L14 - Recursion

November 13, 2019

0.1 Recursion

Administrivia

- wed Dec 4: class cancelled
- Fri Dec 6: no lecture; Monday schedule
- Final exam: comprehensive; weighted toward 2nd half of course

0.2 Calculating Factorial

How do we calculate 6!? We repeatedly calculate partial products:

$$6! = 1 \times 2 \times 3 \times 4 \times 5 \times 6 = 720$$

In [3]:
```
// calculate n! iteratively
#include <stdio.h>
#include <stdlib.h>

int factorial (int n)
{
    int result = 1;

    for (int i = 1; i <= n; i++)
    {
        result = result * i;
    }

    return result;
}

int main()
{
    int n = 6;
    printf("factorial: %d\n", factorial(n));
    return 0;
}
```

```
factorial: 720
```

```
In [7]: // calculate n! recursively
        #include <stdio.h>
        #include <stdlib.h>

        int factorial(int n)
        {
            if (n == 1)
            {
                // the base case of factorial
                printf("Calling factorial(1); returning 1\n");
                return 1;
            }
            else
            {
                // not the base case; call factorial again on a smaller problem
                printf("Calling factorial(%d)\n", n);
                return n * factorial(n-1);
            }
        }

        int main()
        {
            int n = 6;
            printf("factorial: %d\n", factorial(n));
            return 0;
        }
```

```
Calling factorial(6)
Calling factorial(5)
Calling factorial(4)
Calling factorial(3)
Calling factorial(2)
Calling factorial(1); returning 1
factorial: 720
```

0.3 Calculating the sum of an array

We can calculate the sum of integers in an array recursively:

- base case: the sum of an array of one element is the value of that element
  - `sum(a[0] ... a[0]) = a[0]`
- recursive case: the sum of an array is the sum of `a[0]` and the sum of the rest of the array
  - `sum(a[0] ... a[n-1]) = sum(a[0] ... a[n-2] + a[n-1]`

```
In [9]: // calculate an array sum recursively
        #include <stdio.h>
        #include <stdlib.h>

        int sum(int arr[], int n)
        {
            if (n == 1) // base case; array length is 1
            {
                return arr[0];
            }
            // recursive case; add last element to the sum of the rest of the array
            return arr[n-1] + sum(arr, n-1);
        }

        int main()
        {
            int a[] = {1, 2, 3, 4, 5};

            printf("array sum: %d", sum(a, 5));
        }
```

```
array sum: 15
```

0.4 Program Arguments and `scanf()`

We can pass arguments into `main.c`; it will look like `int main( int argc, char *argv[] )`

- `argc` is the count of the program's arguments
- `argv` is an array of *character strings* representing the arguments themselves

Say we run `>> myprog left right centre` at the command line:

- `argc`: 4
- `argv`: {"myprog", "left", "right", "centre"}

Note that `argv[0]` will *always* be the name of the program (and is probably not super useful).

0.4.1 `scanf()`

- reads keyboard input
- accepts the same type specifiers as `printf()`

```
In [10]: #include <stdio.h>

         int main()
         {
             int a, b, c;
```

```
    printf("Enter the value of a:");
    scanf("%d", &a); // toss an int into a, pointerly

    printf("Enter the value of b:");
    scanf("%d", &b);

    printf("Enter the value of c:");
    scanf("%d", &c);

    printf("a, b, c: %d, %d, %d", a, b, c);
}
```