Programming Language Design 2022
Scopes, functions and parameter passing

Hans Hüttel

21 February 2022

Where nothing else is mentioned, chapters and exercises are from the notes "Programming Language Design and Implementation", which are available on the Absalon page for the course.

The slides from the podcast can be found in the "Slides" folder, with LaTeXsources in the "Slide sources" folder. You can cut-and-paste from these to your assignment reports.

Many of the exercises has no single correct answer, so you are expected to prepare for discussing these in the exercise classes.

The video podcast and the slides

The video podcast is available from the main course page on Absalon. You can either watch the podcasts via the media library or download everything. The podcast is yours to keep. All files, podcasts and slides alike, have the same name as the topic of the session. Remember that the podcasts come in several parts.

If you prefer to watch the podcasts online, you can find them in the Media section and watch them there. If you want to download the podcast and watch everything off-line, visit the Video folder in the Files area. The corresponding slides can be found in the Slides folder along with their LaTeX sources in the Slide sources' folder. You can cut and paste from these to your assignment reports.

Many of the exercises have no single correct answer, so you are expected to prepare for discussing these in the exercise classes.

Monday 21 February – *Scopes, functions and parameter passing*

The podcast and slides will cover sections 6.1–6.4 of Chapter 6 of "Programming Language Design and Implementation".

Learning goals for the session *Scopes, functions and parameter passing*

The learning goals for this session are

Learning goals for scope rules

- To understand the definitions of dynamic and static scope rules
- To be able to apply the definitions of scope rules to reason about the behaviour of a program, given specific scope rules
- To be able to reason about the strengths and weaknesses of dynamic and static scope rules.
- To be able to describe how function calls are implemented for dynamic and static scope rules and how memory is allocated (deep/shallow binding and activation records).

Learning goals for parameter passing mechanisms

- To be able to explain the differences between different parameter passing mechanisms.
- To be able to apply the definition of a parameter passing mechanism to reason about the behaviour of a program.
- To be able to reason about the strengths and weaknesses of parameter passing mechanisms.

How you should prepare before we meet on 21 February

Before we meet on 21 February in Aud. 5 at HCØ, watch the podcast (all parts, please!) and read the text. You can do this in any order you like.

Also see if you can come up with answers to the following small problems.

1. Here are two fragments of Python code.

   ```
   xs = []
   def foo():
       xs.append(42)
   foo()
   ```

   and

   ```
       xs = []
   def foo():
       xs +=[42]
   foo()
   ```

   The first one will succeed. The second one gives a run-time error – which error? And why is there a difference here?

2. Here is an ALGOL60 procedure that uses call-by-name:

   ```
   procedure svop (a, b);
   integer a, b, temp;
   begin
     temp := a;
     a := b;
     b:= temp
   end;
   ```

   What is the intended effect of `svop`? Now suppose `x` is an integer-indexed array. What will happen if we call `svop(i, x[i])`? Any why? Is that what we wanted?

What happens on 21 February?

When we meet, we will discuss the small problems that I mentioned above along with a bunch of other small problems that I will make available to you on the day. The goal of this is to help you understand the text better and to prepare you for the exercise session on Tuesday 22 February.

I will introduce the small problems along the way and ask you to discuss them in small groups of two or three people. Every now and then I will discuss the problems with everyone.

Along the way, there will of course also be time for you to ask me any questions that you may have. I will do my best to answer them.

Tuesday 22 February – *Exercises*

For the exercise class (13:00-15:00), prepare the following exercises for presentation and discussion in (online) class:

1. In the podcast we saw the following example.

   ```
   begin
         var x:= 0;
         var y:= 42

         proc p is x:= x+3;
         proc q is call p;

         begin
             var x:= 9;
             proc p is x:= x+1;

             call q;
             y:=x
         end
   end
   ```

   Now imagine that we have *mixed scope rules*. First consider the case where variables follow static scope rules whereas procedures follow dynamic scope rules. What is the final value of `y` then? And if we instead had dynamic scopes for variables but static scope rules for procedures?

2. A book on program design claims that in the language Pascal the construct `while E do C` is equivalent to calling the procedure

   ```
   procedure whiledo(e : Boolean; procedure c)
      begin
         if e then
            begin c;
                  whiledo(e,c)
            end
      end;
   ```

   where the first parameter is a call-by-value parameter and the second parameter is a procedure parameter, with the call `whiledo(E,PC)`, where `PC` is a parameterless procedure whose body is `C`.

   Why is this incorrect? Suggest a suitable correction.

3. A famous influencer on Instagram proposes to extend his favourite programming language with formal parameter list definitions of the form

   ```
   parlist mypars = (T1 X1,...,Tn Xn)
   ```

   where the `T1,...,Tn` are types.

   This could save a lot of space. Instead of writing e.g.

   ```
   function f1(T1 X1,...,Tn Xn)
   ... { body of f1 appears here }
   ```

```
function f2(T1 X1,...,Tn Xn)
... { body of f2 appears here }
```

we could simply write

```
function f1(mypars)
... { body of f1 appears here }

function f2(mypars)
... { body of f2 appears here }
```

Why should the influencer be discouraged?

4. Here is an ALGOL60 procedure that returns a real number. Remember that the only parameter passing mechanism in this language is call-by-name.

```
real procedure bingo(k, l, u, ak)
      value l, u;
      integer k, l, u;
      real ak;
      comment k and ak are call-by-name formal parameters;
   begin
      real s;
      s := 0;
      for k := l step 1 until u do
         s := s + ak;
      bingo := s
   end;
```

Now let `V` be a real-valued array with 100 entries.

What happens if we call `bingo(i, 1, 100, V[i])`?

5. In a language with dynamic scope rules, where is a free identifier in a function definition bound if it does not have a binding occurrence in the context in which it is called?

6. Can you find an example which shows that call-by-name and call-by-reference are not equivalent?