CSE 3302 Notes 5: Control Flow

(Last updated 10/10/12 12:00 PM)

6.1. Expression Evaluation

Variables

Value (container) model

r-value = expression allowed on right side of assignment

l-value = expression allowed on left side of assignment
(address not pointing to a constant)

1. Build suffix array (`sa`) of subscripts based on corresponding suffixes of a text.

```
     0  1  2  3  4  5  6  7  8  9 10 11
s    a  b  c  d  a  b  c  d  a  b  c \0

sa  11  8  4  0  9  5  1 10  6  2  7  3
```

Key-comparison sorts can construct in $O\left(n^2 \log n\right)$ time (expected time for `qsort`):

```
int suffixCompare(const void *xVoidPt,const void *yVoidPt)
{
// Used in qsort call to generate suffix array.
int *xPt=(int*) xVoidPt,*yPt=(int*) yVoidPt;

return strcmp(&s[*xPt],&s[*yPt]);
}

scanf("%s",s);
n=strlen(s)+1;
for (i=0;i<n;i++)
  sa[i]=i;
qsort(sa,n,sizeof(int),suffixCompare);
```

2. Matrix allocation

```
int **succ,**result,**rowPartition,**colPartition,n;

allocate(int size1,int size2,int ***matrix)
{
  int i;

  if (!(*matrix = (int **) malloc(size1*sizeof(int *))))
  {
    printf("Allocate failed\n");
    exit(2);
  }
  for(i = 0; i < size1; i++)
    if (!((*matrix)[i]=(int *) malloc(size2*sizeof(int))))
    {
      printf("Allocate failed\n");
      exit(3);
    }
}

deallocate(int size1,int **matrix)
{
  int i;

  for(i = 0; i<size1; i++)
    free(matrix[i]);
  free(matrix);
}
...
allocate(n,n,&succ);
...
deallocate(n,succ);
```

3. `lrValues.c`

Reference model

All variables are l-values

Using a variable on either side of assignment involves dereferencing

(Scott's "Only one 2" . . .)

Java . . . primitive types

Boxing

Java `Integer` and `int`

Multiway (simultaneous, parallel) assignment

```
a, b = b, a;

i, j, a[i], a[j] = j, a[i], a[j], i;
```

CSE 2320 Fall 2011 Lab 1 . . .

JavaScript 1.7 - Destructuring assignment

```
[a,b] = [1,2];
[a,b] = [b+1,a+3];
[a,a] = [b+2,a+1];
```
What happens?

Initialization

Expression ordering

Argument processing order for C (as opposed to '`,`' operator) - `argOrder.c`

Short-circuit boolean expressions

Lab 3 . . .

Boolean operators to force sub-expression evaluation (for side effects)

C - Use `&` or `*` in place of `&&`, | or `+` in place of `||`

6.2. Structured and Unstructured Flow

gotos - multiple level break

Multilevel returns/Signals (and `setjmp`/`longjmp`)/Exceptions

Continuations (later . . .)

6.3. Sequencing

Not much here . . .

6.4. Selection

Special syntax for if ... then ... else ... and avoiding dangling else

Guarded Commands:

```
if   condition -> statement
  [] condition -> statement
  [] condition -> statement
  ...
  else statement?
fi
```

Switch

Generality of individual expressions

Implementation

O(1) - table/hashtable
O(log n) - binary search
O(n) - like corresponding ifs (JavaScript)

Also, see Duff's device (exploration 6.38) for exploiting C `case` fall-through property.

6.5. Iteration

Enumeration-controlled ("for")

Special syntax for "while" or should number of iterations be known at onset?

p. 259 issues

Jumping into or out-of loop?

Is expression that index variable is tested against required to be constant?

Modifying index variable inside body?

Availability of index variable after loop termination?

Iterators - container abstraction

Comparing two binary search trees?

C++ overloaded operators for iterators - `www.cgal.org`

Functional language iterators

Logically-controlled

Guarded Commands:

```
do   condition -> statement
  [] condition -> statement
  [] condition -> statement
  ...
od
```

6.6. Recursion (later)

Issues compared to iteration

Applicative and normal-order evaluation

Lazy evaluation - delay and force, lazy data structures