

## Lecture 5

### DATA TYPES

#### Primitive Types: (Atomic Types)

##### Numeric:

Integer types (often have alternative sizes: short, byte, long, unsigned integer, etc.)

Negative can be implemented by (explain each)

sign magnitude

2's complement

1's complement

Floating point types

IEEE Standards: (explain)

single precision 1 bit sign; 8 bits exponent, 23 bits fraction

double precision 1 bit sign; 11 bits exponent, 52 bits fraction

Decimal (Fixed) types: real numbers with fixed number of decimal places

Usually implemented as Binary Coded Decimal (BCD)

Ada, COBOL

Boolean

Implemented in most languages except for C where ( integer non-zero value is "true" and zero value "false")

C++ implements boolean as "bool" but can use C's implementation also

Character

ASCII 7-8 bit

Unicode 16-bit (Java, C#) (explain)

=====

Character Strings:

ISSUES:

- Should it be implemented as a character array
- Static or dynamic length

Arrays: Pascal, C, C++, Ada

Primitive type: FORTRAN, BASIC, C++ (standard string), Java

Functionality:

Assignment

Comparison

Searching (case sensitive or not)  
Concatenation  
Length calculation  
Differences in length and length definition

Turbo Pascal example: strings are of maximum length 255. They are implemented as an array indexed from 0 to 255 with the first character (0<sup>th</sup>) having the ASCII value of the length of the string; so, the length of a Turbo Pascal string is the ASCII value of the 0<sup>th</sup> element in the string array.

=====

## Enumeration Type

Ada: type DAYS is (Mon, Tue, Wed, Thu, Fri, Sat, Sun);  
Pascal: type colortype = (red, green, blue);

Why Enumeration types?

Some Implementations.

=====

## Array Types

### ISSUES:

- ❑ ❑ What types are legal for subscripts?
- ❑ ❑ What are the subscript ranges?
- ❑ ❑ How many subscripts (dimensions)?
- ❑ ❑ How are they stored in memory?
- ❑ ❑ Can they be initialized?
- ❑ ❑ Are slices allowed? If so, what kind?
- ❑ ❑ Should brackets or parentheses be used?

Static array: subscripts and size are fixed before run-time.

Fixed stack-dynamic array: arrays created usually in subprograms (functions and procedures) where the subscripts are known and has a fixed size.

Stack-dynamic array: One in which the subscripts are known but the size isn't.

Pseudocode example:

```
void doit ( int size ) {  
    int my_array[size];  
    ...  
}
```

Heap-dynamic array: One in which the subscripts are not known and the size isn't either. In C++ the vector class is somewhat a heap-dynamic array. (The subscripts are always known in C++).

In standard Pascal and in Ada the use of arrays without fixed subscript bounds and sizes as parameters to subprograms can be used very efficiently.

Examples: ...

In APL, Ada, C++ and other languages one can use a +, for example, to add corresponding array elements. In Ada and C++ this is called "overloading" the operator + and must have an explicit definition.

A slice of an array is a substructure of an array. FORTRAN has very liberal ways of doing slices. Ada is more restrictive.

Array Implementations:

Single dimensional array: Element type, Index type, Index lower bound, Index upper bound, address

Column major order vs. Row major order (FORTRAN uses column major order)

=====

Maps and Associative Arrays

=====

Record Types

Implementation: address; Field 1: name, type, offset; ... ;Field n: name, type, offset

=====

Union Types: may store different types during different times during the program's execution.

=====

Set Types:

## Pascal implementation

=====

Pointer Types: Range of values consisting of addresses and a special value called nil (NULL in C/C++). Generally, used to access values on the HEAP.

### ISSUES:

- ❑ ❑ Scope and lifetime of a pointer variable.
- ❑ ❑ Lifetime of a heap-dynamic variable.
- ❑ ❑ Restrictions on the type of values to which they can point.
- ❑ ❑ Use for storage management, indirect addressing, or both

Two operations of pointers: assignment and dereferencing.

C++ comments ... use of \*, &, and ->

Dangling pointer: a pointer that contains the address of a de-allocated heap-dynamic variable.

Lost heap-dynamic variable.

Pascal: pointers are only used to access dynamically allocated, anonymous variables.

Ada pointers are like Pascal's except that lost heap-dynamic variable may be de-allocated at the end of the pointer's scope.

C/C++ pointers can be liberally used (storage management and indirect addressing). Pascal and Ada83 pointers can point only into the heap. C/C++ pointers can point almost anywhere. Pointer "arithmetic" is also used, and array names are just pointers to the first array element.

Java note: In Java there are no programmable pointers; however, the entire language is based upon pointers.

"Garbage Collection"