Computer Programming

Dr. Deepak B Phatak
Dr. Supratik Chakraborty
Department of Computer Science and Engineering
IIT Bombay

Session: “while” and “do while” statements in C++

Quick Recap of Relevant Topics

- Iteration idioms in programming
- Necessity and convenience of iteration
- Glimpse of iteration constructs in C++

Overview of This Lecture

- Iteration using "while" and "do … while" statements in C++
- "break" statement in loops

Recall Generic Iteration Construct

IIT Bombay

Loop Condition

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

}

Loop

Part of program after iteration

Loop Body

“while” Statement in C++

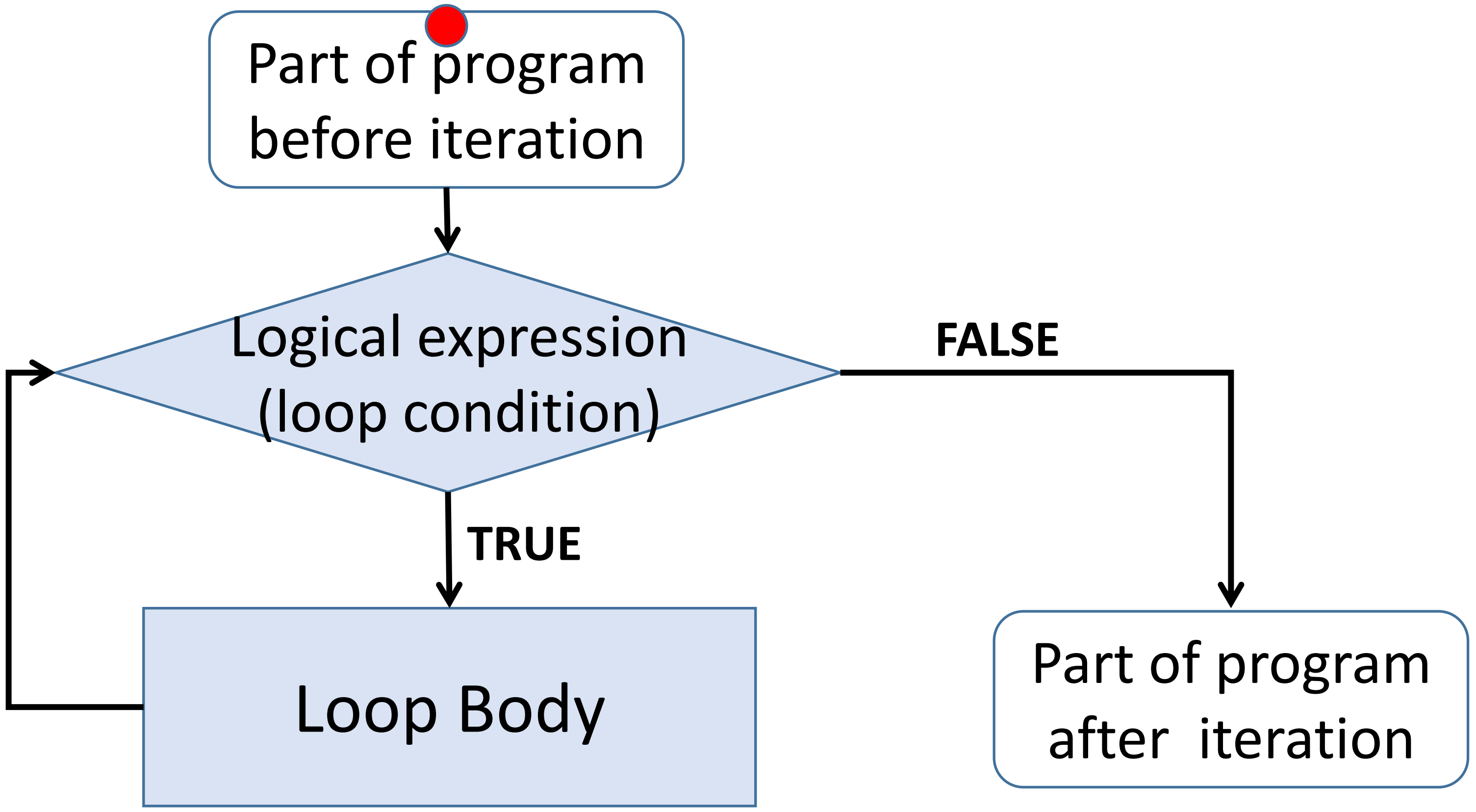
Part of program before execution

```
while (loop condition)
 {

    Block of statements  (Body of “while” loop)

 }
```

Part of program after iteration

Flowchart Representation of “while”

IIT Bombay


Part of program before iteration
Logical expression (loop condition)
FALSE
TRUE
Loop Body
Part of program after iteration

Points To Remember About “while”

while (loop condition) { Loop Body }

- Loop condition checked **before** executing loop body

  Can lead to zero executions of loop body
- Number of times loop condition is checked =

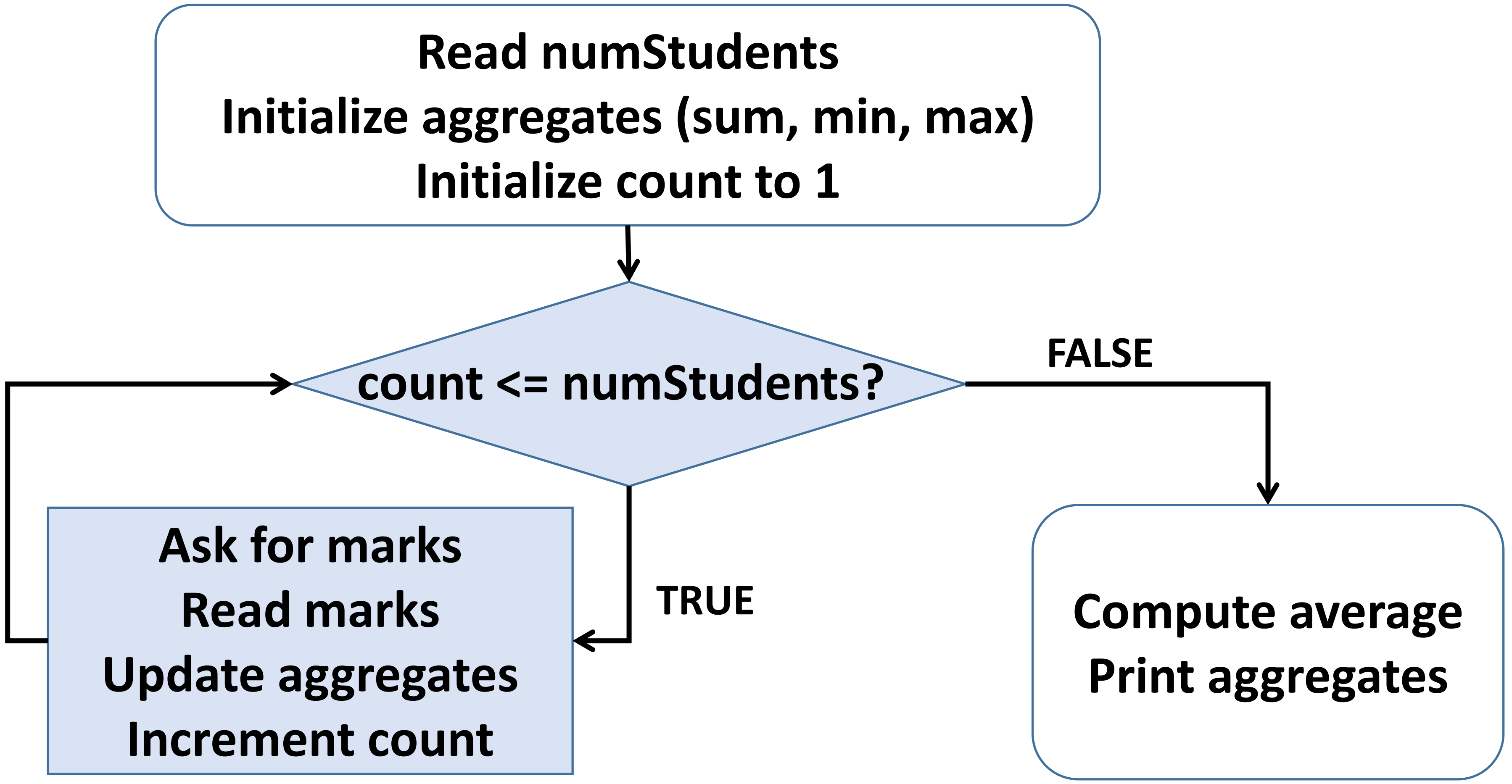
  Number of times loop body executed + 1, if loop terminates
- If loop condition is not changed in loop body, inifinite loop (non-terminating program) possible

Back To Our Problem

Read number of students in CS101, read quiz 1 marks of all CS101 students and print their sum, average, maximum and minimum

Flowchart Representation

IIT Bombay



C++ program with “while”

IIT Bombay

```
int main() {
   int marks, sum = 0, min, max, numStudents;
   float average, count; // Variable declarations
   cout << “Give number of students: “;    cin >> numStudents;
   count = 1.0;  // Count of student marks processed
   while (count <= numStudents) {
      cout << “Give marks of student “ << count << “: “;    cin >> marks;
      // Update sum, max, min
      count = count + 1;
   }
   average = sum/count;
   // Print  average, sum, min, max
   return 0;
}
```

C++ program with “while”

IIT Bombay

```
int main() {
   int marks, sum = 0, min, max, numStudents;
   float average, count;    // Variable declarations
   cout << “Give number of students: “;    cin >> numStudents;
   count = 1.0; // Count of student marks processed
   while (count <= numStudents) {
      cout << “Give marks of student “ << count << “: “;    cin >> marks;
      // Update sum, max, min
      count = count + 1;
   }
   average = sum/count;
   // Print  average, sum, min, max
   return 0;
}
```

C++ program with “while”

IIT Bombay

```
int main() {
   int marks, sum = 0, min, max, numStudents;
   float average, count;    // Variable declarations
   cout << “Give number of students: “;    cin >> numStudents;
   count = 1.0; // Count of student marks processed
   while (count <= numStudents) {
      cout << “Give marks of student “ << count << “: “;    cin >> marks;
      // Update sum, max, min
      count = count + 1;
   }
   average = sum/count;
   // Print  average, sum, min, max
   return 0;
}
```

C++ program with “while”

IIT Bombay

```
int main() {
   int marks, sum = 0, min, max, num
   float average, count;    // Variable
   cout << “Give number of students:
   count = 1.0; // Count of student m
   while (count <= numStudents) {
      cout << “Give marks of student “ << count << “: “;    cin >> marks;
      // Update sum, max, min
      count = count + 1;
   }
   average = sum/count;
   // Print  average, sum, min, max
   return 0;
}
```

```
sum = sum + marks;
if (count == 1) { min = marks; max = marks; }
else {
   min = (min > marks) ? marks: min;
   max = (max < marks) ? marks: max;
}
```

C++ program with “while”

IIT Bombay

```
int main() {
   int marks, sum = 0, min, max, numStudents;
   float average, count;    // Variable declarations
   cout << “Give number of students: “;    cin >> numStudents;
   count = 1.0; // Count of student marks processed
   while (count <= numStudents) {
      cout << “Give marks of student “ << count << “: “;    cin >> marks;
      // Update sum, max, min
      count = count + 1;
   }
   average = sum/count;
   // Print  average, sum, min, max
   return 0;
}
```

Accumulation or Aggregation in Loops

IIT Bombay

```
int main() {
   int marks, sum = 0, min, max
   float average, count;    // Va
   cout << "Give number of stu
   count = 1.0; // Count of stud
   while (count <= numStudent
      cout << "Give marks of stu
      // Update sum, max, min
      count = count + 1;
   }
   average = sum/count;
   // Print  average, sum, min,
   return 0;
}
```

Inputs were provided one after another: "streaming" inputs

We did not remember all inputs seen so far, only some aggregates

Aggregates: "summary" of streaming inputs seen so far so that we can compute final result

Accumulation or aggregation: key to programming with loops for streaming inputs

A Variant Of Our Problem

IIT Bombay

Read quiz 1 marks of CS101 students one at a time

Stop reading if -1000 is entered as marks

Print number of marks entered, sum, average, maximum and minimum

- Difference from earlier version:

  We do not know a priori how many marks will be entered

  Indicated by special end-of-inputs marks (-1000)

  We'll know when to stop only after reading -1000 as marks

Modifying Our Earlier C++ program

IIT Bombay

```
int main() {
   int marks, sum = 0, min, max;
   float average, count;    // Variable declarations
   count = 1.0; // Count of student marks processed
   while (true) {
      cout << “Give marks of student “ << count << “: “;    cin >> marks;
      if (marks == -1000) { … exit loop … }
      else {  … Update sum, max, min … }
      count = count + 1;
   }
   average = sum/(count – 1);
   // Print  count – 1, average, sum, min, max
   return 0;
}
```

Infinite loop !!!

C++ provides an easy way to do this

"break" Statement In "while" Loop

IIT Bombay

```
while (true) {
    cout << "Give marks of student " << count << ": ";
    cin >> marks;
    if (marks == -1000) { break; }
    else {  … Update sum, max, min … }
    count = count + 1;
 }
```

Recall "break" from "switch … case … "

Can We Do Without “break”?

```
bool exitFlag = false;
while (! exitFlag) {
     cout << “Give marks of student “ << count << “: “;
     cin >> marks;
     if (marks == -1000) {exitFlag = true;}
     else {
          … Update sum, max, min …
          count = count + 1;
     }
}
```

Include within “else” block to preserve behaviour of program with “break”

Convenience Of “break” In Loops

```
while (true) {
      cout << “Give marks of student “ << count << “: “;
      cin >> marks;
      if (marks == -1000) { break; }
      else {  … Update sum, max, min … }
      count = count + 1;
  }
```

“break” avoids such annoying complications

Recap: “while” Statement in C++

Part of program before execution

```
while (loop condition)
 {

    Block of statements  (Body of “while” loop)

 }
```

Part of program after iteration

“do … while …” Statement in C++

IIT Bombay

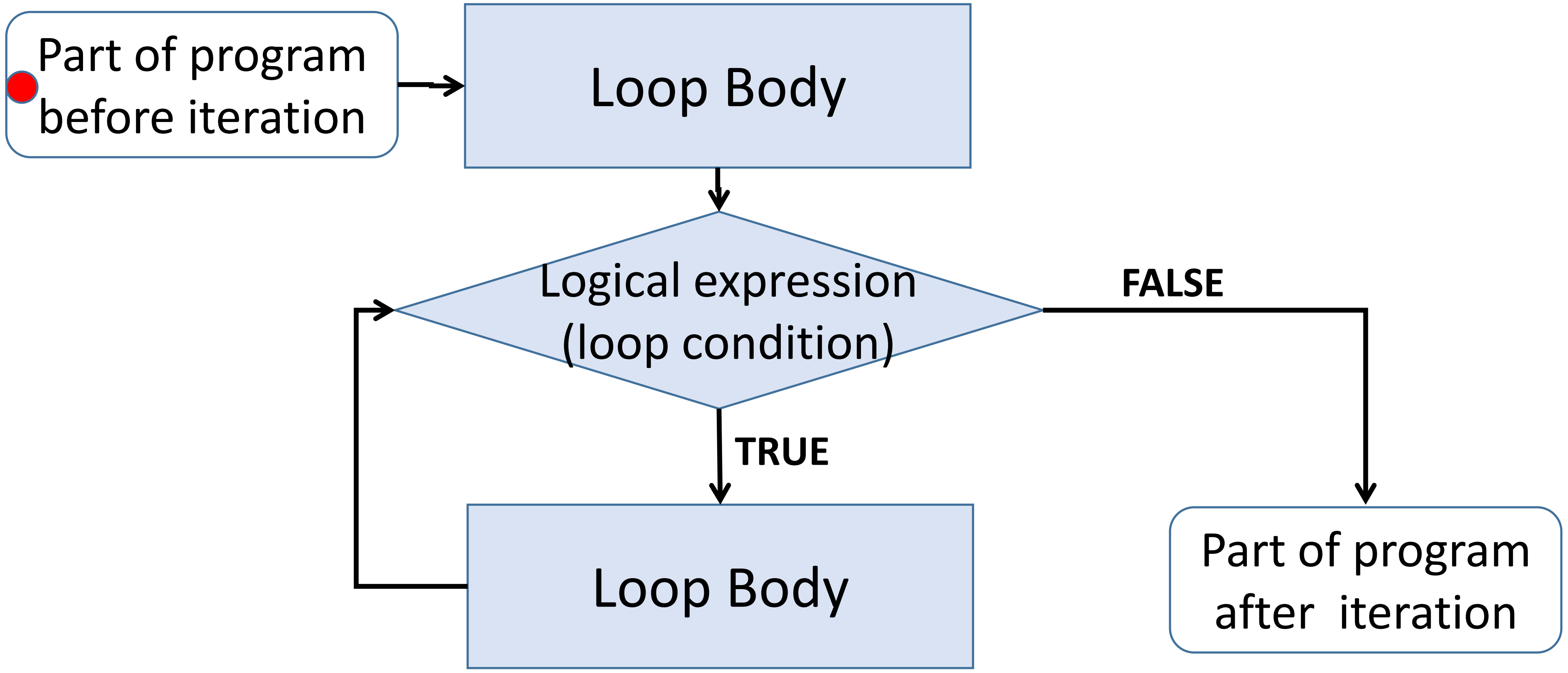
Part of program before execution

```
do
 {

    Block of statements  (Body of “do-while” loop)

 }  while (loop condition)
```

Part of program after iteration

“do … while …” Statement Flowchart

IIT Bombay


Part of program before iteration
Loop Body
Logical expression (loop condition)
FALSE
TRUE
Loop Body
Part of program after iteration

From “while …” to “do … while …”

```
while (loop condition) {
   Loop Body
}
```

```
if (loop condition) {
   do {
         Loop Body
   } while (loop condition);
}
```

From “do … while …” to “while …”

IIT Bombay

```
Loop Body;
while (loop condition) {
   Loop Body
}
```

```
do {
        Loop Body
 } while (loop condition);
```

“break” statements can be used in “do … while …” in same manner as in “while …”

"do … while …" vs "while …"

IIT Bombay

- Almost the same
- Prefer "do … while …" when we are guaranteed to execute loop body at least once
- Prefer "while …" if loop body may not be executed at all
- Programmer's choice

Summary

IIT Bombay

- “while” statement in C++
- “do … while …” statement in C++
- Use of “break” statements