Trees, Recursion, Expressions, Stacks

"Context-Free" appearance of recursive calls.

Records.

How Recursion Works: Review: Automatic Variables, Activations, Activation

CSI 310: Lecture 16
First, note \( n \) is a local variable.

\[
\begin{array}{ll}
\text{int \text{fact}(\text{int} \ n)} & \text{if}(n=0) \text{return 1;}
\end{array}
\]

\[
\begin{array}{ll}
\text{else return fact(n-1);}
\end{array}
\]
DIFFERENT ACTIVATION RECORDS

The storage used for AUTOMATIC VARIABLES is in the activation record.

The EVENT of "calling one function once" is an ACTIVATION.

Each time a function is CALLED, an ACTIVATION RECORD is created.

VARIALEs!
4. Now, automatic variables are in the activation record of the activation.

RETURNED TO.

3. The activation whose call originally created this activation is

RETURNED TO.

2. This activation's activation record "goes away"

1. The return value (if any) is saved for use by the caller.

When a function activation executes the return statement...
particular activation of the (recursive) function MergeSort is doing.

VERY IMPORTANT: AUTOMATIC variables are used to keep track of what a
But, running fact(0) calculates $0! = 0$ is true and returns 1.

Then computes $720 \times 7 = 5040$ and returns it.

For example, running fact(7) computes $7! = 5040$, which returns

```c
int fact(int n)
{
    if (n==0) return 1;
    else return fact(n-1)*n;
}
```

Example: When the function runs, calls the same function, either directly or indirectly.

Definition: A function is recursive means the body of the function sometimes,

re-implemented itself.

(3) Elegant way to write programs; performance can be improved with routine.

(2) Powerful problem solving technique.

(1) Understanding, not just programming, data structures and algorithms.

Recursion: Trees, Recursion, Expressions, Stacks are closely related.
{ return Solution; }

Solution = MERGE( Anst, Anst2 );

node * Anst2 = MERGE( Subprogram2 );

node * Anst = MERGE( Subprogram1 );

// RECURSE one or more times:
Subprogram2 = Instance; // SPLITT removed half the original list.

Subprogram1 = SPLITT ( Instance );

node * Subprogram1, Subprogram2;
}

else
{

Solution = Instance; // TRIVIAL SORT OF A 1-ITEM LIST!

if ( LENGTH ( Instance ) == 1 )

dnode * Solution;

node * MERGE (node * Instance)

 Divide and Conquer Pattern Applied To Sorting

University at Albany Computer Science Dept.
of the half that WAS REMOVED from the linked list.

(2) The return value is the addr. of the first node
the nodes removed.

// post: (1) instance=ort_regal value, but with approx. half
linked list of AT LEAST TWO C-strings.
// pre: instance=the addr. of the first node of a
node * SPLIT (node * instance )

C-strings will be compared using strcmp() from <string.h>
dnode is a linked-list node type; each node's data is the address of a C-string.

Merge sort explained using Pre/Postconditions:
// Non-Functional Requirement: Use the Merge Sort Algorithm.

strcmp(order).

Post: The return value is the address of the first node of the linked list of the original C-strings sorted into.

Pre: Instance=the address of the first node of a

node * MergeSort(struct C-Strings * instance);
(q) One arc from this tree’s root to the root of each of the trees specified under or the root.

(c) Zero or more rooted trees, with no nodes or arcs in common with each other.

(a) One root node.

What is a tree?
An expression: (q), are substructures of the expression. Any operator and operands under (c) (and overlapping). (and) or more expressions (and) as operands (no or has an operator, it has one sublevel. (and) or has a top level operator, except either is an identifier or constant.

Under (q), the root of each of the trees specified (c) (and). One arc from this tree's root to other or the root. (and) no nodes or arcs in common with each (p) zero or more rooted trees, with (a) one root node. (and) A tree has:
XXX employed.

smartness with the complex C++ precedence/associativity rules, FIRE that
obvious. (2) If a programmer you are supervising times to show off his/her

2 practical rules: (1) If it's doubtful or subtle, USE PARENTHESES to make it

\[
\begin{align*}
&3^2 = 9 \\
&5 \times 7 = 35 \\
&5 \times (4 + 3) = 35 \\
&5 + 20 = 25 \\
&5 \times 4 + 3 = 23
\end{align*}
\]

not precedence than addition from elementary school.

must memorize or look up. They begin with "multiplication has higher

Rather complicated operator precedence and associativity rules people

How do you know which operator is evaluated first?

\[
A = B + C - D * E + F * G
\]

Not fully parenthesized:

expressions.

To make learning these ideas easier, we will start with fully parenthesized
The top level operation "Assign to A" is executed LAST! Why MUST it be done last? It uses the results of the all previous operations!

7. Assign it also to A.

6. Assign the last sum to B.

5. Add subtraction's result to this last product.

4. Multiply F and G.

3. Subtract that from G, remember result.

2. Multiply old value of E by D.

1. Increment E first.

means:

\[
A = (B \times C) + ((E + D) \times F)
\]

Fully parenthesized:

\[
A = B \times C + (E + D) \times F
\]
Example of an expression and its Parse Tree

\[ A = (B = (C - (D \times (E + + ))) + (F \times G)) \]

Details continued on the next 2 frames.
top level operator is multiplication (\( \ast \))

(F*G

\( F \) \( G \))
\( C - (D \times (E++)) \)

Top level operator is subtraction ( - )

\( (C - (D \times (E++))) \)

Top level operator is subtraction ( - )

\( D \times (E++) \)

Top level operator is multiplication ( * )

\( (E++) \)

Top level operator is increment ( ++ )

\( F \)

Identifier
An expression

definitions:

Your Job: Check that these examples, purporting
to be „trees and expressions“ FIT these
...
Expression deClause (p) is OK.

The operands are subtractions:

Expression deClause (c) is OK.

Top level operator is subtraction (−)

Expression deCexpression as operands.

Is I or more

which

identifier

E

Top level operator is increment (++)

(++)

Top level operator is multiplication (*)

D(E++)

Is OK?

Clause (a)

(c) − D*(E++)

Has an operator:
and return its result.

(3) Combine the results from (2) using the meaning of the operator to compute
of the operands. (Only one call for a unary operator.)

(2) RECURSIVEELY Call Evaluate(L1), call Evaluate(L2) for each of the trees
identifier. So, return it or its value.

(1) If L is just one node only, then the expression must be a constant or

Evaluater(ParserTree L)

The following recursive algorithm evaluates an expression when given its parse

The expression is called the expression's Parse Tree.

The "tree of an expression" is called the expression's Parsing Tree.

The (rather difficult and non-trivial) job of building out the tree from a given

expression with

expression's structure with
recursive function.

Lab4 is to OBSERVE the stack of ACTIVATION RECORDS during the run of a

(Insert into a stack is called push, delete from a stack is called pop.)

and non-recursive.

other data relevant to all C/C++ function calls and returns, both recursive
runs C/C++ programs. Implementing and organizing local variables and

3. The "run-time stack" of activation records, internal to the system when it

2. Storing and organizing intermediate results when evaluating expressions.

1. Figuring which pairs of parentheses MATCH in a correctly nested

3 uses for stacks:

ONLY ONE END (called the top.)

that access, insertion and deletion are permitted at

What is a stack? A stack is a sequence that is restricted so
reversing.

5. Analyze/evaluate fully parenthesized expression

4. Testing string (simpler) string for nesting recursively.

3. Testing string <({{}}(())[])> for properly nested parenthetization using a stack.

2. Reversing string BATMAN using a recursive procedure, visualizing the stack of activation records. Each activation record stores one character.

1. Reversing string BATMAN using a stack.

At this point, we did low-tech dramatizations: