Stepwise Refinement

The Situation

- Programming from scratch…
- How can we approach this?
  - Where do we start?
  - What functions will we need?
  - How to structure the program?
  - …
- Stepwise Refinement → **stepwise approach**

Stepwise Refinement - Idea

- From coarse-grained to fine-grained.

Procedure

1. **Outline**
   coarse-grained structure using comments

2. **Refine**
   comments repeatedly with
   - finer-grained comments
   - code
   - (hypothetical) function calls

Example

Example

Goal:

Write a simple program which computes the mid-point between two points on a line.

Example

Coarse-grained structure

```
int main () {
    // input the two points

    // compute mid-point

    // output mid-point
}
```

Example

Refine

```
int main () {
    // input the two points

    // compute mid-point

    // output mid-point
}
```

Example

Refine

```
int main () {
    // Input of values
    std::cout << "Input the first point: ";
    double a;
    std::cin >> a;

    std::cout << "Input the second point: ";
    double b;
    std::cin >> b;

    // compute mid-point

    // output mid-point
}
```

Example

Refine

```
int main () {
    // Input of values
    std::cout << "Input the first point: ";
    double a;
    std::cin >> a;

    std::cout << "Input the second point: ";
    double b;
    std::cin >> b;

    // compute mid-point

    // output mid-point
}
```

Example

Refine

```
int main () {
    // Input of values
    std::cout << "Input the first point: ";
    double a;
    std::cin >> a;

    std::cout << "Input the second point: ";
    double b;
    std::cin >> b;

    // Computation of mid-point
    double m = mid_point(a, b);

    // output mid-point
}
```

Example

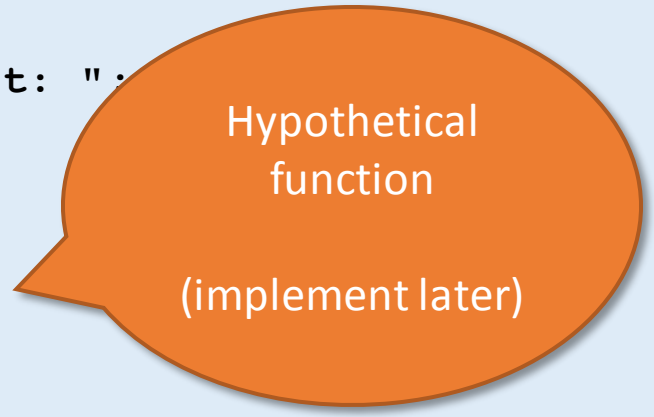
Refine

```
int main () {
    // Input of values
    std::cout << "Input the first point: ";
    double a;
    std::cin >> a;

    std::cout << "Input the second point: ";
    double b;
    std::cin >> b;

    // Computation of mid-point
    double m = mid_point(a, b);

    // output mid-point
}
```



Example

Refine

```
int main () {
    // Input of values
    std::cout << "Input the first point: ";
    double a;
    std::cin >> a;

    std::cout << "Input the second point: ";
    double b;
    std::cin >> b;

    // Computation of mid-point
    double m = mid_point(a, b);

    // output mid-point
}
```

Example

Refine

```
int main () {
    // Input of values
    std::cout << "Input the first point: ";
    double a;
    std::cin >> a;

    std::cout << "Input the second point: ";
    double b;
    std::cin >> b;

    // Computation of mid-point
    double m = mid_point(a, b);

    // Output of computed result
    std::cout << "The mid-point is: " << m << "\n";
}
```

Example

Cleanup

```
int main () {
    // Input of values
    std::cout << "Input the first point: ";
    double a;
    std::cin >> a;

    std::cout << "Input the second point: ";
    double b;
    std::cin >> b;

    // Computation and output
    std::cout << "The mid-point is: " << mid_point(a, b)
              << "\n";
}
```

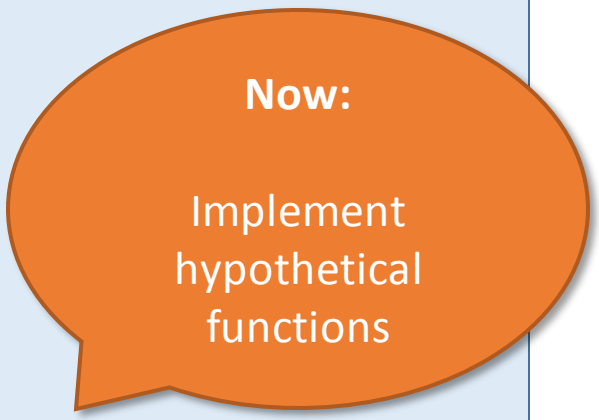
Example

Cleanup

```
int main () {
    // Input of values
    std::cout << "Input the first point: ";
    double a;
    std::cin >> a;

    std::cout << "Input the second point: ";
    double b;
    std::cin >> b;

    // Computation and output
    std::cout << "The mid-point is: " << mid_point(a, b)
              << "\n";
}
```



Example

Cleanup

Using Stepwise Refinement

```
int main () {
    // Input of valu
    std::cout << "Inp                    ";
    double a;
    std::cin >> a;

    std::cout << "Input the second point: ";
    double b;
    std::cin >> b;

    // Computation and output
    std::cout << "The mid-point is: " << mid_point(a, b)
              << "\n";
}
```

Now:

Implement hypothetical functions

```
mid_point(a,b)
```

Example

Coarse-grained structure

```
// POST: returns the mid-point between a and b.
double mid_point (double a, double b) {

    // compute mid-point

    // return
}
```

Example

Refine

```
// POST: returns the mid-point between a and b.
double mid_point (double a, double b) {

    // compute mid-point

    // return
}
```

Example

Refine

```
// POST: returns the mid-point between a and b.
double mid_point (double a, double b) {

    double mid_point = (a + b) / 2;

    // return
}
```

Example

Refine

```
// POST: returns the mid-point between a and b.
double mid_point (double a, double b) {

    double mid_point = (a + b) / 2;

    // return
}
```

Example

Refine

```
// POST: returns the mid-point between a and b.
double mid_point (double a, double b) {

    double mid_point = (a + b) / 2;

    return mid_point;
}
```

Example

Cleanup

```
// POST: returns the mid-point between a and b.
double mid_point (double a, double b) {
    return (a + b) / 2;
}
```