School of Computer Science

COMP SCI 1103/2103 Algorithm Design & Data Structure

Recursion 3

Previously on ADDS

- Recursion
- Checklist of recursion
- Four stage in programming recursion
- How to improve the efficiency of recursion
  - *Recursive helper function*: Recall that creating new strings made our isPalindrome so inefficient, but we made it work with only one string
  - *Tail recursion*; c++ compilers understands when they can do a **tail call elimination**

Overview

- In this lecture we will discuss:
  - Improving efficiency through Memoisation
  - More examples of recursion.
  - Indirect recursion

Improving Efficiency with Memoisation - Trucks Revisited

- *Memoisation (not a spelling error!)*: store the obtained results of recursive function calls somewhere, so that you don't go through it again

  - For the call `numTrucks(10, 2)`, how many times do we have to recursively call `numTrucks(2, 2)`?
  - Instead of making another call, we could store results and lookup in a table
    - Assumes lookup takes less time than function call and calculation;
  - How would we redesign numTrucks() to do this?

More examples of recursion - Greatest Common Divisor

- GCD is a mathematic problem to find the largest positive integer that is divisor of two or more integer
- Iterative way

```
int iterGCD(int a, int b) {

    int newB;

    while(b!=0){
        newB = a % b;
        a = b;
        b = newB;
    }
    return a;
}
```

- Recursive way
  - Euclidean algorithm
  - gcd(a,b) = gcd(b, a%b)
  - Is this a tail recursion?

```
int recursiveGCD(int a, int b) {

    if (b==0)
        return a;

    return gcd(b, a%b)
}
```

Example - Maze

```
* ******
*    * *
* **** *
* * *  *
* * ** *
*   *  *
*** *  *
*      *
****** *
```

- Escaping a Maze
  - Some starting position
  - The walls of the maze are indicated by *.
  - char * maze[9][9]

- Think recursively!
  - Currently located at [x,y]. Can you escape from here?
  - Base:
    - If [x,y] is empty, on boundary, and is not the starting position, then YES!
    - If [x,y] is not empty, or is out of the boundary then NO.
  - Recursion: Can you escape from one of the neighboring locations?
    - Without revisiting the current location (To prevent an infinite loop).

Example - Maze

- Pseudo code

```
bool escape(int x, int y){
  if((x<0)||(x>9)||(y<0)||(y>8)){
    return false;
  }

  if(maze[x][y] == '*'){
    return false;
  }

  if we are on the boundary
          if maze[x][y] == ' ' && we are not at the starting position
                  return true;

  if maze[x][y] == 'v'
           return false;

  maze[x][y] = 'v'; // before moving to the next position
                    // mark the current possition as visited

  if ((escape(x+1,y) || (escape(x-1,y) || (escape(x,y-1) || (escape(x,y+1))
       return true;
   else
      return false;
}
```

Indirect Recursion

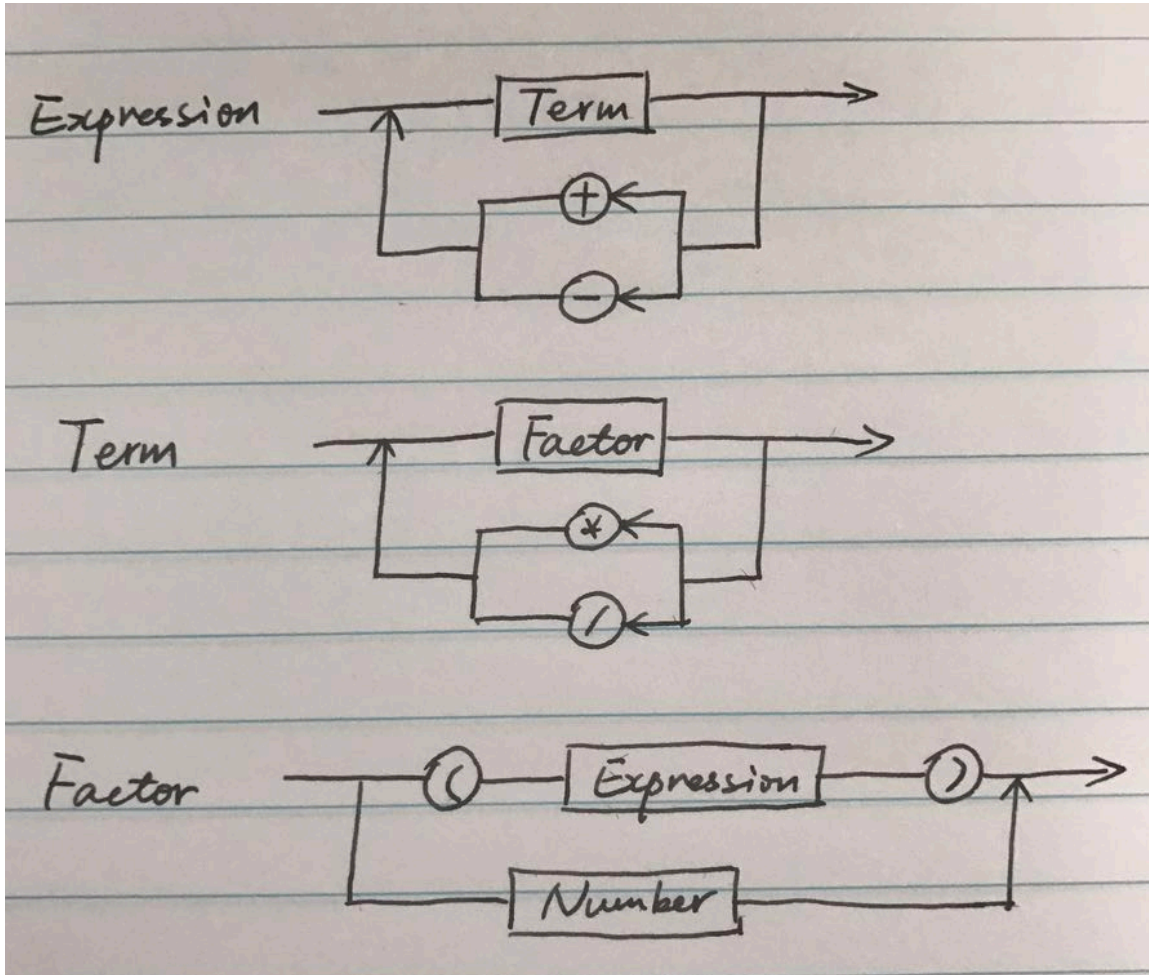
- So far we've looked at direct recursion - a function calling itself.
- Indirect recursion occurs when a function calls itself through the intermediary of another function.
- For example a function `Func1` calls a function `Func2`, which then calls `Func1` again.

Example

- Compute the values of arithmetic expressions

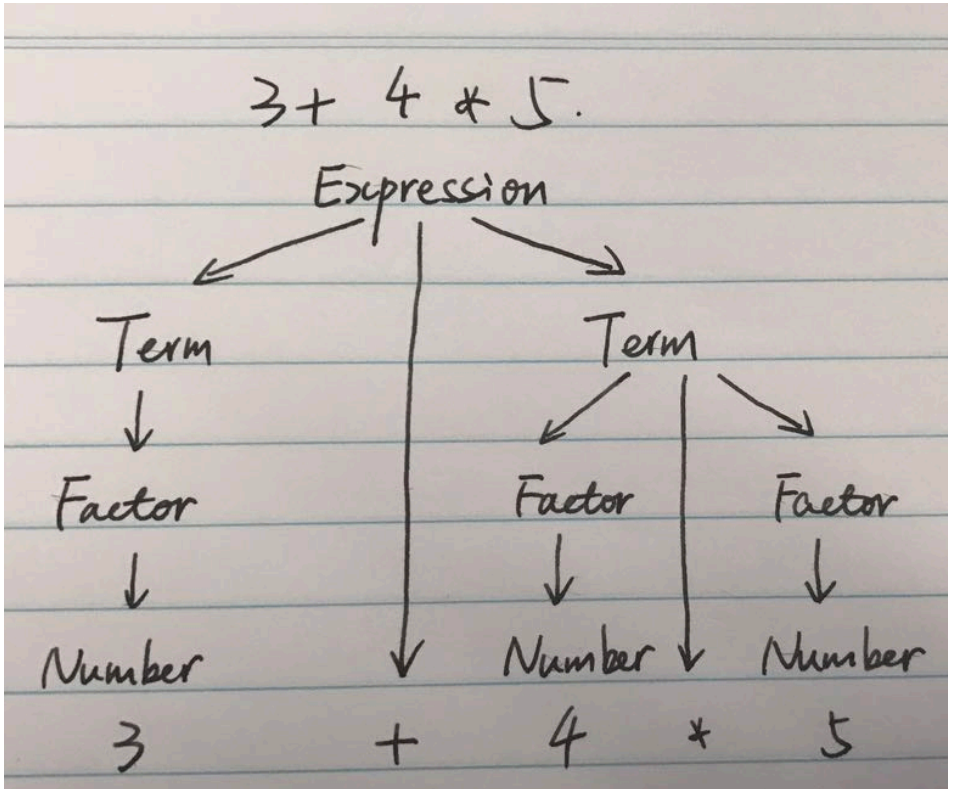
- Example
  - 3+4*5
  - (3+4)*5
  - 1+(2+(3+(4+5)))

- Is there an infinite loop?

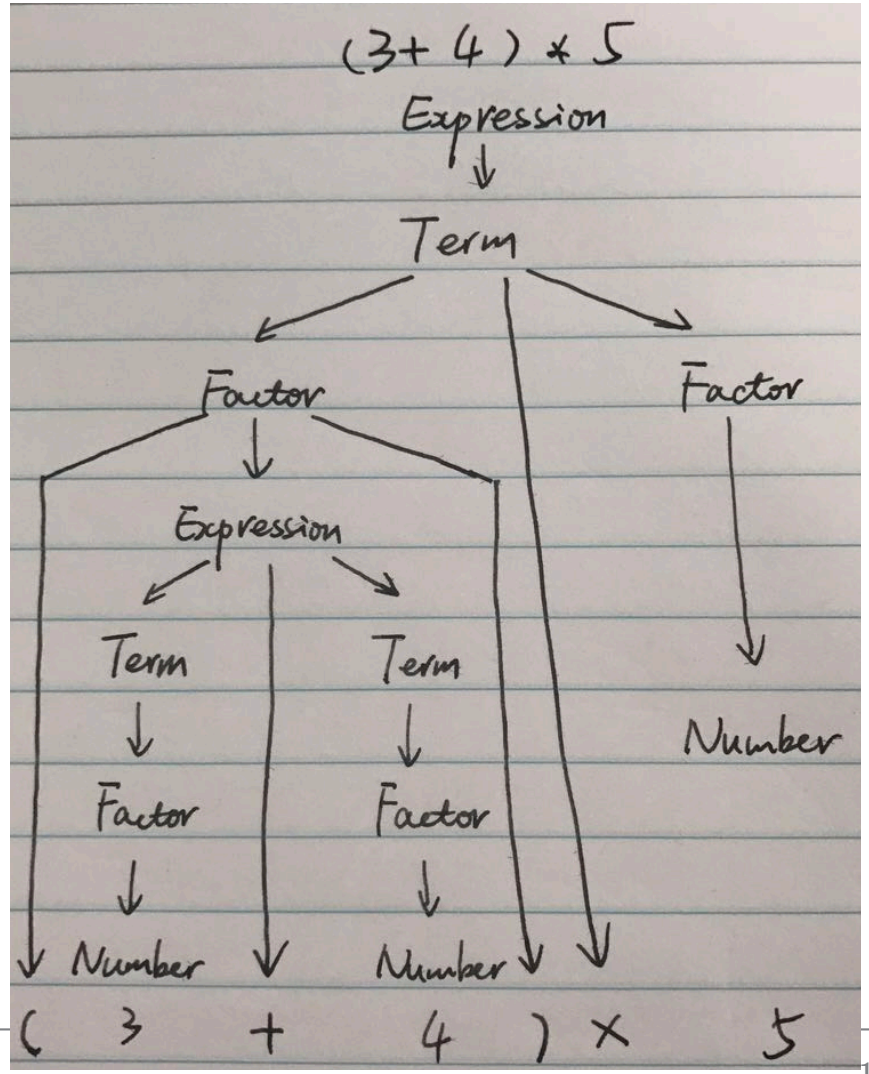


Example

- Syntax tree for 3+4*5



- (3+4*5)



Sample code for getExpression

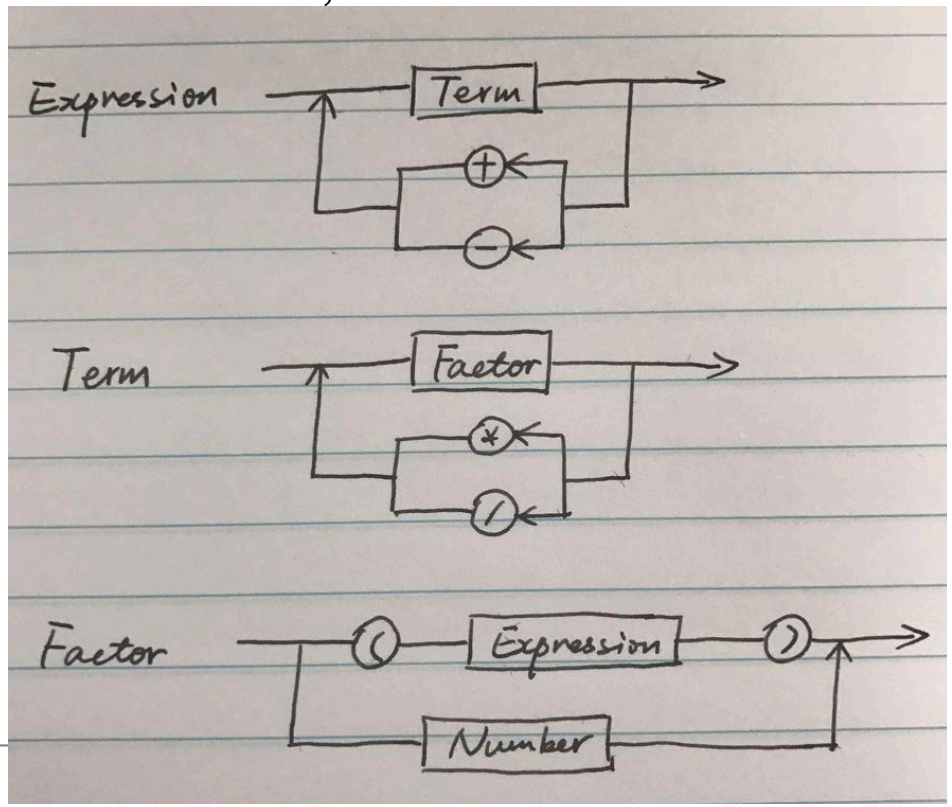
- Assume we have
  - getChar()
    looks ahead in input string
  - removeChar()
    removes a char from input string
  - getNumber()
    reads a number from the input string and removes the corresponding characters from that
- How can we write getExp() using these helper functions?

example code

```
int getExp(){
   int value = getTerm();
   char next = getChar();
   while ((next =='+') ||(next=='-')){
            removeChar();
            int value2 = getTerm();
            if(next =='+')
               value += value2;
            else
               value -= value2;
   }
   return value;
}
```

```
int getFactor(){
   char next = getChar();
   if(next =='(') {
         value = getExp();
         removeChar();
         return value;
   }
    else
         return getNumber();
}
```

```
int getTerm(){
   int value = getFactor();
   char next = getChar();
   while ((next =='*') ||(next=='/')){
            removeChar();
            int value2 = getFactor();
            if(next =='*')
               value *= value2;
            else
               value /= value2;
   }
   return value;
```



Example

- For calculating (3+4)*5
- getExp()
  - getTerm()
    - getFactor() ->consume ‘(’
      - getExp() ->3+4, return 7
    - ->consume ‘)’
    - getFactor() –> return 5
  - 7*5 -> return 35
- return 35

Summary

- Recursion is a useful tool for understanding problems and producing readable solutions.
- In designing recursive functions, we need to keep in mind the two important factors of recursion
  - Base cases
  - Recursion relationship
- We saw More examples of recursion
- Indirect recursion
  - Harder to track and control


THE UNIVERSITY
of ADELAIDE