAL val:10 at line:10
                                                    11
                                                    12
                                                                    i := i+1:
                                                    13
                                                              ENDWHILE
INTLITERAL val:1 at line:12
                                                    14
                                                           ENDIF
                                                    15
                                                           RETURN a+b:
Accepted
                                                    16
                                                           END
                                                    17 END
```

Nikhil Hegde, IIT Dharwad

## scanner.1 file



Nikhil Hegde, IIT Dharwad

## microParser.ypp file

```
iostream is needed for std::pair
Whatever you put in between %{ and %} is copied to microParser.tab.cpp file but not
microParser.tab.hpp file.
                                              This is the type of the data object passed from lexer to scanner. This object is a union and
                                              has fields intliteraldata and stringliteraldata. This union cannot contain only
                                              basic data types such as int, float, char, and pointers.
                                                     As the details of this type is required by the scanner and as this type contains
#include<iostream>
                                                     std::string and std::pair, scanner.l needs to include corresponding
int yylex();
void yyerror(char const* errmsq);
                                                     headers. This inclusion must be done before #include"microParser.tab.hpp"
                                                     because the .hpp file does not include these headers as mentioned above. Also
                                                     note that all the fields inside %union are pointers types.
%union{
std::pair<int, int>* intliteraldata;
std::pair<std::strinq, int>* strinqliteraldata;
                                                   Note two ways of associating fields of the data object with tokens: 1) %token
                                                  <intliteral> INTLITERAL. In this case you don't have to separately define %token
%token <intliteraldata> INTLITERAL
                                                  INTLITERAL 2) %token STRINGLITERAL followed by %type<stringliteraldata>
%token STRINGLITERAL
                                                  STRINGLITERAL. In this case, we mean explicitly the type of stringliteral is the type
%type <stringliteraldata> STRINGLITERAL
                                                  of the semantic record of STRINGLITERAL token.
program: PROGRAM id BEGIN pgm body END {printf("Accepted\n");return 0;};
   ..other CFG rules go here
str: STRINGLITERAL {printf("STRINGLITERAL val:%s at line:%d\n",(($1)->first).c str(), ($1)->second);delete $1;};
primary: LPAREN expr RPAREN {}
          id {}
          INTLITERAL {printf("INTLITERAL val:%d at line:%d\n",($1)->first, ($1)->second);delete $1;}
          FLOATLITERAL ();
                    • ($1) is the reference to data object of STRINGLITERAL. Whenever the category of the token matched in scanner is a
                       STRINGLITERAL, the scanner creates an object of type as that of stringliteraldata, initializes it with appropriate values and
                       sends it to parser. The return statement in scanner only returns the token category. The token value is set by the scanner
                       using yylval object. The parser refers to this object using $1 in this case in the semantic action using ($1).
```

- first and second are the names by which you access the fields of a std::pair object

• c str() is needed to convert std::string to C-style strings (char \*). If you use cout to print you don't need this.

delete the object created if no longer required.