CS32 Intro to Computer Science II

Baoxiong Jia & Muthu Palaniappan, DIS 1C Week 5
UCLA Spring 2021

About Us

- TA: Baoxiong Jia
  - Email: baoxiongjia@cs.ucla.edu
  - Office Hours: Tuesday 8:30-10:30am
  - Thursday 8:30-10:30am
  - Discussion 1C: Friday 12:00-13:50pm

- LA: Muthu Palaniappan
  - Email: muthupal@g.ucla.edu
  - Office Hours: Monday 10:30-11:30am
  - Wednesday 10:30-11:30am

Outline

- Inheritance
- Polymorphism
- Recursion

Inheritance

1. Construction order:
    a. Construct the base part, consulting the member initialization list
    (If not mentioned there, use base class's default constructor).
    b. Construct the data members, consulting the member initialization list.
    (If not mentioned there, use member's default constructor if it’s of a class type, else leave uninitialized.)
    c. Execute the body of the constructor.
2. Destruction order:
    a. Execute the body of the destructor.
    b. Destroy the data members (doing nothing for members of builtin types).
    c. Destroy the base part.

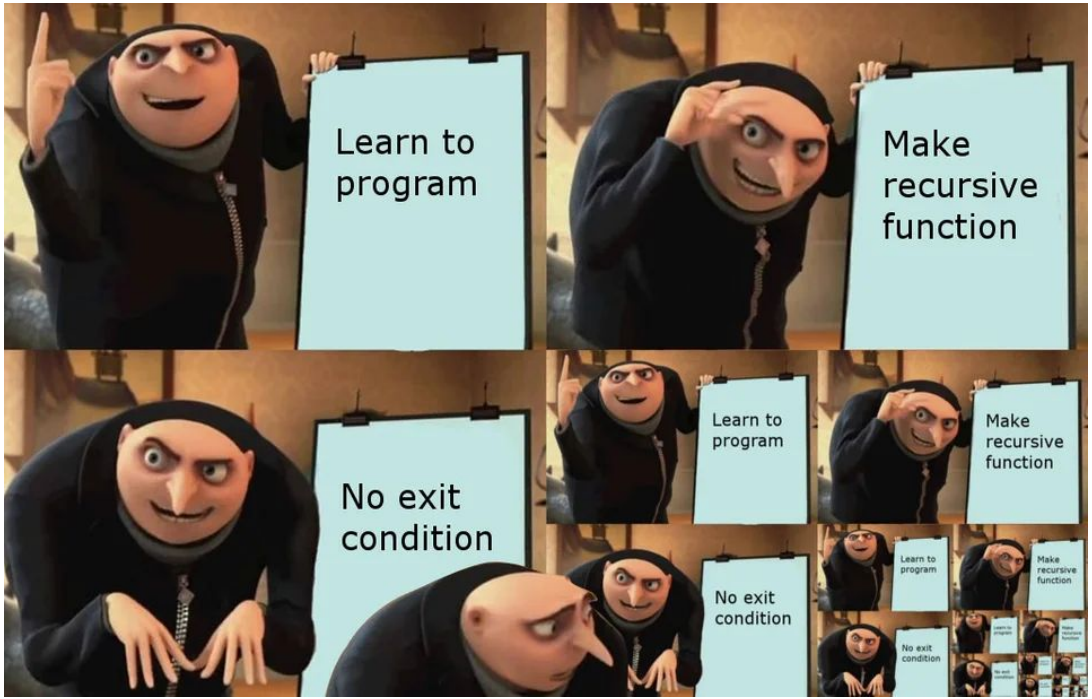
https://repl.it/@jiajerry/InheritanceOrder

Polymorphism

- The action of using a Base pointer/reference to access any variables whose types are derived from the Base class.
- A call to a member function will cause a different function to be executed depending on the type of object that invokes the function.

https://repl.it/@jiajerry/Polymorphism

Recursion


Learn to program
Make recursive function
No exit condition
Learn to program
Make recursive function
No exit condition
Learn to program
Make recursive function
No exit condition

Recursion

Recursive functions:

{

0. base case // a method to deal with the smallest problem

...

return;

1. recursive case // a method to deal with a smaller problem than the original one

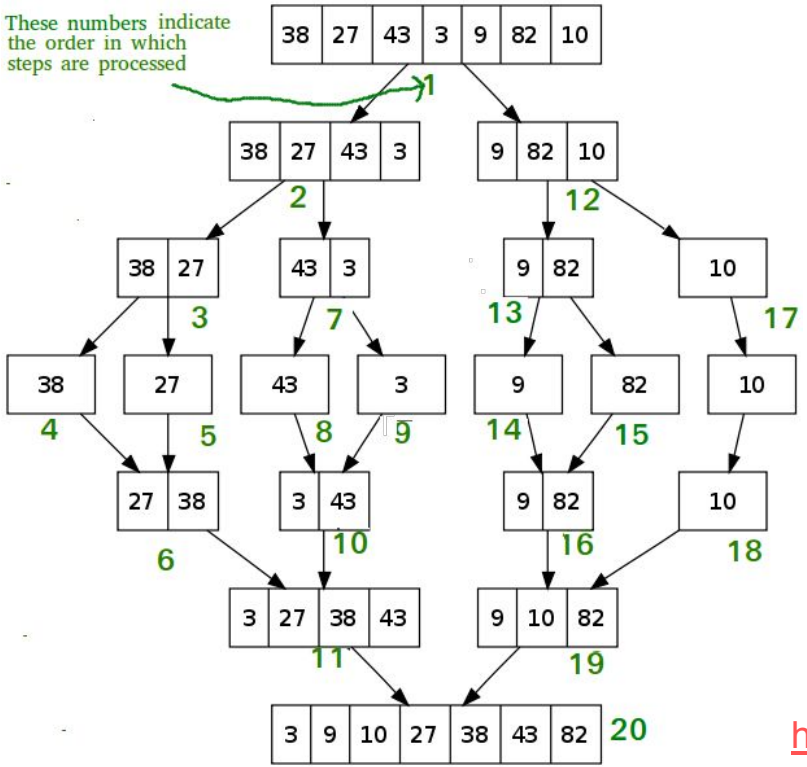
{

1. collect the results from the subproblem, by calling the function again, passing parameters that are a subset of the original parameter, if necessary.
2. process the results collected from the subproblem, if necessary
3. process the values in current function call (myself), if necessary
4. return the processed result if the super problem requires any, or simply return to finish the current function call

}

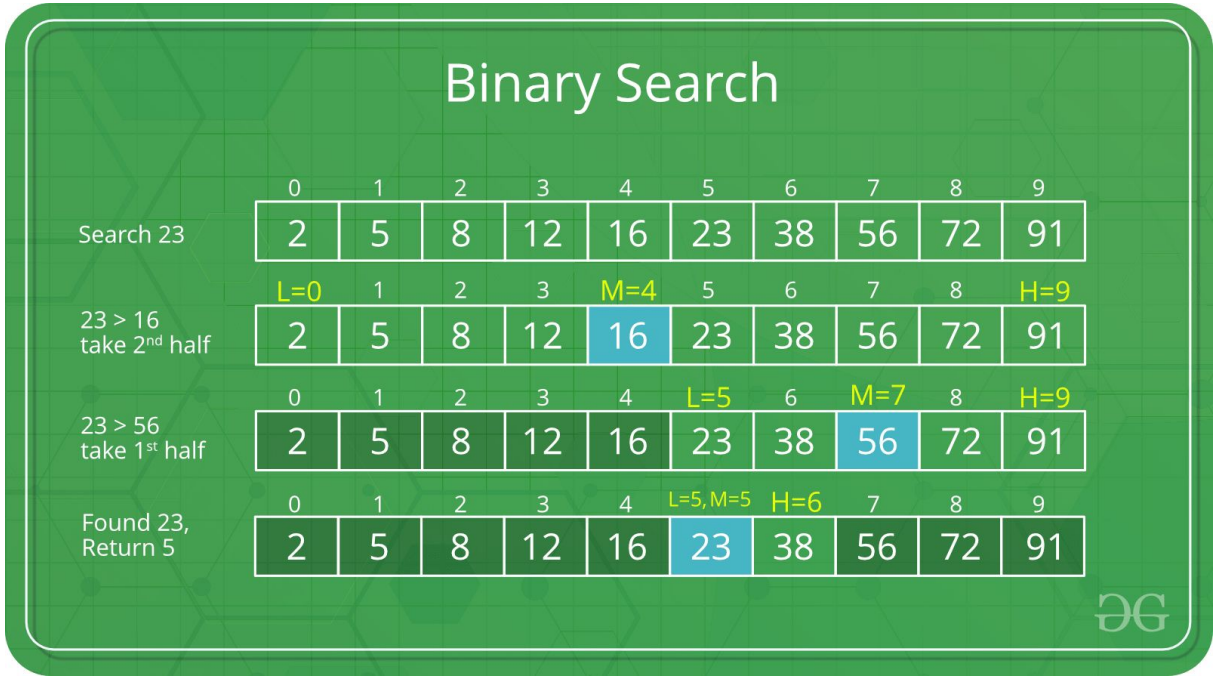
}

Example: Merge Sort



https://repl.it/@jiajerry/MergeSort

Example: Binary Search



https://repl.it/@jiajerry/BinarySearch

Example: Fibonacci Sequence

The Fibonacci numbers, commonly denoted F(n) form a sequence, called the Fibonacci sequence, such that each number is the sum of the two preceding ones, starting from 1. That is,

```
fib(0) = 0, fib(1) = 1
fib(N) = fib(N - 1) + fib(N - 2), for N > 1.

Given N, calculate fib(N).

Input: 4
Output: 3
Explanation: fib(4) = fib(3) + fib(2) = 2 + 1 = 3.
```

https://repl.it/@jiajerry/Fibonacci

Example: Power Operation

Given two positive integer a and b, calculate a^b (a to the power of b).

power(a,b)

for example:
intput: a = 3, b = 8;
output: 6561

https://repl.it/@jiajerry/Exponential

Example: Alternating Print

```
Given an array, write a recursive function that prints the array, alternating from each end.
For example, given
int x [10] = {1,2,3,4,5,6,7,8,9,10}

alternatingPrint(x, 10)

should output:
1 10 2 9 3 8 4 7 5 6
```

https://repl.it/@jiajerry/AlternatingPrint

Example: Sum of Linked List

```
Given a linked list of integers, return the sum of each node

Example:

Input: 9 -> 10 -> 5 -> nullptr
Output: 24
```

https://repl.it/@jiajerry/SumOfLinkedList

Example: Triple Step

A child is running up a staircase with n steps and can hop either 1 step, 2 steps, or 3 steps at a time. Implement a method to count how many possible ways the child can run up the stairs.

f(1) = 1

f(2) = 2

f(3) = 4

https://repl.it/@jiajerry/TripleStep

Example: Sum of Digits

Implement the function `sumOfDigits` recursively. The function should return the sum of all of the digits in a *positive* integer.

```
int sumOfDigits(int num);

sumOfDigits(176); // should return 14
sumOfDigits(111111); // should return 6
```

https://repl.it/@jiajerry/SumOfDigits

Example: isSolvable

Implement the following function in a recursive fashion:

```
bool isSolvable(int x, int y, int c);
```

This function should return true if there exists nonnegative integers *a* and *b* such that the equation $ax + by = c$ holds true. It should return false otherwise.

Ex: `isSolvable(7, 5, 45) == true //a == 5 and b == 2`
Ex: `isSolvable(1, 3, 40) == true //a == 40 and b == 0`
Ex: `isSolvable(9, 23, 112) == false`

https://repl.it/@jiajerry/isSolvable

Example: isPalindrome

Implement the function `isPalindrome` recursively. The function should return whether the given string is a palindrome. A palindrome is described as a word, phrase or sequence of characters that reads the same forward and backwards.

```
bool isPalindrome(string foo);

isPalindrome("kayak"); // true
isPalindrome("stanley yelnats"); // true
isPalindrome("LAs rock"); // false (but the sentiment is true :))
```

https://repl.it/@jiajerry/Palindrome