Compiler Construction

Names, scopes and lifetimes

Names, Identifiers, Symbols

Terminology

Names, **identifiers** and **symbols** are three ways to denote the same thing

Goal

Mechanism to refer some entities: variables, type, function, namespace, control structure, etc

Identifiers

- usually alphanumeric and underscore, letter fist, without whitespaces
- ALGOL 60 and Fortran ignore withespaces
- Limitation on the length
  - 6 characters for the original Fortran (Fortran 90, 31)
  - ISO C: 31 characters
  - No limit for most of the other
- Case insensitive (Modula-2 and Ada)

cstats: Counting Symbols

```
g++ -E -P "$@" \
  | tr -cs '[:alnum:]_' '[\n*]' \
  | grep '^[[:alpha:]]' \
  | grep -v -E -w "$cxx_keywords" > $tmp.1
total=$(wc -lc < $tmp.1 \
        | awk '{print $1 " (" $2 " chars)"}')
sort $tmp.1 \
  | uniq -c \
  | sed 's/^   //;s/\t/ /' \
  | sort -rn >$tmp.2
unique=$(sed -s 's/.* //' $tmp.2 | wc -lc \
         | awk '{print $1 " (" $2 " chars)"}')
echo $total occurrences of $unique symbols.
sed 42q $tmp.2 \
  | pr --page-width=60 --column=3 --omit-header
rm -f $tmp.*
```

Lemon (as-of 2019-01-15)

```
15182 (78642 chars) occurrences of 1082 (8875 chars) symbols.
1868 gt                176 lineno              87 rule
 943 quot              155 lt                  87 h
 654 i                 149 cp                  82 np
 458 amp               148 s                   78 filename
 373 lemp              146 name                72 z
 347 rp                139 cfp                 71 fp
 306 n                 116 next                70 array
 297 psp               109 stp                 69 ht
 227 fprintf           108 p                   69 config
 199 sp                107 a                   62 errorcnt
 198 out               101 type                62 action
 187 j                  94 state               61 lem
 182 x                  91 symbol              60 d
 177 ap                 89 c                   56 data
```

GCC's C Parser

```
18958 (198353 chars) occurrences of 5835 (89396 chars) symbols.
2676 tree                89 new_type_flag     38 build_nt
1579 ttype               70 cpp_reader        36 itype
1123 yyvsp               69 build_tree_lis    36 build_x_binary
 909 yyval               67 parse             35 yychar
 358 ftype               65 y                 35 frob_opname
 247 t                   61 obstack           35 d
 206 gt_pointer_ope      58 GTY               34 e
 200 common              46 identifier        33 tree_code_type
 192 size_t              43 error             33 operator_name_
 175 code                40 cp_global_tree    33 C
 171 tree_code           39 yyn               32 got_scope
 123 FILE                39 s                 31 IDENTIFIER_NOD
  97 rtx                 39 lookups           30 tree_class_che
  95 type                38 TREE_LIST         30 global_trees
```

Tiger Compiler's Driver (as-of 1.70)

```
8544 (83423 chars) occurrences of 1320 (16098 chars) symbols.
 603 std              76 FILE             48 hash
 354 size_t           74 false_type       47 iterator_trait
 351 noexcept         73 declval          47 begin
 334 size_type        71 reverse_iterat   46 compare
 274 basic_string     64 difference_typ   46 char_traits
 268 type             62 pointer          42 integral_const
 202 constexpr        61 pair             41 allocator
 158 char_type        56 int_type         40 C
 153 forward          55 locale_t         39 first
 114 value            53 value_type       37 string
  96 decltype         53 move_iterator    37 replace
  94 true_type        52 move             37 basic_istream
  80 size             50 traits_type      36 exception_ptr
  77 base             48 length           35 wstring
```

Save Time and Space

One unique occurrence for each identifier

- **In C** a simple *const char**
- **In C++** an iterator in a std::set
  Sets have the important property that inserting a new element into a set does not invalidate iterators that point to existing elements.

Definition of scopes

Question answered by scopes

When are names created and destroyed?

Definition of scopes

Question answered by scopes

When are names created and destroyed?

Scope

The **textual** region in the **source file** in which the name is *available*

Why scopes?

Scoped-less world

- No scopes in assembly or in MFS
- Without scopes, names have global influence

Scoped world

- The programmer can focus on local influence
- Scopes are the first form of modularity

Static vs. Dynamic Scoping

Static scoping

The scope can be compute a compile time

Dynamic scoping

The scope depends on runtime conditions such as the function calls

Static scoping

- In most languages (Ada, C, Tiger, FORTRAN, Scheme, Perl (`my`), etc.).
- Enables static binding.
- Enables static typing.
- Enables strong typing (Ada, ALGOL 68, Tiger).
  - safer
  - faster
  - clearer

Dynamic Scoping

In most scripting/interpreted languages (Perl (`local`), Shell Script, TEX etc.) but also in Lisp

Dynamic Scoping in TeX

```
% \x, \y undefined.
{
  % \x, \y undefined.
  \def \x 1
  % \x defined, \y undefined.
  \ifnum \a < 42
    \def \y 51
  \fi
  % \x defined, \y may be defined.
}
% \x, \y undefined.
```

Prevents static typing

Scopes in C++

- Block scope
- Function parameter scopes
- Function scopes
- Namespace scopes
- Class scopes
- Templates scopes
- *modules?*

Scopes in Tiger

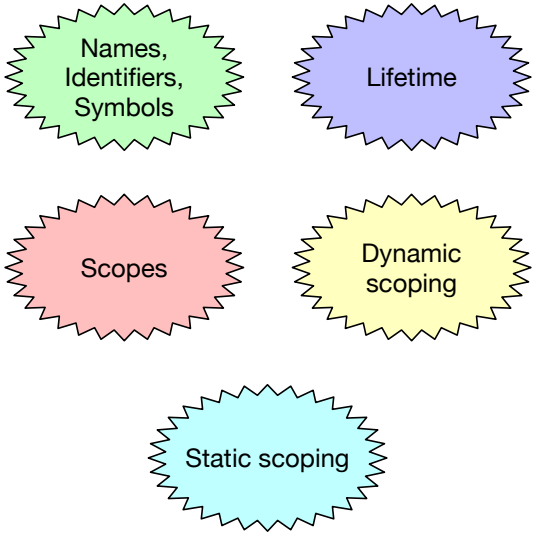
```
let
  var a := 3
  function f1() =
         ( a := a + 1 )
in
  let var a := 4
      function f2() =
             ( f1() )
  in
      f2()
  end
end
```

Lifetime

When are objects created and destroyed?

- Deferred to a later lecture
- Lifetime is a different matter, related to the execution (as opposed to visibility).
- Extent bound to lifetime of block tend to promote global variables (Pascal).

Summary


Names, Identifiers, Symbols
Lifetime
Scopes
Dynamic scoping
Static scoping