CS 230
Programming Languages

11 / 08 / 2022

Instructor: Michael Eckmann

Today's Topics

- Questions? / Comments?
- More types from chapter 6
  - Union
  - Array

Unions

- A Union is a **type** that can store *one of* different type values during the execution of a program. Why?
- Design issues
  - Type checking
    - Free
      - No type checking
    - Discriminated
      - Has type checking
  - Are unions allowed in records / classes

Unions

- A Union is a **type** that can store one of different type values during the execution of a program. Why?
- Design issues
  - Type checking
- Free (Fortran, C and C++ have this type)
  - No type checking
- Discriminated (Ada has this type)
  - Has type checking
- Java and C# (the newer languages) do not have unions.

Unions

- An example in C++.

```
union Number {
        int x;
        float y;
}
// in some function somewhere we can:
Number num;
num.x = 100;
// then we print the contents of num.x and num.y

num.y = 25.4;
// then we print the contents of num.x and num.y
```

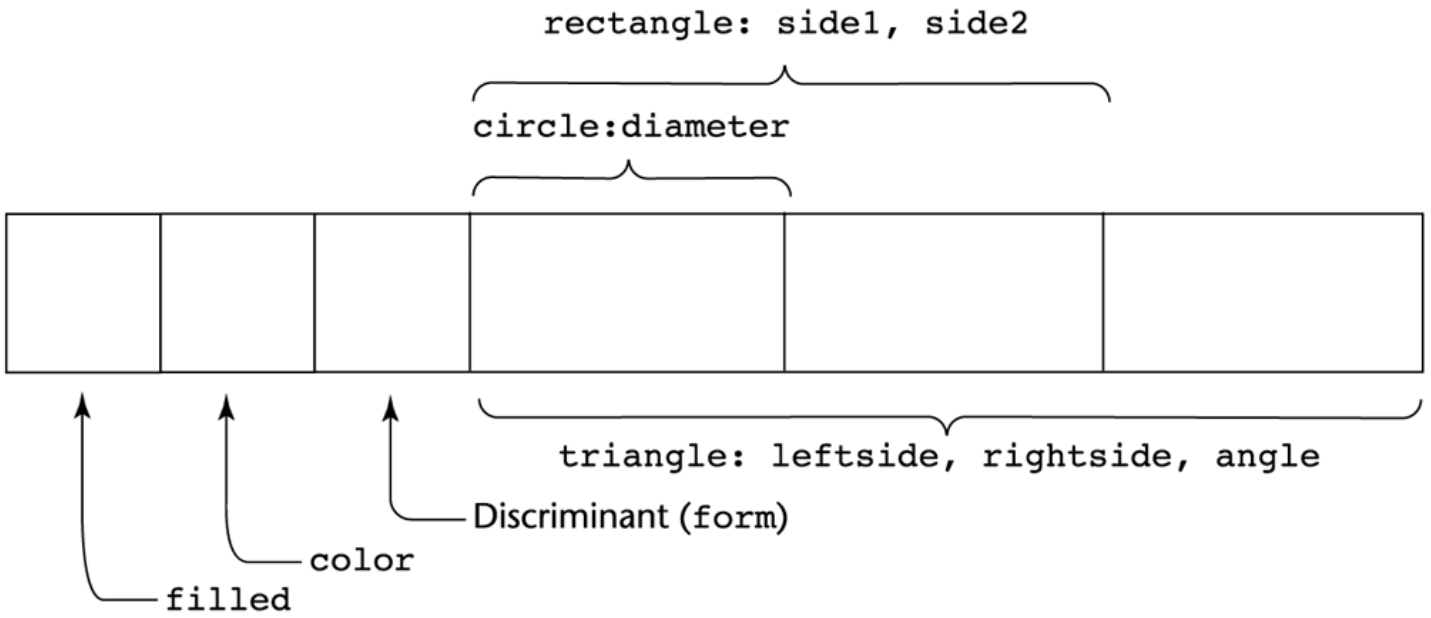
- note: this example is a modified version from Deitel & Deitel's C++ How to Program.

Unions

- When num.x = 100; and we print the contents of num.x and num.y, num.x will print fine, but a reference to num.y will be a logic error and will not contain 100. num.y will interpret the bits of how 100 was stored as an int as a float.
- Same problem if we assign num.y a float value and try to access num.x
- These problems exist because C++ has Free Unions. There is no type checking done to prevent reference to the wrong name within the union at any given time.
- Discriminated Unions would have to carry what kind of information to disallow the problem described above in C++?

Unions

- Discriminated union example of memory storage



Unions

- Evaluation of Unions.

Arrays

- We all know what arrays are.
- Design issues
  - Legal types for subscripts
  - Are references to subscript expressions range checked?
  - When are subscript ranges bound?
  - When is allocation bound?
  - Are ragged and multidimensional arrays allowed?
  - Is initialization allowed at allocation time?
  - Slices?

Arrays

- Most languages use the name of the array followed by an index in square brackets to specify an element of an array
  - e.g. arrayOfNames[5]
- Ada uses parentheses surrounding the index. Ada also uses parentheses for subprogram (function) calls to enclose the parameters. Any idea why they might have done that? Is it a good idea?

Arrays

- Ada chose to use parentheses in these two ways because both array references and function calls are similar in that they map the index (or parameters) to a value that is returned.
- It certainly reduces readability to some degree because the reader of the program can't tell the difference between an array reference and a function call with one parameter.

Arrays

- There are two types associated with any array
  - The type of the elements of the array
  - The type of the subscript
    - Most commonly integers (or subranges of integers)
    - Ada allows boolean, characters and enumeration types as subscripts
- Range checking the subscripts (indices) When are we talking here?
  - C, C++, Perl, and Fortran --- no range checking
  - Java, ML, C# do range check
  - Ada range checks by default, but can be programmer disabled
  - Why not range check?

Arrays

- Lower bound of subscripts
  - Usually 0 (C-based languages) Fortran95 defaults to 1.
  - Why use 0?
  - Given an array name and an index, how does the program find that element in memory?

Arrays

- Lower bound of subscripts
  - Usually 0 (C-based languages) Fortran95 defaults to 1.
  - Why use 0?
  - Given an array name and an index, how does the program find that element in memory?
    - The descriptor will contain
      - the address of first element of array
      - The type of each element → size of one element
    - So address of list[i] will be

      addressOfList + i*sizeOfElement

Arrays

- Four categories of arrays
  - Categories based on
    - Binding to subscript ranges
    - Binding to storage
    - Where the storage is (e.g. stack or heap)
  - If Binding to subscript ranges and storage is fixed for their lifetimes after binding, then the size cannot change.
  - The first 3 of the 4 following categories have this in common.

Arrays

- Four categories of arrays (Static, Fixed Stack-dynamic, Fixed Heap-dynamic, Heap-dynamic)
  - Static – subscript ranges and storage allocation is static. Statically bound variables are bound BEFORE run-time.
  - Fixed stack-dynamic – subscript ranges statically bound, but allocation done at declaration elaboration time. When is declaration elaboration time?
  - Let's compare these two. What are the advantages of each?

Arrays

- Static vs. Fixed stack-dynamic
  - Static
    - speed efficient --- no dynamic allocation or deallocation (at runtime) so they're faster.
  - Fixed stack-dynamic
    - space efficient --- if have block scoped arrays, they're only allocated when they need to be in memory instead of allocating all of them before run-time. And they can be deallocated from the stack when they're no longer needed.

Arrays

- Back to the four categories of arrays
- Note that for static and fixed stack-dynamic arrays, because the subscript range is statically bound, the size of the array is needed to be known prior to runtime. Besides static and fixed stack-dynamic there are
  - Fixed heap-dynamic
    - Subscript ranges dynamically bound
    - Storage allocation is dynamic (on the heap)
    - Both are fixed after they're bound which is different from:
  - Heap-dynamic
    - Subscript ranges dynamically bound
    - Storage allocation is dynamic (on the heap)
    - Both are changeable during execution.

Arrays

- In functions (therefore block scope)
  - C & C++ use static arrays if programmer uses the static modifier
  - C & C++ use fixed stack-dynamic arrays by default (without the static modifier)
- C & C++ also use fixed heap-dynamic arrays
- Java - fixed heap-dynamic arrays
  - Their subscripts and allocation are dynamically bound but fixed afterwards.

Arrays

- Python, Perl and Javascript (and C# has capability)
  - Heap-dynamic arrays
  - Perl has push to allow array to grow
  - Python has append (to grow) and pop (to shrink) and can concatenate arrays to grow as well

Arrays

- Array initialization
  - Sometimes the initialization implicitly states the length
  - In C, C++, C# and Java this happens
  - Ada has an interesting feature that allows initialization of some elements of the array to certain values and all others to another value.

Arrays

- Array operations (operate on array as a whole)
  - Some languages allow concatenation of arrays and can use relational operators on arrays
  - Others like APL have arithmetic operations on arrays like vector multiplication and matrix transpose, etc.

Arrays

- Rectangular vs. jagged arrays
  - Only makes sense when talking about multidimensional arrays
  - Jagged if rows **can** have different numbers of columns in the same array
  - I made can bold above, because Java, C and C++ support jagged arrays but **do not** support rectangular arrays.
  - This is because multidimensional arrays are implemented as arrays of arrays.
  - Fortran, Ada and C# provide rectangular arrays
  - When are different numbers of columns for a row useful?

Arrays

- Slices
  - Not a new data type
  - Instead, it is a way to reference part of an array as a separate unit.
  - e.g. One row, or some subset of consecutive elements in a column, or some rectangular subset of rows etc.
  - Python has support for slices
    - e.g. list[3:7]
    - that would be a slice of list from index 3 to 6

Arrays

- Implementation of array types
  - How to access an array element
    - Access function code needs to produce the address of an element
  - Descriptor for arrays
    - Needs to have information in it to construct the access function
- That is, when an element of an array is referenced, the index along with the information in the descriptor needs to be able to find the address in memory where that element lives.
- If run-time range checking is being done, what do you think the descriptor will contain?

Multidimensional Arrays

- Design issue:
  - How to store the multidimensional array in memory?
  - A 1-D array is typically stored sequentially in continguous memory locations

Multidimensional Arrays

- Design issue:
  - How to store the multidimensional array in memory?
  - Row major order
  - vs.
  - Column major order
  - Fortran uses column, all others use row

Multidimensional Arrays

- How to access 2-dimensional array elements at run-time (assuming row major order and all rows have same number of columns and index starts at 0).
  - We have the subscripts to look for
  - We need address of first element
  - What else do we need?

Multidimensional Arrays

- How to access 2-dimensional array elements at run-time (assuming row major order and all rows have same number of columns and index starts at 0).
  - We have the subscripts to look for
  - We need address of first element
  - We need the size of one element (determined from the type of the array.)
  - We need to know the number of columns per row
  - Luckily all this info is stored in the descriptor for the array

Multidimensional Arrays

- Address of element at row i, column j is =

addressOfArray +

(numCols * i + j) * sizeOfElement;

Pointers

- Pointer types
  - Store memory addresses or nil (no address.)
- For indirect addressing
- For dynamic storage management
  - Can access data at a particular memory address in the heap
- Provide a way to have data structures that shrink and grow during execution.
- Allocate space and deallocate space that the pointer points to