VE477 Introduction to Algorithms
Lab 1

Taoyue Xia, 518370910087

2021/09/28

2 Getting started with OCaml

2.1 More documentation

- Anonymous functions can be defined using the **fun** or **function** construct, for example, (fun x -> x * x).

  Anonymous functions are values and are used to be passed to other functions as inputs.

- Capital letters cannot be used for variables and functions, underscore is used instead.

- In OCaml, every piece of code is wrapped into a module, and a module can be submodule of another module. When you create a file, for example, named "amodule.ml", it will automatically define a module named "Amodule".

  Different from OOP, we don't need to create a new instance to use the methods and variables.

- **List** is the array type defined in OCaml. In package **Lists**, many operation methods on arrays are defined, such as **"List.partition"** and **"List.map"**.

- **List.map** and **List.iter** both can iterate through a list. However, **List.map** needs anonymous function to pass into it, and it will return a new list, while **List.iter** is like the usage of a for loop.

  List.map can map the input to an output according to the anonymous function, and **List.iter** is a more efficient way to iterate comparing with for and while loops.

- Folding can apply the defined anonymous function to elements in the list, and return the accumulated result.

  For example, **List.fold_left (+) 0 [1; 3; 5; 7]**;;, which will add 1 to 0, and then add 3 to the previous result, until 7 is added, which will give a result of $1 + 3 + 5 + 7 = 16$.

- Tail recursive functions can be optimized by compilers from recursive functions to while loops, if the recursion call is at the end of the function definition, and they run in constant stack space, avoiding stack overflow, which is important.

- **ref** means creating a reference and turn it into a pointer. **ref** is used when you want to make data mutable.
- A functor is a module that is parametrized by another module. It can define the data type of the new created module, like the template in C++ does. However, functors can refine signatures of modules, but templates are limited due to private and public attributes.
- **type t = type1 | type2 | type3 ... | type n**, or

  type t = {**type1 : t1**; **type2 : t2**; ...; **typen : tn**}
- Sum types generalize what is often known as enum, union, or variant record. Product types are like tuples, products of types.

  Sum types can be used to model finite sets, which is like enum in C. Product types are used when multiple properties exist simultaneously.