Today’s Topics

Team Project

- Introduce this year’s team project

S/SL

- S/SL, the Syntax/Semantic Language
**Team Project**

- Implement a compiler for a brand new programming language 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 2020

The Qust Programming Language

• This year's course project consists of implementing a compiler for the new modern programming language *Qust*

• *Qust* ("not *Quite Rust*") is an extended subset of Mozilla Corp's *Rust* programming language, used in the Firefox web browser, the Thunderbird email client, and the Bugzilla bug-tracking system

• We will implement *Qust* by modifying and extending the phases of the *PT Pascal* compiler to handle *Qust* instead
The Qust Programming Language

- Qust is a modular programming language with features similar to other modern languages like Swift, Ruby, and Python
  - However, from our point of view, it is a modification and extension to PT Pascal
- In particular, Qust changes PT to add:
  - Rust syntax for programs, statements, declarations, comments and operators
  - Rust modules, functions and scopes
  - Rust string data type and operations
  - Rust match statement with default alternative
  - Rust cascaded if statements
  - Rust increment / decrement assignments
  - Rust immutable variables
CISC / CMPE 458 - Project 2020

• Pascal `begin ... end` → Rust `{ ... }

```pascal
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.
```

```rust
mod main(output) { }

fn bar(a:int) { }

if a < b { }

} else {
  loop {
    break if c >= d;
  }
}
```
• Rust cascaded \texttt{else if}

\begin{verbatim}
if a < b then
  begin
    ...
  end
else
  if b < c then
    begin
      ...
    end
  else
    if c < d then
      begin
        ...
      end
    else
      begin
        ...
      end;
  else
    begin
      ...
    end;
end;
\end{verbatim}
Pascal `case`  →  Rust `match` with default alternative

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

```rust
match x {
  | 1 =>
    ...
  | 2 =>
    ...
  | 3 =>
    ...
  | _ =>
    ...
}
```
Pascal `while ... do, repeat ... until` → Rust `while, loop`

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

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

(general loop, not in Pascal)

Rust `while, loop`

```rust
while x<10 {
  ...
}

loop {
  ...
  break if x==10;
}

loop {
  ...
  break if x>=10;
  ...
}
CISC / CMPE 458 - Project 2020

- Pascal declarations → Turing-style declarations

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

var d: integer;
e: integer;
f: integer;

(type t = integer;

let mut d: int,
mut e: int,
mut f: int;

(let area: int = length * width;

let square = length * length;

let mut r = 0;

(immutable variables, not in Pascal)

(implicit variable type, not in Pascal)

(initialized variables, not in Pascal)
```
CISC / CMPE 458 - Project 2020

- Rust **modules**

```rust
mod main(input, output) {
    let mut x: int = 0;

    mod M {
        fn privateProc (mut a: int) {
            ...
        }

        pub fn publicProc (mut b: int) {
            ...
        }
    }

    publicProc(x);    // ok
    privateProc(x);   // error!
}
```
CISC / CMPE 458 - Project 2020

- Rust-like **string** type

```rust
mod main(input, output) {
    let mut x: str;
    x = "foo";
    let y: str = x + "bar"; // y == "foobar"
    let n = y; // n == 6
    x = y / 3:5; // x == "oba"
}
```

- PT Pascal has only the single **char** type, which is **removed** from Qust
CISC / CMPE 458 - Project 2020

• More Rust-like input/output statements

```rust
write('Hi there'); print("Hi there");
writeln(); println();
read(x); read(x);
readln(); readln();
```
Rust increment/decrement assignments

\[
\begin{align*}
x & := x + 1; & x & += 1; \\
y & := y - x; & y & -= x;
\end{align*}
\]
An Example Qust Program

// Program to conjugate regular French verbs
mod main (input,output) {
    let mut infinitive: str;
    loop {
        print ("Please give me a regular French 'er' verb "
            + "(end with 'arreter'))"); println();
        read (infinitive);
        println();
        print ("Thanks, here is the present conjugation"); println();
        let root: str = infinitive / 1 : (?infinitive - 2);
        print ("The root of this verb is ", root, ":" )); println();
        println();
        if infinitive / (?infinitive - 1) : ?infinitive == "er" {
            let root1: str = root / 1:1;
            if root1 == "a" || root1 == "e"
                || root1 == "i" || root1 == "o" || root1 == "u" {
                print ("J" + root + "e");
            } else {
                print ("Je " + root + "e");
            }
        }
        println();
    }
}
An Example Qust Program (cont’d)

```qust
print ("Tu " + root + "es"); println();
print ("Il ou elle " + root + "e"); println();
if root / ?root : ?root == "g" {
    print ("Nous " + root + "eons");
} else {
    print ("Nous " + root + "ons");
}
println();
print ("Vous " + root + "ez"); println();
print ("Ils ou elles " + root + "ent"); println();
} else if infinitive / (?infinitive - 1) : ?infinitive == "ir"{ 
    print ("I'm too tired to do an 'ir' verb"); println();
} else {
    print ("I don't like the looks of this verb"); println();
}
println();
break if infinitive == "arreter";
readln();
```
An Example Qust Module

mod queue {
    let mut qstart = 1,
    mut qend = 0,
    mut qlength = 0;

    let mut contents: [itemtype: qsize];

    pub fn enqueue (item: itemtype) {
        qlength += 1;
        qend = (qend + 1) % qsize;
        contents [qend] = item;
    }

    pub fn dequeue (mut item: itemtype) {
        item = contents [qstart];
        qstart = (qstart + 1) % qsize;
        qlength -= 1;
    }

    pub fn empty (mut yes: bool) {
        if qlength == 0 {
            yes = true;
        } else {
            yes = false;
        }
    }
}
CISC / CMPE 458 - Project Schedule

Team Project

• **Today** - project specification handed out
  • **First tutorial** - tomorrow (Thursday) evening at 6:30 pm, Botterell Hall B139 (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