CS 230
Programming Languages

10 / 20 / 2022

Instructor: Michael Eckmann

Today's Topics

- Questions? / Comments?
- Scope
- Chapter 6 Data Types
  - Floating point
  - Other common data types
  - Character strings
  - Enumeration types

Scope

- Scope is the range of statements in which a variable is visible (which means where it can be referenced.)
- Static scope
- Static – scope of variables can be determined **prior to run-time**. *It is a **spatial** relationship.*
  - Nesting of
    - functions/methods (p. 220)
    - Blocks
  - Answers the question – When we reference a name of a variable which variable is it?

Scope

- Dynamic scope
- Scope of variables are determined **at run-time**. *It is a* ***temporal*** *relationship, based on calling sequence.*
  - Advantage: convenience
  - Disadvantage: poor readability, poor reliability
  - Also answers the question – When we reference a name of a variable which variable is it? -- but we may get a different answer vs. if the language used static scoping.
  - example

Constants and initialization

- What good are named constants?
  - Enhances readability. Anything else?

- Initialization
  - Binds a value to a variable when that variable is bound to storage (i.e. when memory is allocated.)
  - Occurs once for static variables
  - Occurs each time code is executed for dynamic variables (either stack-dynamic or heap-dynamic.)

More Ch. 6 - Data Types

- A Data type
  - defines a set of allowable values
  - defines the operations on those values

Ch. 6 - Data Types

- How well do the types match real-world problems is a question you might ask when considering if a language has useful types.
- Does the language you're evaluating support/provide the following:
  - Built-in types --- those that are built into the language (e.g. the primitive types in Java)
  - Standard types (via a standard library) --- (e.g. those like String, etc. that are provided in the Java API)
  - User defined types --- example anyone?
  - Structured data types
    - e.g. Arrays, records
- Why might we care if a language provided a type as a primitive type vs. through some standard library?

Ch. 6 - Data Types

- Languages maintain a descriptor at compile-time and/or run-time for each variable
- Descriptor
  - collection of attributes about a var stored in memory.
  - Static (during compilation process) vs. dynamic (during run time)
  - Used for type checking and allocation & deallocation
  - What is stored is only what is necessary to determine correct types, and where to find the values in memory
  - e.g. Type Name, Address, Length/Size

Primitive Data Types

- Not defined in terms of other types
- Used with type constructors to provide structured types
- Integer types
  - Signed vs. unsigned
  - Sign-magnitude vs. Two's complement vs. one's complement representation of integers.

Primitive Data Types

- Floating point are approximations to decimal numbers. (e.g. let's convert 0.1 in decimal to binary)
  - Fully continuous?
  - Precision and range.
    - single precision IEEE format:
      - 1 sign bit, 8 bits for exponent, 23 bits for fraction
    - double precision IEEE format:
      - 1 sign bit, 11 bits for exponent, 52 bits for fraction

Primitive Data Types

- How to “fix” problems with floating point?

Primitive Data Types

- How to “fix” problems with floating point?

- Intervals
  - Interval arithmetic could guarantee that result of all operations are within the interval
  - Have a min and max bound on results.

Primitive Data Types

- Decimal types
  - Most languages we use don't have a decimal type.
  - Cobol does, so does C#.
  - BCD
  - 1 or 2 digits per byte
  - Why have decimal types if they waste storage?

Primitive Data Types

- Decimal types
  - Most languages we use don't have a decimal type.
  - Cobol does, so does C#.
  - BCD
  - 1 or 2 digits per byte
  - Why have decimal types if they waste storage?
    - Because they precisely store decimal values, whereas floating point types do not

Primitive Data Types

- Boolean
  - Could be represented by a single bit, but are often represented by a byte for more efficient memory access.
- Character
  - ASCII (8 bit)
  - ISO 8859-1 (8 bit)
  - Unicode (16 bit) --- Java & c# use this
    - 1st 128 are ASCII

ASCII

ASCII value	Character	Control character	ASCII value	Character	ASCII value	Character	ASCII value	Character
000	(null)	NUL	032	(space)	064	@	096	
001	☺	SOH	033	!	065	A	097	a
002	☻	STX	034	"	066	B	098	b
003	♥	ETX	035	#	067	C	099	c
004	♦	EOT	036	$	068	D	100	d
005	♣	ENQ	037	%	069	E	101	e
006	♠	ACK	038	&	070	F	102	f
007	(beep)	BEL	039	'	071	G	103	g
008	◘	BS	040	(	072	H	104	h
009	(tab)	HT	041	)	073	I	105	i
010	(line feed)	LF	042	*	074	J	106	j
011	(home)	VT	043	+	075	K	107	k
012	(form feed)	FF	044	,	076	L	108	l
013	(carriage return)	CR	045	-	077	M	109	m
014	♫	SO	046	.	078	N	110	n
015	☼	SI	047	/	079	O	111	o
016	►	DLE	048	0	080	P	112	p
017	◄	DC1	049	1	081	Q	113	q
018	↕	DC2	050	2	082	R	114	r
019	!!	DC3	051	3	083	S	115	s
020	¶	DC4	052	4	084	T	116	t
021	§	NAK	053	5	085	U	117	u
022	▬	SYN	054	6	086	V	118	v
023	↨	ETB	055	7	087	W	119	w
024	↑	CAN	056	8	088	X	120	x
025	↓	EM	057	9	089	Y	121	y
026	→	SUB	058	:	090	Z	122	z
027	←	ESC	059	;	091	[	123	{
028	(cursor right)	FS	060	<	092	\	124	|
029	(cursor left)	GS	061	=	093	]	125	}
030	(cursor up)	RS	062	>	094	^	126	~
031	(cursor down)	US	063	?	095	_	127	⌂

Unicode

- http://www.unicode.org/standard/WhatIsUnicode.html
- Has characters in alphabets of many languages

Character strings

- Implemented as
  - Character array vs. primitive type
  - Static vs. dynamic length
- C & C++ strings are terminated with a null character (ASCII 0).
  - Length is not maintained but can be determined. The end of the string is determined by the null character.
  - All that needs to be stored is a pointer to the first character in the string.
  - Limited dynamic length (limited because there is a maximum length)

Character strings

- Java has two different types for Strings.
  - String
    - Static length, constant strings
  - StringBuffer
    - Dynamic length and allows direct subscripting
- Fortran 95
  - String is a primitive type
  - Has typical operations for strings
- Issues
  - What to do when assigning strings of different lengths? etc.

Character strings

- Pattern matching with character strings
  - Python, Perl, JavaScript, PHP built in pattern matching with regular expressions
  - Included in class libraries of C++, Java and C#

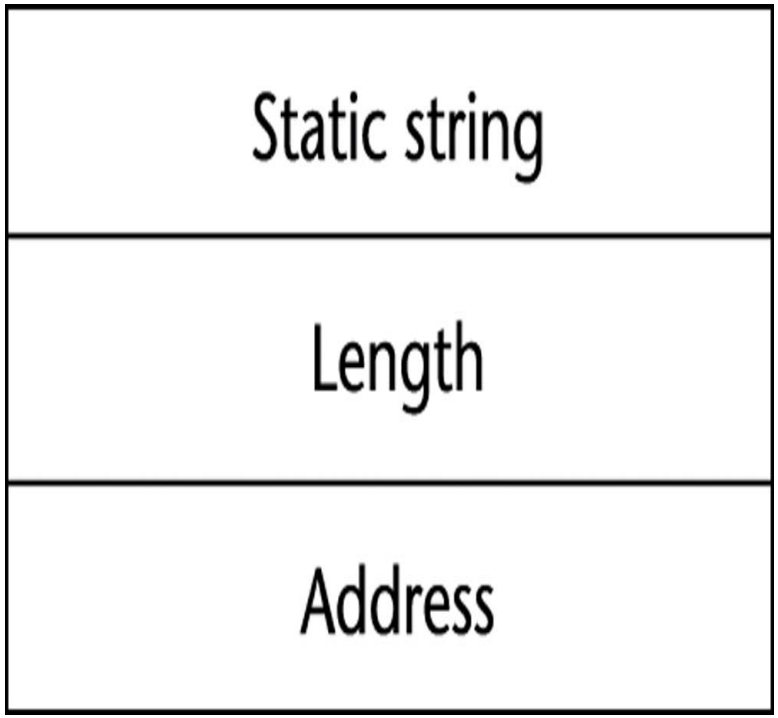
Character strings

- Length
  - Static length
  - Limited dynamic length
  - Dynamic length
    - Requires dynamic storage allocation and deallocation
    - Maximum flexibility

Character strings

- Implementation
  - Most often **software** (as opposed to **hardware**) implementation of storage, retrieval and manipulation
- Descriptor
  - Type name
  - Length (for static strings)
  - Address of first character

Character strings
Descriptor for static strings



Michael Eckmann - Skidmore College - CS 230 - Fall 2022

Character strings

- Descriptor for limited dynamic
  - Type name (Limited dynamic string)
  - Maximum length
  - Current length
  - Address of first character
- For dynamic length strings
  - Could be stored in linked list
  - Could be stored in adjacent storage cells
    - What about when length changes? How can this be done?

Character strings

- Linked list vs. adjacent evaluation
  - Linked lists
    - string operations become complex (following pointers around), no direct access to each character
    - allocation and deallocation is simple and fast
    - but requires more storage (for the pointers or references)
  - Adjacent
    - faster string operations (because contiguous), allows direct access to each character
    - less storage
    - but allocation and deallocation is slower

Ordinal / Enumeration

- Ordinal types
  - Range of values are associated with positive integers
- Many languages provide support for user defined ordinal types like Enumerations.
- Enumeration types
  - Named constants represent positive integers
  - Design issues
    - Are enumeration type values coerced to integer?
      - Same operations and range of values of integers
    - Are any types coerced to enumeration type values?
      - Could break the allowable values of the enum type
  - Range of values intentionally limited.

Enumeration

- e.g. In C#

```
enum months {Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec};
```

- e.g. In C++

```
enum months {Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec};


 months m;        // declare a variable of type months
```

- Represented as 0 to 11 typically but could be given programmer specified values in the declaration.
- They look the same, but C# enum types not coerced to integer whereas C++'s are --- so, C# doesn't allow operations that may not make sense, while C++ does. e.g. In C++, we can add 1 to an enum

Enumeration

- e.g. In C++

```
enum months {Jan, Feb, Mar, Apr, May, Jun, Jul,
Aug, Sep, Oct, Nov, Dec};


// declares currentMonth of type months and assigns
// an initial value
months currentMonth = Jun;
```

- In C++, we can add 1 to an enum
  - e.g. currentMonth++;
- Any danger here?

Enumeration

- Java provides support of enumerations (since Java 1.5). When used, they create full classes for the enum. See:

http://java.sun.com/j2se/1.5.0/docs/guide/language/enums.html

- What are advantages to enumeration types?

Enumeration

- What are advantages to enumeration types?
  - All forms of implementation of them offer better readability
  - Reliability
    - No arithmetic operations on the enum types in Ada and C# --- this prevents adding/subtracting which might not make sense like in adding months together
    - Ada and C# also restrict values within the range for the type.
    - C treats them like integers so we don't get those advantages

Enumeration

- Reliability
  - C++ allows numeric values to be assigned to enum types if they are explicitly cast.
  - C++ also allows values to be assigned to enum types but the range is checked for validity.
    - This just checks that the integer is between the lowest and highest enum value. Enum values need not be consecutive integers. Any problem here?