Compilers

3rd year
Spring term

Mick O'Donnell: michael.odonnell@uam.es

Based on the book:

Compiladores e interpretes: teoría y práctica (Estrella Pulido, Marina de la Cruz, Alfonso Ortega, Manuel Alfonseca)

11 Error Detection

11.1 Introduction

Detection of Errors

- To be useful, a compiler should detect all errors in the source code and report them to the user.
- These errors culd be:
  - Lexical errors: e.g., badly formed identifiers or constants, symbols which are not part of the language, badly formed comments, etc.
  - Syntactic errors: chains of syntactic units that do not conform to the syntax of the source language.
  - Semantic errors: e.g., operations conducted on incompatible types, undeclared variables, double declaration of variable, reference before assignment, etc.
  - Run-time errors: errors detectable solely at run time, pointers with null value or whose value is outside allowed limits, or indexing of vectors with unsuitable indices, etc.

3

Error Detection

Introduction

Detection of Errors

- To be useful, a compiler should detect all errors in the source code and report them to the user.
- These errors culd be:

Compilation Errors	• Lexical errors: e.g., badly formed identifiers or constants, symbols which are not part of the language, badly formed comments, etc. • Syntactic errors: chains of syntactic units that do not conform to the syntax of the source language. • Semantic errors: e.g., operations conducted on incompatible types, undeclared variables, double declaration of variable, reference before assignment, etc.
Execution Errors	• Run-time errors: errors detectable solely at run time, pointers with null value or whose value is outside allowed limits, or indexing of vectors with unsuitable indices, etc.

4

Error Detection

Introduction

A good compiler…

- Reports ALL errors
- Does not falsely report errors.
- Does not repeatly report the same error.

5

Error Detection

Introduction

Detection of Errors

- How many errors will the **c** compiler report?

```
void main () {
     int i,j,k;
     i=0; /* Asigno 0 a i //
     j=;
     =k;
}
```

6

Detection of Errors

- How many errors will the **c** compiler report?

```
void main () {
     int i,j,k;
     i=0; /* Asigno 0 a i //
     j=;
     =k;
}
```

- Answer: 2

  1) Comment on line 3 not terminated

  2) Function main has no closing bracket

False Detection of Errors

- Sometimes, due to one error, the compiler loses track of context and may end up seeing the rest of the program as erroneous

Error Detection

Introduction

False Detection of Errors

- Take the following program:

```
void main () {          // line 1
int i, j;               // line 2
i=1;                    // line 3
while (i) {             // line 4
  int j;                // line 5
   …
  }                     // line 10
j=2;                    // line 11
if (i<j) j++; // line 12
…
}                       // line 65
```

- If we delete the { on line 4, 'int j' is taken as the body of the while loop.
- This declaration of j is reported as an error (already declared in line 2)
- The close bracket on line 10 is seen as closing the main() function.
- All lines 11-65, meant to be part of 'main()', are reported to be "instruction located outside a function"
- Any real errors in lines 11-65 would NOT be reported!

9

Error Detection

Introduction

Recovery from Errors

- In such cases, an intelligent compiler might recognise that a parenthesis was missing, and look for a place it could close the block, and then resume processing as normal.
- Possibly indentation might give a clue.
- Or trying each location between tokens to find the one which produces the fewest errors

10

Over-reporting errors

- Assume we forget to declare a variable
- If the variable is referenced 20 times, should we receive 20 error messages? Or just 1?
- Ideally, when a reference to an undeclared variable is found:
  - The symbol table is checked to see if any previous such error was detected for this variable.
  - If yes, no message is given,
  - Else:
    - An error message is printed,
    - The symbol table is modified to indicate this.

Over-reporting errors

- Alternatively, the symbol table can be used to store each occurence undeclared reference to the variable.
- At the end of the scope, a single message is printed:
  - *Undeclared reference to variable j on lines 21, 22 and 23*

Automatic Correction of Errors

- Ideally, a compiler could identify the presence of errors, and automatically correct the code to remove the error
- An executable could then be produced even with source-code errors
- Time in the code-compile-debug cycle could be saved
- BUT: if the fix provided by the compiler is wrong, it may be difficult to locate the bug at run-time.
- For this reason, commecial compilers do not generally offer this feature.

Automatic Correction of Errors

- If the compiler DOES do automatic correction, it must document its corrections (printed warnings).
- Otherwise the programmer is debugging a program that does not correspond to their source code!

Some types of Error Correction: orthographic correction

- When a symbol is encountered that is not declared, the compiler can use techniques similar to spell-checkers in word processors to guess the correct symbol.
- Easiest to start with the symbol table, find all identifiers visible in the current scope, and look for closest match.
- Include also list of reserved words.
- The system can try 1 character mutations:
  - All chars the same except one different (more likely if keys adjacent on keyboard)
  - All chars the same except one extra
  - All chars the same except for one less
  - All chars the same except 2 are swapped in position
- Prefer symbols which are declared but never assigned, or assigned but not referenced
- The parser can help: if a reserved word is expected at this point, prefer closest reserve-word before known identifiers.

Some types of Error Correction:

Orthographic correction: Space insertion

- Another case is where a space char is missing.
- Here we need to test insertion of a space at each point in the token
- The parser can help here also, informing which keywords are expected
- For instance, with "`voidmain`", and we expect amongst other tokens "void", then the correction is easy.

Some types of Error Correction: Syntactic Correction

- **Deletion of Token**
- When a sequence of tokens does not parse, one can try deleting each token in turn until the sequence parses.
- For instance, given

```
else x=0;
```

- …where there is no previous if statement,one could eliminate 'else' and produce a correct parse.

Some types of Error Correction: Syntactic Correction

- **Insertion of Token**
- When a sequence of tokens does not parse, one can try **inserting** a token in various places.
- For instance, given

```
if (i<j) {x=1; y=2; else x=0;
```

- One could try deleting "else"
- But more likely, the insertion of symbol } before the else.
- The compiler could be set such that, when an else does not match an earlier "if", automatically try "}" insertion.

Error Detection: SUMMARY

- How to detect all the compilation errors in a program.
- How to avoid false error reports.
- How to avoid reporting a cascade of spurious errors.
- How to avoid repeating the same error message.
- How to correct errors automatically.