LOOP PROGRAMMING TECHNIQUES

#1: Sum of 10 Known Values

- Write a C++ program to compute the sum of the first 10 natural numbers
- $\sum_{i=1}^{10} i$
- Should we use a `while` or `for` loop?

Compute $\sum_{i=1}^{10} i$

```
int sum(0);
for (int i = 1; i <= 10; i++)
{
  sum += i;
}

cout << "sum is " << sum << endl;
```

#2: Sum of 10 Unknown Values

- Write a C++ program to compute the sum of 10 integers entered by the user
- $\sum_{i=1}^{10} k_i$, where k_i is entered by the user
- Should we use a `while` or `for` loop?

Compute $\sum_{i=1}^{10} k_i$

#3: Sum of 10 Unknown Values

- Write a C++ program to compute the sum of all negative integers, and the sum of all positive integers, from 10 integers entered by the user

- Should we use a `while` or `for` loop?

Sum of negative and positive integers

#4: Sum of X Unknown Values

- Write a C++ program to compute the sum of all negative integers, and the sum of all positive integers, entered by the user until the user enters 0

- Should we use a `while` or `for` loop?

Sum of negative and positive integers until user enters 0

#5: Function Evaluation

- Write a C++ program to evaluate (implement) the function $f(x) = 10x^2 + 3x - 2$ for integer values of x in the range $[x_{min}, x_{max}]$, which are entered by the user

- Should we use a `while` or `for` loop?

Function Evaluation: $f(x) = 10x^2 + 3x - 2$

#6: Prompt for Correct Input

- Write a C++ program that asks the user for a person's age. If an incorrect age is given, the program will repeatedly ask for the age again until a correct age is entered.
  - A correct age is between [1, 110]
- Should we use a `while` or `for` loop?

Prompt for Correct Input

#7: Sum of Cubes

- Write a C++ program to compute the sum of the cubes of the first n numbers, where n is entered by the user
- $\sum_{i=1}^{n} i^3$
- Should we use a `while` or `for` loop?

Sum of Cubes

#8: Summation Over Two Variables

- Write a C++ program to compute the following nested summation, where n is entered by the user

- $\sum_{i=0}^{n} \sum_{j=0}^{i} (i - j)$

- Should we use a `while` or `for` loop?

Summation Over Two Variables

- Compute $\sum_{i=0}^{n} \sum_{j=0}^{i} (i - j)$
- Table of (i - j) (where j ≤ i):

(i - j)	j=0	j=1	j=2	j=3	…
i=0	0				…
i=1	1	0			…
i=2	2	1	0		…
i=3	3	2	1	0	…
…	…	…	…	…	…

Algorithm

- $\sum_{i=0}^{n} \sum_{j=0}^{i} (i - j)$
  - How are we going to compute this?

- Determine an ***algorithm***

From "Programming and Problem Solving with C++" By Nell Dale:

- An ***algorithm*** is "a step-by-step procedure for solving a problem"

Algorithm

Compute $\sum_{i=0}^{n} \sum_{j=0}^{i} (i - j)$

1. $sum \leftarrow 0$;
2. for $i \leftarrow 0$ to n do
3. for $j \leftarrow 0$ to i do
4. $sum \leftarrow sum + (i - j)$;

Summation Over Two Variables

- Declare variable to hold running sum
- Input n from the user

```
long sum(0);  // Initialize sum to zero
int n;

cout << "Enter max value of i: ";
cin >> n;
```

Summation Over Two Variables

```
long sum(0);  // Initialize sum to zero
int n(0);

cout << "Enter max value of i: ";
cin >> n;

for (int i = 0; i <= n; i++)
{
   for (int j = 0; j <= i; j++)
   {
      sum += i - j;
   }
}

cout << "sum_{i=0}^n sum_{j=0}^i (i-j) = "
   << sum << endl;
```

#9: Prime Numbers

- Write a C++ program to display the prime numbers between 2 and `n`, where `n` is entered by the user
- What is a ***prime number***?
  - A whole number greater than 1 whose only factors are 1 and itself
  - E.g., 2, 3, 5, 7, 11, 13, etc
- A ***composite number*** has more than two factors
- Should we use a `while` or `for` loop?

Prime Numbers

- How do we know if a given number is a prime number?
  - Why is 6 NOT a prime number?
    - Factors are 1, 2, 3, and 6
  - Why is 7 a prime number?
    - Factors are 1 and 7

Prime Numbers Algorithm

- Given a number k, is it a prime number?
- Algorithm
  - Compute k mod j, where j takes on the values $2 \ldots k - 1$
  - If k mod j is 0 for ANY value of j then k is NOT a prime number

Algorithm

1. for $k \leftarrow 2$ to n do
2. composite ← false;
3. for $j \leftarrow 2$ to k - 1 do
4. if (k mod $j = 0$) then
5. composite ← true;
6. if (composite = false) print k

Variable *composite* is called a *Boolean flag*

prime.cpp

```
...
int main()
{
  int n(0);
  bool flag_composite(false);

  cout << "Enter n: ";
  cin >> n;
...
```

prime.cpp

```
...
  cout << "Prime numbers:" << endl;
  for (int k = 2; k <= n; k++)
    {
      flag_composite = false;
      for (int j = 2; j < k && !flag_composite; j++)
        {
          if (k % j == 0) // if (k mod j == 0)
            {
              flag_composite = true;
            }
        }

      if (!flag_composite)
        {
           cout << k << endl;  // k is prime
        }
    }
...
```

```
for (int k = 2; k <= n; k++) {
      flag_composite = false;
      for (int j = 2; j < k && !flag_composite; j++) {
          if (k%j == 0)           // if (k mod j == 0)
            { flag_composite = true; }
        }

      if (!flag_composite)
        { cout << k << endl; }   // k is prime
    }
```

```
> prime.exe
Enter n: 20
2
3
5
7
11
13
17
19
```

Types of Loops

- **Pretest Loops** check the looping condition first, then begins execution
  - `while`
  - `for`
- **Posttest Loops** begins execution first, then checks looping condition
  - `do-while`

Summary

- **`while`** loops
  - Repeat until some condition is fulfilled;
    - Unknown # of iterations
  - Pretest loop.
    - May iterate 0 times

- **`for`** loops
  - Used for counting;
    - Known # of iterations
  - 3 parts: `for (initialize; condition; alter){...}`
  - Pretest loop.
  - May iterate 0 times

- **`do-while`** loops
  - Example: “Do you wish to continue?”
  - Posttest loop.
  - Will always iterate at least once