An Introduction to Programming though C++

Abhiram G. Ranade

Lecture Sequence 3.2

Ch. 7: Loops

Mark averaging

"Read marks of students from the keyboard and print the average."

- Number of students not given explicitly.
- If a negative number is entered as mark, then it is a signal that all marks have been entered.
- Assume at least one positive number is given.

Examples:

- Input: 98 96 -1,   Output: 97
- Input: 90 80 70 60 -1,   Output: 75

- Cannot be done using what you know so far.
  - ***repeat*** repeats fixed number of times. Not useful
- Need new statement e.g. ***while, do while***, or ***for***.
- ***repeat, while, do while, for*** : "looping statements".

Outline

- The while statement
  - Some simple examples
  - Mark averaging
- The break statement
- The continue statement
- The do while statement
- The for statement
- Loop invariants
  - GCD using Euclid’s algorithm

The while statement

Form: ***while (condition) body***

1. Evaluate ***condition***.
2. If ***false***, execution of statement ends.
3. If ***true***, execute ***body***. ***body*** can be a single statement or a block, in which case all the statements in the block will be executed.
4. Go back and execute from step 1.

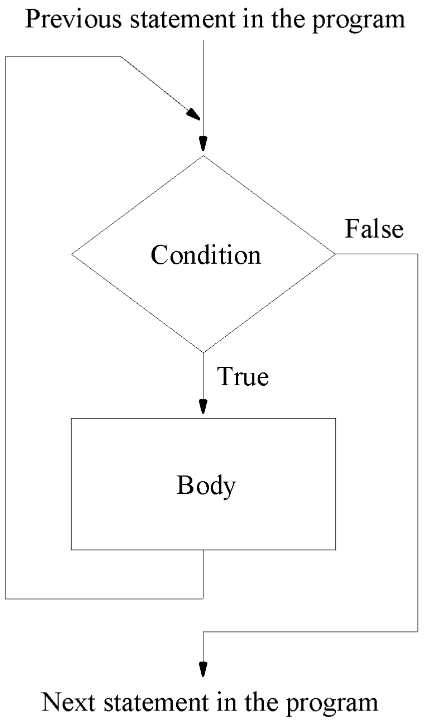
- The ***condition*** must eventually become ***false***, otherwise the program will never halt. Not halting is not acceptable.
- If ***condition*** is ***true*** originally, then value of some variable in ***condition*** must change in the execution of ***body***, so that eventually ***condition*** becomes ***false***.
- Each execution of the body = iteration.

While flowchart

...previous statement...

while(condition) body

...Next statements...


Previous statement in the program
Condition
False
True
Body
Next statement in the program

A silly example

```
main_program{
    int x=2;
    while(x > 0){
        x--;
        cout << x << endl;
    }
  cout <<"Done."<<endl;
}
```

- First x=2 is executed.
- Next, x > 0 is checked
- x=2 is > 0, so body entered.
- x is decremented, becomes 1.
- x is printed. (1)
- Back to top of loop.
- x=1 is > 0, body entered.
- x is decremented, becomes 0.
- x is printed. (0)
- Back to top of loop.
- x=0 is not > 0. body not entered.
- “Done.” printed

while vs. repeat

- Anything you can do using repeat can be done using while.

```
repeat(n){ xxx }
```

- Equivalent to

```
int i=n;
while(i>0){i--; xxx }
```

Assumption: the name i is not used elsewhere in the program.

- If it is, pick a different name

Exercise

- What will the following program fragment print?

```
int x=306, y=77, z=0;
while(x > 0){
  z = z + y;
  x--;
}
cout << z << endl;
```

- Write the following using **while.**

```
repeat(4){
  forward(100);
  right(90);
}
```

What we discussed

When to use ***while*** statement:

- We need to iterate while a condition holds
- We need not know at the beginning how many iterations are there.

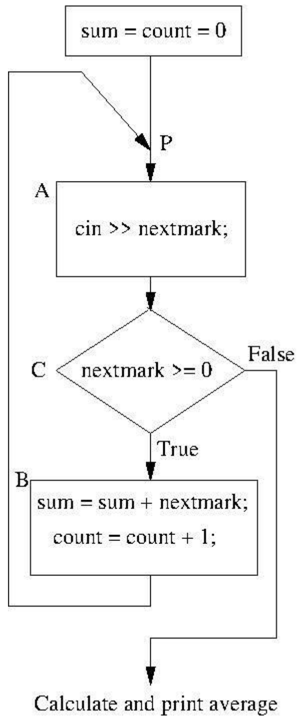
Whatever we can do using ***repeat*** can be done using ***while***

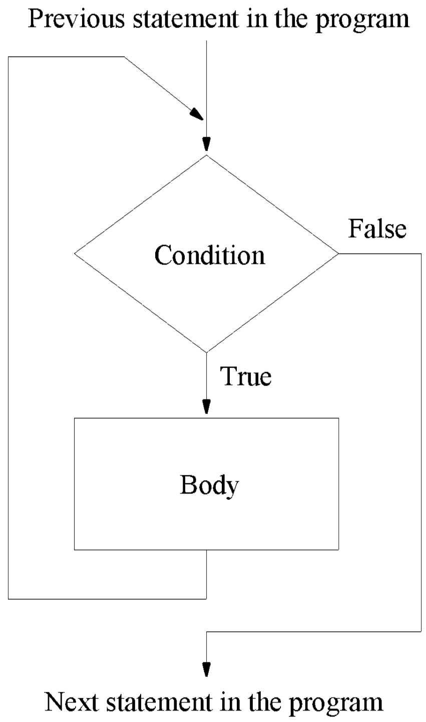
- Converse not true

Mark averaging: manual algorithm

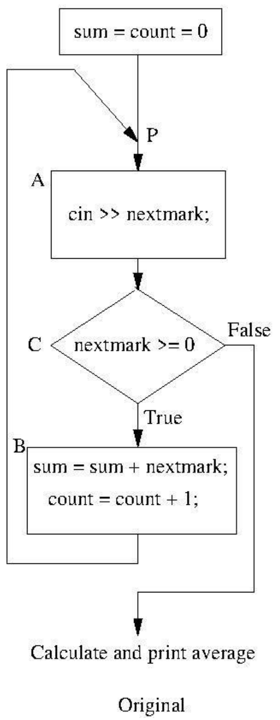
1. Set sum = count = 0.
2. Read next value into nextmark
3. If nextmark < 0, then go to step 7, if it is >= 0, continue to step 4.
4. sum = sum + nextmark
5. count = count + 1
6. Go to step 2.
7. Print sum/count.

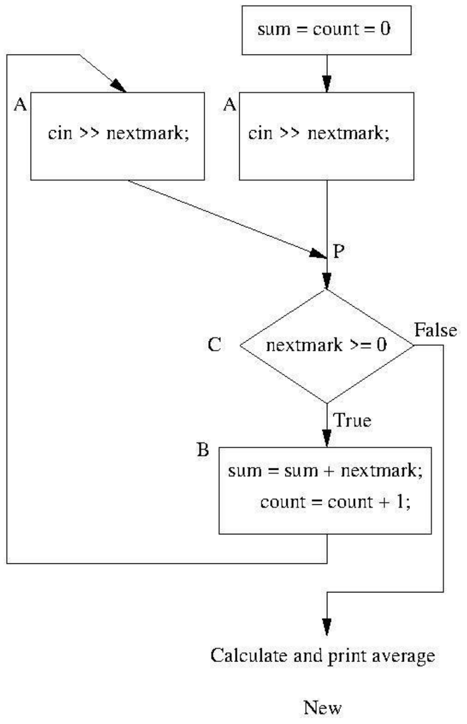
Structures do not match.




Previous statement in the program
Condition
False
True
Body
Next statement in the program

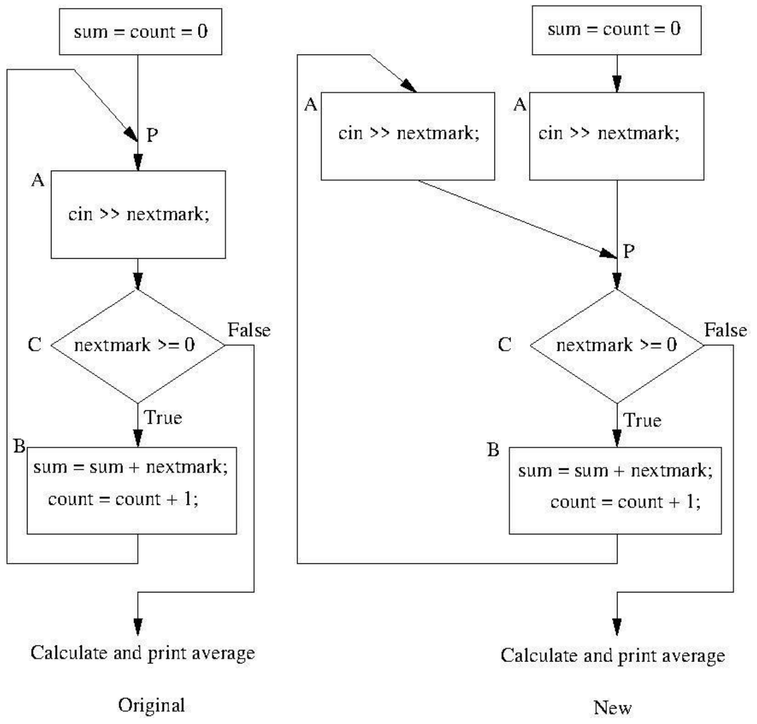
A different flowchart for mark averaging





- **Claim**: If a certain block is executed after two paths merge, then we might as well execute it separately on the two paths before the paths merge, and not after.
- Blocks B,A can be merged in that order to get the body of while.

A different flowchart for mark averaging



```
main_program{
  float nextmark, sum = 0;
  int count = 0;
  cin >> nextmark;       // A
  while(nextmark >= 0){  // C
    sum = sum + nextmark; // B
    count = count + 1;   // B
    cin >> nextmark; // A copy
  }
  cout << sum/count << endl;
}
```

Exercise

Write a turtle controller which receives commands from the keyboard and acts per the following rules:

- If command = ‘f’ : move turtle forward 100 pixels
- If command = ‘r’ : turn turtle right 90 degrees.
- If command = ‘x’ : finish execution.

What we discussed

- In while loops, the first step is to check whether we need to execute the next iteration.
- In some loops that we want to write, the first step is not a condition check. The condition check comes later.
- Such loops can be modified by making a copy of the steps before the condition check.
- NEXT: loops in which condition checks can happen also inside the body

The break statement

Form: The ***break*** keyword is a statement by itself.

What happens when control reaches break statement:

- The execution of the ***while*** statement which contains it is terminated.
- The execution continues from the next statement following that ***while*** statement.

Example of break

```
main_program{
  float nextmark, sum = 0;
  int count = 0;
  while(true){
            cin >> nextmark;
            if(nextmark < 0) break;
            sum += nextmark;
            count++;
  }
  cout << sum/count << endl;
}
```

- The condition of the while statement is given as ***true*** – body will always be entered.
- If nextmark < 0:
  - the while loop execution will terminate
  - Execution continues from the statement after while, i.e. cout ...
- Exactly what we wanted!
  - No need to copy code.
- Some programmers do not like break statements because continuation condition gets hidden inside body, instead of being at the top.
- Condition for breaking = compliment of condition for continuing loop

The continue statement

- The ***continue*** is another single word statement.
- If it is encountered in execution:
  - The control directly goes to the beginning of the loop for the next iteration,
  - Statements from the ***continue*** to the end of the loop body are skipped.

Example

Mark averaging with an additional condition:

- If a number > 100 is read, ignore it.
  - say because marks can only be at most 100
- Continue execution with the next number.
- As before stop and print the average only when a negative number is read.

Code for new mark averaging

```
main_program{
  float nextmark, sum = 0;
  int count = 0;
  while(true){
            cin >> nextmark;
            if(nextmark > 100) continue;
            if(nextmark < 0) break;
            sum += nextmark;
            count++;
  }
  cout << sum/count << endl;
}
```

The do while statement

- Not very common.
- Discussed in the book.

Exercise

- Write the turtle controller program using the break statement.

Requirements:

  - ‘f’ : mover forward 100 pixels.
  - ‘r’ : right turn 90 degrees.
  - ‘x’ : exit program.

What we discussed

- The break statement enables us to exit a loop from inside the body.
- If break appears inside a while statement which is itself nested inside another while, then the inner while statement is terminated.
- The continue statement enables us to skip the rest of the iteration.