Six Stages of Debugging

1. That can’t happen.
2. That doesn’t happen on my machine.
3. Please don’t let that happen.
4. Why does that happen?
    a. The other guy’s code is buggy.
    b. The compiler is buggy.
5. Oh, I see.
6. How did that ever work?

Call Stacks +
On Writing Good Code

CMPT 125
Mo Chen
SFU Computing Science
22/1/2020

Lecture 8

Today:

- Function Call Stack
- Recursion
- Good Coding Principles

Stacks - a Brief Introduction

A *stack* is an ordered collection of items, to which you may insert an item (a *push*) or remove an item (a *pop*), where removal follows a last-in-first-out order (LIFO).

a stack of plates

a stack of books

a stack of pancakes

Function Calls

- Function calls & return values in LIFO order.
  - When a function completes, control returns to the function that called it.
- A function call is characterized by 4 things:
  - its parameters
  - its local vars
  - its return value
  - its return address
- All 4 things are maintained on the *call stack.*
  - Push / pop one *stack frame* per function call.

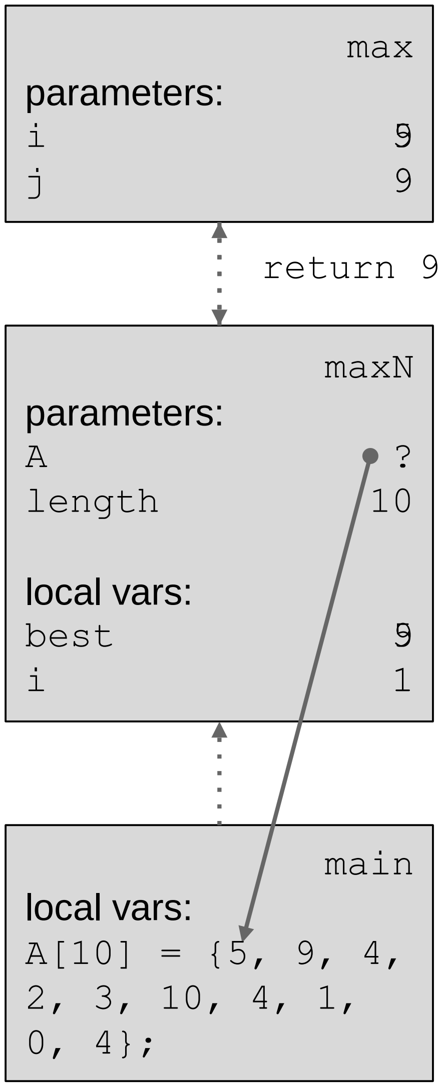
Remember that:
- parameters have local scope
- variables have local scope
- parameters are pass by value

Functions Calling Functions

```
int max(int i, int j) {
    if (i < j) { i = j; }
    return i;
}

int maxN(int A[], int length) {
    int best = A[0];
    for (int i = 1; i < length; i++) {
        best = max(best, A[i]);
    }
    return best;
}

int main () {
    int A[10] = {5, 9, 4, 2, 3, 10, 4, 1, 0, 4};
    printf("The highest was %d.\n", maxN(A, 10));
    return 0;
}
```



Recursive Functions

base cases

```
unsigned int fac(unsigned int n) {
    if (n <= 1) {
        return 1;
    }
    return n * fac(n-1);
}

int main () {
    printf("4! = %u\n", fac(4));
    return 0;
}
```

Factorial

$0! = 1$

$1! = 1$

$n! = n \times (n - 1)!$, when $n \geqslant 2$

recursive definition

`main( … )` is also a function!

- Running your program is the same thing as making a single function call to `main( … )`
  - main function “called” from command shell
  - return value passed to command shell

- `main` can take arguments
  - `int main(int argc, char *argv[]) { … }`
  - `argv[argc]` is an array of strings — the same sequence of strings you typed on the command line

Stack Variables

- Stack memory is sequential.
- Stack memory is recycled when function terminates.
  - don’t return pointers to recycled stack variables!
  - an important issue in dynamic memory allocation
- Variables on the stack cannot grow / shrink.
  - would have to move **everything** above it on the stack to make room!

Code serves two purposes

- Code is the precise expression of an algorithm to the computer.
  - follows instructions literally
- Code is the expression of an algorithm to another programmer.
  - concerned with the problem the algorithm tries to solve
  - “another programmer” might be a future you!

Coding Style - Making It Easy to Read!

- Comments in C: /* *block* */ OR // *inline*
  - block comments for: pre- / post-conditions, expected behaviours, revision documentation
  - inline comments for: assertions, and / or a high-level description of algorithm, perhaps at a pseudocode level
- Variable naming
  - choose names to help with understanding of code
  - naming conventions vary between codeshops
- Whitespace
  - indentation, blank lines
  - expression formatting

Remember This Slide?

```
int range(int A[], int n) {
     int lo = min(A, n);
     int hi = max(A, n);
     return hi-lo;
}

int range(int list[], int list_length) {
     int lowest = minN(list, list_length);
     int highest = maxN(list, list_length);
     return highest-lowest;
}
```

What does this do?

```
int f(int n) {
    int p = 1;
    while(n) {
        p = p * n;
        n--;
    }
    return p;
}
```

```
// compute and return n!
int factorial(int n) {
    int product = 1;
    while(n > 0) {
        product *= n;
        n--;
    }
    return product;
}
```