\*\* flex expr.l

\*\*\* bison expr.y

\*\*\*\* gcc -o b expr.tab.c expr.c

Logica-> raux && logica

|raux || logica

|raux

raux-> !raux

|rlaux

rlaux → (logica)

|relacional

relacional-> ari >= ari

| ari <= ari

| ari > ari

| ari < ari

| ari == ari

| ari != ari

a1->exprAri

expAri → expr + termo

| expr – termo

| termo

termo → termo \* fator

|termo / fator

| fator

fator → num

| ( expr )

| - fator

\*1

e && e && e

(e||e)&&e

!(\_ && \_)

Aula 16 - out

-Fazer arvore ou hash, para servir como tabela de símbolos.

-Lista para as variáveis da mesma linha (int a,b,c,d;)

L->LD

| D

D->TV;{insTabSimb(V.listaId,T.tipo);Liberar(V.listaId);}

T->int {T.tipo = int;}

|String{T.tipo = String;}

V->V,id{V.listaId = insere(V.listaId, id.lexval);} (\*\*yylval)

|id{V.listaId = crialista(id.lexval);}

\*\*\*yylval.nomeId = yytext; → strcpy(yylval.nomeId,yytext);

Struct Atributo

{

char nomeId[20];

int tipo; // 0 = int, 1 = String

Lista = \*listaId;

}

-No arquivo .y, YYSTYPE struct Atributo