GOTO

x^2 + y^2 = N

for (x=0; x<N; x++){

for (y=0; y<N; y++){

if (x \* x + y\*y == N){

cout << “Found”;

–-----> GOTO

}

}

}

(Same for both C and Pascal)

**4 Places to worry about:**

**Label Declaration** (Pascal Specific)

<declaration>

<label\_declaration>

<label\_decl>=>label<namelist>;

label names must be unique.

[0] → {Label\_name, TK\_A\_LABEL, address, seen (initialize to false), [array\_of\_goto\_list], scope\_info}

We'll come back to scope later.

*Goto Label and Label may occur in any order*

**Goto Label:**

This happens in a statement:

<statement> → <goto statement>

<goto\_statement> → goto <label>

//C only: if (token == TK\_ID) { put it into symbol table }

gen1(op\_jump);

hole ← ip;

gen4(0);

append hole to goto\_list

store some information about where we are right now. So that we can check for illegal goto.

**Label:**

<statement> → <labelling>

<labelling> → TK\_A\_LABEL

if (seen) error();

else{

seen = true;

address ← IP;

}

**At the end:**

If label is not found then corresponding goto(s) are illegal and error should be reported.

<pascal\_program>

<declaration>

<begin\_statement>

FIX GOTOS

for all labels in symbolTable:

if (!seen && goto\_list != null){

error();

}

for all entries in the goto list:

patch the hole with the label's address.

Label record in symbol table:

TK\_A\_LABEL

Seen

Address

GOTO List

Scope Info

→ Next Label

–----------- –-------------- –------------- –----------- –---------- –--------- –--------

=========================================================================================

In case of error. Say where the error occurred.

Everytime I see begin (Or a { in C) I enter a block.

Jump to the same or lower block is allowed.

Each block gets and ID which is of type string.

Length of string = level, starting with 0. Sub blocks get 00. When a block is closed the upper level gets incremented. A subsequent block on the same level becomes 001.

The following code is illegal but won't cause error in our code

for i := 1 to 10 do

goto l

for i := 1 to 10 do

l:

For this we have to introduce the concept of implicit block. Everytime we encounter if, for, while, we automatically create a block if begin/end statement does not exist.

While <cond> do {open\_block} <statement> {close\_block}

<begin\_statement> →

begin {open\_block} <statement> {close\_block} end

{IMPORTANT}

Pascal allows numbers as names of labels.

This is not needed for the purposes of this course.