CS1100 – Introduction to Programming

Lecture 10

Instructor: Shweta Agrawal (shweta.a@cse.iitm.ac.in)

Mini-calculator using While loop

Input operator and two operands, output result till quit.

Mini-calculator using While loop

Input operator and two operands, output result till quit.

```
#include<stdio.h>
int main() {
    int x; int y; char op;
    printf("Input the operator \t"); scanf ("%c", &op);
    while (op != 'q') {
       printf("Input the first integer \t"); scanf ("%d", &x);
       printf("Input the second integer \t"); scanf ("%d", &y);
       switch (op) {
          case '+': printf("x+y = %d\n", x+y); break;
          case '%': printf("x mod y = %d\n", x%y); break;
          case '/': printf("x/y = %.2f\n", (x*0.1)/y); break;
          case 'q': ; break;
          default : printf("invalid operator\n"); break;
       }
       getchar(); printf("Input the operator \t");
       scanf ("%c", &op);
    }
}
```

Mini-calculator: Case Insensitive Quit

```
#include<stdio.h>
int main() {

    // read op.
    while ((op != 'q') || (op != 'Q')) {
        // all the same stuff.
    }
}
```

Mini-calculator: Case Insensitive Quit

```
#include<stdio.h>
int main() {

    // read op.
    while ((op != 'q') || (op != 'Q')) {
        // all the same stuff.
    }
}
```

- Is the condition correct?

Mini-calculator: Case Insensitive Quit

```
#include<stdio.h>
int main() {

    // read op.
    while ((op != 'q') || (op != 'Q')) {
        // all the same stuff.
    }
}
```

- Is the condition correct?
- We want to stop when op is either 'q' or 'Q'.

Mini-calculator: Case Insensitive Quit

```
#include<stdio.h>
int main() {

    // read op.
    while ((op != 'q') || (op != 'Q')) {
        // all the same stuff.
    }
}
```

- Is the condition correct?
- We want to stop when op is either 'q' or 'Q'.
  (op == 'q' || op == 'Q') then quit the loop.

Mini-calculator: Case Insensitive Quit

```
#include<stdio.h>
int main() {

    // read op.
    while ((op != 'q') || (op != 'Q')) {
        // all the same stuff.
    }
}
```

- Is the condition correct?
- We want to stop when op is either 'q' or 'Q'.
  (op == 'q' || op == 'Q') then quit the loop.
- We want to enter the loop when the above condition is false!

Mini-calculator: Case Insensitive Quit

```
#include<stdio.h>
int main() {

    // read op.
    while ((op != 'q') || (op != 'Q')) {
        // all the same stuff.
    }
}
```

- Is the condition correct?
- We want to stop when op is either 'q' or 'Q'.
  (op == 'q' || op == 'Q') then quit the loop.
- We want to enter the loop when the above condition is false!
  !(op == 'q' || op == 'Q')

Mini-calculator: Case Insensitive Quit

```
#include<stdio.h>
int main() {

    // read op.
    while ((op != 'q') || (op != 'Q')) {
        // all the same stuff.
    }
}
```

- Is the condition correct?
- We want to stop when op is either 'q' or 'Q'.
  (op == 'q' || op == 'Q') then quit the loop.
- We want to enter the loop when the above condition is false!
  !(op == 'q' || op == 'Q') same as (op != 'q' && op != 'Q')

Mini-calculator: using a true expression

Let the while loop run forever!

Mini-calculator: using a true expression

Let the while loop run forever!

```
while (1) {
   printf("Input the operator \t"); scanf ("%c", &op);
   printf("Input the first integer \t"); scanf ("%d", &x);
   printf("Input the second integer \t"); scanf ("%d", &y);
   switch (op) {
      case '+': printf("x+y = %d\n", x+y); break;
      // other cases..
   }
   getchar();
   //printf("Input the operator \t");
   //scanf ("%c", &op);
}
```

Mini-calculator: using a true expression

Let the while loop run forever!

```
while (1) {
   printf("Input the operator \t"); scanf ("%c", &op);
   printf("Input the first integer \t"); scanf ("%d", &x);
   printf("Input the second integer \t"); scanf ("%d", &y);
   switch (op) {
      case '+': printf("x+y = %d\n", x+y); break;
      // other cases..
   }
   getchar();
   //printf("Input the operator \t");
   //scanf ("%c", &op);
}
```

- How does the program terminate?
- We can make a check and break out of the loop!

Mini-calculator: using a true expression

Let the while loop run forever and break it when needed.

Mini-calculator: using a true expression

Let the while loop run forever and break it when needed.

```
while (1) {
   printf("Input the operator \t"); scanf ("%c", &op);
   if (op == 'q' || op == 'Q') {
       break;
   }
   printf("Input the first integer \t"); scanf ("%d", &x);
   printf("Input the second integer \t"); scanf ("%d", &y);
   switch (op) {
     case '+': printf("x+y = %d\n", x+y); break;
      // other cases..
   }
   getchar();
}
```

- How does the program terminate?

Mini-calculator: using a true expression

Let the while loop run forever and break it when needed.

```
while (1) {
   printf("Input the operator \t"); scanf ("%c", &op);
   if (op == 'q' || op == 'Q') {
       break;
   }
   printf("Input the first integer \t"); scanf ("%d", &x);
   printf("Input the second integer \t"); scanf ("%d", &y);
   switch (op) {
     case '+': printf("x+y = %d\n", x+y); break;
      // other cases..
   }
   getchar();
}
```

- How does the program terminate?

break takes you out of the current statement (it could be switch / loop).

Stepping back : Stages of Program Design ...

To solve a problem using a computer program :

Stepping back : Stages of Program Design ...

To solve a problem using a computer program :

- Need to develop an idea of the sequence of statements and the flow of the control through the statements to achieve a given task.

Stepping back : Stages of Program Design ...

To solve a problem using a computer program :

- Need to develop an idea of the sequence of statements and the flow of the control through the statements to achieve a given task.
- **Algorithm** : The description of the idea - *a step-by-step procedure to solve the problem*.

Stepping back : Stages of Program Design ...

To solve a problem using a computer program :

- Need to develop an idea of the sequence of statements and the flow of the control through the statements to achieve a given task.
- **Algorithm** : The description of the idea - *a step-by-step procedure to solve the problem*.
- **Pseudo-code :** A language-independent but program-like description of an algorithm.

Stepping back : Stages of Program Design ...

To solve a problem using a computer program :

- Need to develop an idea of the sequence of statements and the flow of the control through the statements to achieve a given task.
- **Algorithm** : The description of the idea - *a step-by-step procedure to solve the problem*.
- **Pseudo-code :** A language-independent but program-like description of an algorithm.
- **Flowchart :** A diagramatic representation of the algorithm is called a *flowchart*.

More examples

- Enter a number and check whether that number is a Perfect number or not. If the sum of all factors equals that number, it is called a perfect number.

More examples

- Enter a number and check whether that number is a Perfect number or not. If the sum of all factors equals that number, it is called a perfect number.
- Write a program in C to find the prime numbers within a range of numbers.

More examples

- Enter a number and check whether that number is a Perfect number or not. If the sum of all factors equals that number, it is called a perfect number.
- Write a program in C to find the prime numbers within a range of numbers.
- Write a C program to check whether a number is a palindrome or not.

An Illustration : Collatz Sequence

An Illustration : Collatz Sequence

Algorithm : To get a Collatz sequence from a number, if it's even, divide it by two, and if it's odd, multiply it by three and add one. Continue the operation on the result of the previous operation until the number becomes 1.

An Illustration : Collatz Sequence

Algorithm : To get a Collatz sequence from a number, if it's even, divide it by two, and if it's odd, multiply it by three and add one. Continue the operation on the result of the previous operation until the number becomes 1.

Pseudo-code:

```
begin program
 while n is more than 1
   show n
   if n is odd then
     set n = 3n + 1
   else
     set n = n / 2
   endif
 endwhile
 show n
end program
```

An Illustration : Collatz Sequence

Algorithm : To get a Collatz sequence from a number, if it's even, divide it by two, and if it's odd, multiply it by three and add one. Continue the operation on the result of the previous operation until the number becomes 1.

Pseudo-code:

```
begin program
 while n is more than 1
   show n
   if n is odd then
     set n = 3n + 1
   else
     set n = n / 2
   endif
 endwhile
 show n
end program
```

C-program segment:

```
while(n > 1) {
  printf("%d,",n);
  if (n%2 == 1)
    n = 3*n+1;
  else
    n = n/2;
}
printf("%d.",n);
```

Programs : Termination and Correctness

Someone claims that $\text{For every } n,\ \sum_{k=1}^{n} k = \frac{n(n-1)}{2}$.

Programs : Termination and Correctness

Someone claims that $\text{For every } n,\ \sum_{k=1}^{n} k = \frac{n(n-1)}{2}$. You ask for proof !

Programs : Termination and Correctness

Someone writes a program and claims that it solves the problem correctly for all inputs.

Programs : Termination and Correctness

Someone writes a program and claims that it solves the problem correctly for all inputs. You ask for proof !

Programs : Termination and Correctness

Someone writes a program and claims that it solves the problem correctly for all inputs. You ask for proof !

- "Hello World" program is easy. Hand simulation works. Infeasible when program receives inputs (many possibilities).

Programs : Termination and Correctness

Someone writes a program and claims that it solves the problem correctly for all inputs. You ask for proof !

- "Hello World" program is easy. Hand simulation works. Infeasible when program receives inputs (many possibilities).
- An interesting complication : even for a given input, a loop could run for infinite times (hence not hand-simulatable).

Programs : Termination and Correctness

Someone writes a program and claims that it solves the problem correctly for all inputs. You ask for proof !

- "Hello World" program is easy. Hand simulation works. Infeasible when program receives inputs (many possibilities).
- An interesting complication : even for a given input, a loop could run for infinite times (hence not hand-simulatable).
- There has to be mathematical statements which proves:

Programs : Termination and Correctness

Someone writes a program and claims that it solves the problem correctly for all inputs. You ask for proof !

- "Hello World" program is easy. Hand simulation works. Infeasible when program receives inputs (many possibilities).
- An interesting complication : even for a given input, a loop could run for infinite times (hence not hand-simulatable).
- There has to be mathematical statements which proves:
  - **Termination:** The loop (hence, the program) terminates.

Programs : Termination and Correctness

Someone writes a program and claims that it solves the problem correctly for all inputs. You ask for proof !

- "Hello World" program is easy. Hand simulation works. Infeasible when program receives inputs (many possibilities).
- An interesting complication : even for a given input, a loop could run for infinite times (hence not hand-simulatable).
- There has to be mathematical statements which proves:
  - **Termination:** The loop (hence, the program) terminates. (Idea : Some measure decreases as loop executes every time, and hence finally, it should terminate).

Programs : Termination and Correctness

Someone writes a program and claims that it solves the problem correctly for all inputs. You ask for proof !

- "Hello World" program is easy. Hand simulation works. Infeasible when program receives inputs (many possibilities).
- An interesting complication : even for a given input, a loop could run for infinite times (hence not hand-simulatable).
- There has to be mathematical statements which proves:
  - **Termination:** The loop (hence, the program) terminates. (Idea : Some measure decreases as loop executes every time, and hence finally, it should terminate).
  - **Correctness:** The program is correctly producting the expected output.

Programs : Termination and Correctness

Someone writes a program and claims that it solves the problem correctly for all inputs. You ask for proof !

- "Hello World" program is easy. Hand simulation works. Infeasible when program receives inputs (many possibilities).
- An interesting complication : even for a given input, a loop could run for infinite times (hence not hand-simulatable).
- There has to be mathematical statements which proves:
  - **Termination:** The loop (hence, the program) terminates. (Idea : Some measure decreases as loop executes every time, and hence finally, it should terminate).
  - **Correctness:** The program is correctly producting the expected output. (Idea : After every iteration of the loop, the partial result that it computes implies the final result).

Programs : Termination and Correctness

Someone writes a program and claims that it solves the problem correctly for all inputs. You ask for proof !

- "Hello World" program is easy. Hand simulation works. Infeasible when program receives inputs (many possibilities).
- An interesting complication : even for a given input, a loop could run for infinite times (hence not hand-simulatable).
- There has to be mathematical statements which proves:
  - **Termination:** The loop (hence, the program) terminates. (Idea : Some measure decreases as loop executes every time, and hence finally, it should terminate).
  - **Correctness:** The program is correctly producting the expected output. (Idea : After every iteration of the loop, the partial result that it computes implies the final result).
- Algorithm designer's job. Not so much programmer's.

More Practice Problems

- Write a program to check if a given number n is prime or not.

More Practice Problems

- Write a program to check if a given number n is prime or not.

  Algorithm: Check, for every number m in the range 2 to $n-1$, whether m divides n or not. If none divides, then you can declare that it is a prime number. If one of them divides, then you can declare right away that is is a composite number.