

# Computer Programming

Dr. Deepak B Phatak

Dr. Supratik Chakraborty

Department of Computer Science and Engineering  
IIT Bombay

Session: Iteration Idioms: Motivation

# Quick Recap of Relevant Topics

---



- Structure of a simple C++ program
- Sequential execution of statements
- Conditional execution of statements
- Programming to solve simple problems

# Overview of This Lecture

---



- Need for iteration in programming
  - Convenience
  - Necessity
  - Intuitive programming
- Generic iteration construct
- Iteration constructs in C++

# A Simple Problem

---



Read quiz 1 marks of ten CS101 students and print their sum, average, maximum and minimum

Can we solve using what we've learnt so far?

assignment statements, input/output statements

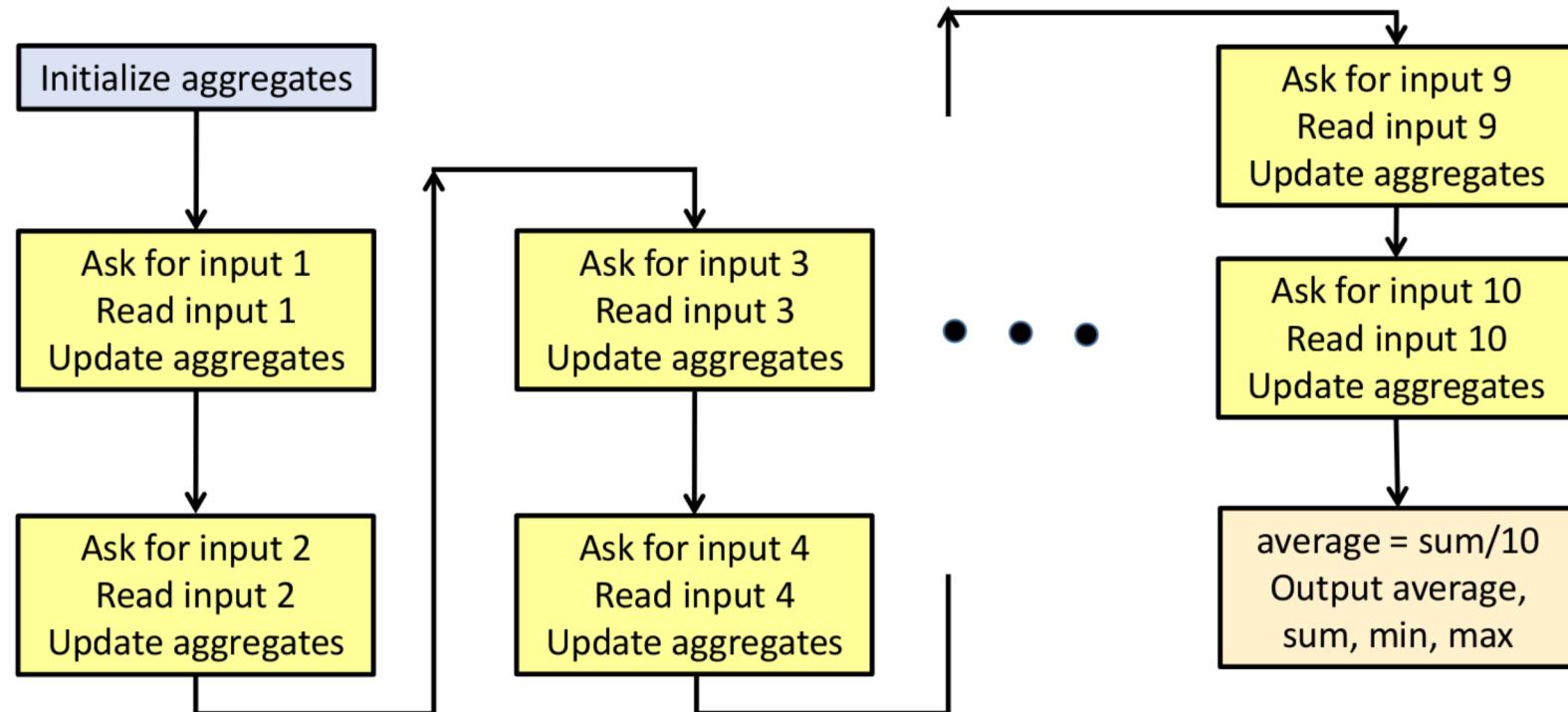
arithmetic expressions

sequential and conditional execution of statements

# Overall Strategy

- Maintain “running” sum, max and min (aggregates)
- Initialize aggregates
- Read input 1 and update aggregates
- Read input 2 and update aggregates
- ...
- Read input 10 and update aggregates
- Compute average as sum/10
- Output sum, average, max, min

# A Simple Flowchart



# C++ Program

---



```
int main() {  
    // Variable declarations  
    int marks, sum, min, max;  
    float average;  
    // Initialization of aggregates  
    sum = 0; average = 0;  
  
    // Further code comes here  
    return 0;  
}
```

# C++ Program

---



```
int main() {  
    // Variable declarations and initialization of sum and average  
  
    cout << "Give quiz marks of student 1: ";  
    cin >> marks;  
    sum = sum + marks;  
    // Initialize min and max with first input  
    min = marks; max = marks;  
    // Further code comes here  
    return 0;  
}
```

# C++ Program

---



```
int main() {  
    // Variable declarations and initialization of sum and average  
    // Read marks of student 1, and update aggregates  
    cout << "Give quiz marks of student 2: ";  
    cin >> marks;  
    sum = sum + marks;  
    min = (min > marks) ? marks: min;  
    max = (max < marks) ? marks: max;  
    // Further code comes here  
    return 0;  
}
```

# C++ Program

---



```
int main() {  
    // Variable declarations and initialization of sum and average  
    // Read marks of students 1 and 2, and update aggregates  
    cout << "Give quiz marks of student 3: ";  
    cin >> marks;  
    sum = sum + marks;  
    min = (min > marks) ? marks: min;  
    max = (max < marks) ? marks: max;  
    // Further code comes here  
    return 0;  
}
```

# C++ Program

---

```
int main() {  
    // Variable declarations and initialization of sum and average  
    // Read marks of students1, 2 ...10, and update aggregates  
    // Calculate and print aggregates  
    average = sum/10.0;  
    cout <<“Average: “ <<average <<“Sum: “ <<sum;  
    cout <<“Min: “ <<min <<“Max: “ <<max <<endl;  
    return 0;  
}
```

# Some Observations

- (Almost) same instructions repeated multiple times

```
cout << "Give marks of student 3: ";
```

```
cin >> marks;
```

```
sum = sum + marks;
```

```
min = (min > marks) ? marks : min;
```

```
max = (max < marks) ? marks : max;
```

Slightly different for student 1:  
min = marks; max = marks;

- Intuitively, we would like to execute (almost) the same instructions for all students

# Some Observations

---



- Suppose we had a construct in C++ that allowed us to effectively say  
**Repeat the following block of instructions a specified number of times**
- Could we write a less repetitive C++ program to aggregate quiz 1 marks ?

# Another Attempt At Our C++ Program



```
int main() {  
    // Variable declarations and initialization of aggregates  
    int count = 1;  
    // Repeat the following block of code 10 times  
    { cout << "Give marks of student " << count << ":";  
        cin >> marks;  
        sum =sum +marks;  
        // Update min and max appropriately  
        count =count +1;  
    } // End of block of code to be repeated  
    // Code for computing average and printing comes here  
    return 0;  
}
```

# Another Attempt At Our C++ Program



```
int main() {  
    // Variable declarations and initialization of aggregates  
    int count = 1;  
    // Repeat the following block of code 10 times  
    { cout << "Give marks of student " << count << ":"; Updating min and max  
        cin >> marks;  
        sum =sum +marks;  
        if (count ==1) {min =marks; max =marks; }  
        else {min =(min >marks) ? marks:min; max =(max <marks) ? marks: max; }  
        count =count +1;  
    } // End of block of code to be repeated  
    // Code for computing average and printing comes here  
    return 0;  
}
```

# Another Attempt At Our C++ Program

```
int main() {  
    // Variable declarations and initialization of aggregates  
    int count = 1;  
    // Repeat the following block of code 10 times  
    { cout << "Give marks of student " << count << ":";  
        cin >> marks;  
        sum =sum +marks;  
        if (count ==1) {min =marks; max =marks; }  
        else {min =(min >marks) ? marks:min; max =(max <marks) ? marks: max; }  
        count =count +1;  
    } // End of block of code to be repeated  
    // Code for computing average and printing comes here  
    return 0;  
}
```

# Another Attempt At Our C++ Program

```
int main() {  
    // Variable  
    int count = 0;  
    // Repeat till user enters -1  
    { cout << "Give marks of student " << count << ". ".  
        cin >> mark;  
        sum = sum + mark;  
        if (count == 0) min = max = mark;  
        else { min = min < mark ? min : mark;  
               max = max > mark ? max : mark; }  
        count = count + 1;  
    } // End of block of code to be repeated  
    // Code for computing average and printing comes here  
    return 0;  
}
```

# Repetition in Programming



- Wasteful
  - If you can achieve something by coding once, why code again?
- Potential source of bugs and inconsistencies
  - Afterthought: Want to say “Thank you” after each marks is read
  - `cout <<“Thank you”;` at 10 places
  - What if there was a typo (“**Think yoo**”) at one place?
  - Can be more dangerous than just a message being printed wrong
- Maintainability of large code with repetition difficult
  - Small change in replicated code requires replicating change at several places
- **Reuse as much code as possible, avoid repetitions consciously**

# More General Iteration

```
int main() {  
    // Variable declarations and initialization of aggregates  
    int count = 1;  
    // Repeat the following block of code 10 times  
    { cout << "Give marks of student " << count << ":";  
        cin >> marks;  
        sum = sum + marks;  
        if (count == 1) { min = marks; max = marks; }  
        else { min = (min > marks) ? marks:min; max = (max < marks) ? marks: max; }  
        count = count + 1;  
    } // End of block of code to be repeated  
    // Code for computing average and printing comes here  
    return 0;  
}
```

What if we want to aggregate marks of “n” students, where “n” is user specified?

Number of repetitions cannot be determined when writing program

# More General Iteration

```
int main() {  
    // Variable declarations and initialization of aggregates  
    int count = 1;  
    // Repeat the following block of code 10 times  
    { cout << "Give marks of student " << count << ":";  
        cin >> marks;  
  
        // End of block of code to be repeated  
    }  
    // Code for computing average and printing comes here  
    return 0;  
}
```

What if we want to aggregate marks of “n” students, where “n” is user specified?

Necessity of repetition/iteration construct:  
Problem cannot be solved without this

marks: max; }

Number of repetitions cannot be determined when writing program

# More General Iteration

```
int numStudents, count;
cout << "Give number of students in CS101: "; cin >> numStudents
count = 1;
```

**// Repeat the following block of code while (count <= numStudents)**

```
{ cout << "Give marks of student " << count << ": ";
  cin >> marks;
  sum = sum + marks;
  if (count == 1) { min = marks; max = marks; }
  else { min = (min > marks) ? marks:min; max = (max < marks) ? marks: max; }
  count = count + 1;
} // End of block of code to be repeated
```

Iterate while a logical condition is satisfied

Crucial:  
Affects logical condition for loop termination

**// Code to compute aggregates and print them**

# A Generic Iteration Construct



- General structure of program with iteration

Part of program before iteration

Iteration initialization (setting up initial values, etc)

Iterate/Repeat as long as a logical condition stays true

{

Block of statements

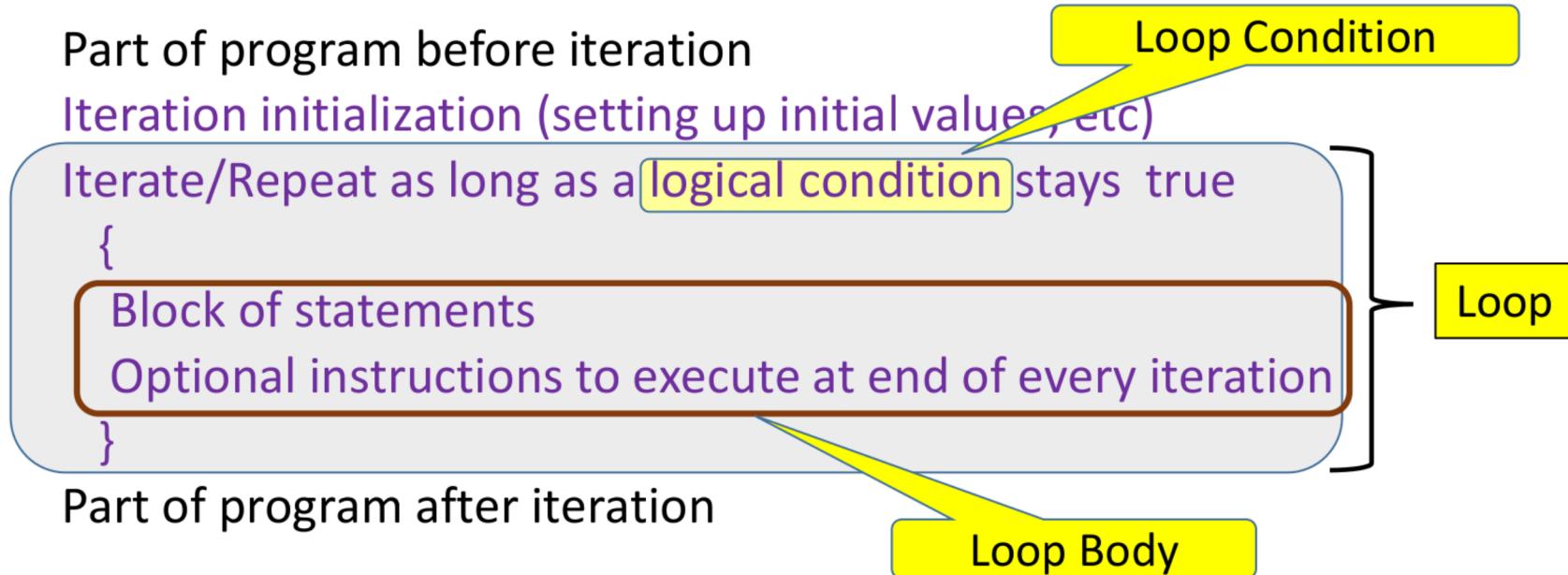
Optional instructions to execute at end of every iteration

}

Part of program after iteration

# A Generic Iteration Construct

- General structure of program with iteration



# C++ Iteration Constructs



# Summary

---

- Iteration idioms in programming
  - Necessary in general
  - Convenient to write intuitive code
  - Enables code reuse, avoids pitfalls of repetition
- Glimpse of iteration constructs in C++