Today's Topics

Team Project

- Introduce this year's team project

S/SL

- S/SL, the Syntax/Semantic Language

CISC / CMPE 458 - Project 2019

Team Project

- Implement a compiler for a new language called Mini-Turing (MiniT) by modifying the existing PT Pascal compiler
- You will get your own copy of the PT compiler source code for your team, to change as you wish
- The project consists of four assignments, changing each of the four phases of the compiler
- Both your TAs and I will give you advice on how to change each phase - you can use the advice, or do it your own way
- Project is done in teams of 4
- You must choose your team and sign the contract by next Wednesday
- Be careful that all team members stay involved in all phases - course material is learned from doing the project!

CISC / CMPE 458 - Project 2019

The Mini-Turing Programming Language

- MiniT is a new modern, modular programming language based on a subset of Turing, a successor to Pascal designed for teaching
- We will implement MiniT by modifying and extending the phases of the PT Pascal compiler to handle MiniT instead
- MiniT changes PT in the following ways:
  - Turing syntax for programs, declarations and statements, removal of PT end-of-statement semicolons
  - Turing **%** comments replace PT { ... } and (* ... *) comments
  - Turing **modules** (anonymous classes) added
  - Turing **string** type and operations replace PT **char** type
  - Turing general **case** statement with **default** added
  - Turing **elsif** clause added to **if** statements
  - Turing general **loop** statement replaces PT **repeat** and **while**

CISC / CMPE 458 - Project 2019

- Pascal **begin** ... **end** –> Turing **end if**, **end loop**, …

```
program foo(output);

procedure bar(a:integer);
begin
   ...
end;
begin
   if a < b then
      begin
         ...
      end
   else
      begin
          while c < d do
             begin
                ...
             end;
      end;
end.
```

```
external output

procedure bar(a:integer)

   ...
end procedure

if a < b then

   ...

else

   loop
       exit when c >= d
      ...
   end loop
end if
```

CISC / CMPE 458 - Project 2019

- Turing **elsif** clause

```
if a < b then
   begin
      ...
   end
else
   if b < c then
      begin
         ...
      end
   else
      if c < d then
         begin
            ...
         end
      else
         begin
            ...
         end;
```

```
if a < b then

   ...

elsif b < c then

   ...

elsif c < d then

   ...

else

  ...
end if
```

CISC / CMPE 458 - Project 2019

- Pascal **case** –> Turing generalized **case** with default clause

```
case x of
  1:
      begin
         ...
      end;
  2:
      begin
         ...
      end;
  3:
      begin
         ...
      end;


end;
```

```
case x of
   label 1 :
         ...
   label 2 :
         ...
   label 3 :
         ...
   label :
         ...

end case
```

CISC / CMPE 458 - Project 2019

- Pascal **while … do, repeat ... until** –> Turing generalized loop

```
while x<10 do
begin
   ...
end;
```

```
loop
   exit when x>=10
   ...
end loop
```

```
repeat
   ...
until x=10;
```

```
loop
   ...
   exit when x=10
end loop
```

(general loop, not in Pascal)

```
loop
   ...
   exit when x=10
   ...
end loop
```

CISC / CMPE 458 - Project 2019

- Pascal declarations –> Turing-style declarations

```
const a = 1;                    const a := 1
      b = 2;                    const b := 2
      c = 3;                    const c := 3

type t = integer;               type t: integer

var  d: integer;                var d,e,f: integer
     e: integer;
     f: integer;
```

CISC / CMPE 458 - Project 2019

- Turing **modules**

```
external input,output
var x: integer

module M
    procedure privateProc (var a:integer)
        ...
    end procedure

    procedure * publicProc (var b:integer)
        ...
    end procedure
end module

publicProc(x)     % ok
privateProc(x)    % error!
```

CISC / CMPE 458 - Project 2019

- Turing-style **string** type

```
external input,output
var x,y: string
var n: integer

x := "foo"
y := x + "bar"      % y = "foobar"
n := ? y            % n = 6
x := y @ 3..5       % x = "oba"
```

- PT Pascal has only the single **char** type, which is removed from MiniT

CISC / CMPE 458 - Project 2019

- More Turing-like input/output statements

```
write('Hi there');          put("Hi there")
writeln;                    putln

read(x);                    get(x)
readln;                     getln
```

CISC / CMPE 458 - Project 2019

- Turing/C/Java-style short assignments

```
x := x + 1;          x ++
y := y - 1;          y --

x := x + y;          x += y
y := y - x;          y -= x
```

An Example MiniT Program

```
% Program to conjugate regular French verbs
external input,output

var infinitive,root: string

loop
   put ("Please give me a regular French 'er' verb "
        + "(end with 'arreter')") putln
   get (infinitive)
   putln

   put ("Thanks, here is the present conjugation") putln

   root := infinitive @ 1 .. (?infinitive - 2)
   put ("The root of this verb is '", root, "'") putln
   putln

   if infinitive @ (?infinitive - 1) .. ?infinitive = "er" then
     if (root @ 1..1 = "a") or (root @ 1..1 = "e")
           or (root @ 1..1 = "i") or (root @ 1..1 = "o")
           or (root @ 1..1 = "u") then
        put ("J'" + root + "e")
     else
        put ("Je " + root + "e")
     end if

     putln
```

An Example MiniT Program (cont'd)

```
      put ("Tu " + root + "es") putln
      put ("Il ou elle " + root + "e") putln

      if root @ ?root..?root = "g" then
         put ("Nous " + root + "eons")
      else
         put ("Nous " + root + "ons")
      end if

      putln

      put ("Vous " + root + "ez") putln
      put ("Ils ou elles " + root + "ent") putln

   elsif infinitive @ (?infinitive - 1) .. ?infinitive = "ir" then
      put ("I'm too tired to do an 'ir' verb") putln
      putln
   else
      put ("I don't like the looks of this verb") putln
      putln
   end if

   exit when infinitive = "arreter"

   getln

end loop
```

An Example MiniT Module

```
module queue
   var qstart,qend: 0..qsizem1
   var qlength: 0..qsize
   var contents: array [0..qsizem1] of itemtype
   procedure * enqueue (item: itemtype)
      qlength ++
      qend := (qend + 1) mod qsize
      contents [qend] := item
   end procedure
   procedure * dequeue (var item: itemtype)
      item := contents [qstart]
      qstart := (qstart + 1) mod qsize
      qlength --
   end procedure
   procedure * empty (var yes: boolean)
      if qlength = 0 then
         yes := true
      else
         yes := false
      end if
   end procedure
   % initialization
   qend := 0; qstart := 1; qlength := 0
end module
```

CISC / CMPE 458 - Project Schedule

Team Project

- Today - project specification handed out
  - First tutorial - tomorrow (Thursday) evening at 6:30 pm, BioSci 1103 (don't miss it!)
- Next week:
  - Team contracts due in class next Wednesday
  - First assignment handed out: Scanner/Screener phase

Next

- S/SL, the Syntax/Semantic Language