CSCI 1370

MARCH 1, 2017

REVISITING FUNCTIONS

FUNCTIONS

- Recall that we can use functions to organize our programs, e.g.:

```
#include<iostream>
using namespace std;

void sayHello()
{
   cout << “Hello world!”;
}

int main()
{
   sayHello();
}
```

FUNCTIONS

- Recall that functions can take *parameters*, e.g.:

```
#include<iostream>;
using namespace std;

void printNumber(int i)
{
   cout << i;
}

int main()
{
   printNumber(1370);
}
```

FUNCTIONS

- Recall that functions can *return* data, e.g.:

```
#include<iostream>;
using namespace std;

int addNumbers(int num1, int num2)
{
   return num1 + num2;
}

int main()
{
   int sum = addNumbers(2, 2);
}
```

EXERCISE: MAX FUNCTION

- With a partner, write a *function* that:
  - takes two parameters of type `int`: `num1` and `num2`
  - and returns the larger of the two numbers

EXERCISE: MAX FUNCTION (CONT.)

- Now, modify your program so that within the main function you:
  - call the max function
  - and print out the result

REVISITING DATA TYPES

DATA TYPE: `bool`

- Has two values: `true` and `false`.
- Used for manipulating logical (Boolean) expressions.
- `bool`, `true`, and `false` are *reserved* words.

RANGE FUNCTION

- With a partner, write a program that:
  - gets three integers from the user (call them `x`, `y`, and `z`),
  - stores in a *Boolean* variable (call it `isInRange`) whether `x`'s value is between that of `y`'s and `z`'s,
  - and prints the value of `isInRange` to the screen.

RANGE PROGRAM

```
1    #include <iostream>
2    using namespace std;
3
4    int main() {
5        int x, y, z;
6        bool isInRange;
7        cout << “Enter 3 integers:”;
8        cin >> x >> y >> z;
9        if ( (y <= x && x <= z) || (z <= x && x <= y) ) {
10           isInRange = true;
11       } else {
12           isInRange = false;
13       }
14       cout << isInRange;
16   }
```

EXERCISE

- With a partner, determine what gets printed to the screen when the user inputs:
  - `3, 2, 1`
  - `10, 5, 20`
  - `-10, -20, -5`
  - `-1, -1, -3`
  - `2, 2, 2`

ITERATIVE EXECUTION

EXERCISE

- With a partner, write a program that:
  - gets five numbers from the user,
  - computes their sum,
  - and prints the result to the screen.

PROGRAM

```
1    #include <iostream>
2    using namespace std;
3
4    int main() {
5        int num1, num2, num3, num4, num5;
6        cout << “Enter 5 numbers:”;
7        cin >> num1 >> num2 >> num3 >> num4 >> num5;
8        cout << num1 + num 2 + num3 + num4 + num5;
10   }
```

ALTERNATIVE: USE *ITERATION*

```
1    #include <iostream>
2    using namespace std;
3
4    int main() {
5        int input, sum = 0, i = 1;
6        cout << “Enter 5 numbers:”;
7        while (i <= 5) {
8            cin >> input;
9            sum = sum + input;
10           i = i + 1;
11       }
12       cout << sum;
13   }
```

SUMS 5 NUMBERS:

```
#include <iostream>

using namespace std;


int main() {

    int input, sum = 0, i = 1;

    cout << “Enter 5 numbers:”;

    while (i <= 5) {

        cin >> input;

        sum = sum + input;

        i = i + 1;

    }

    cout << sum;

}
```

SUMS 50 NUMBERS:

```
#include <iostream>

using namespace std;


int main() {

    int input, sum = 0, i = 1;

    cout << “Enter 50 numbers:”;

    while (i <= 50) {

        cin >> input;

        sum = sum + input;

        i = i + 1;

    }

    cout << sum;

}
```

SUMS 5 NUMBERS:

```
#include <iostream>

using namespace std;


int main() {

    int input, sum = 0, i = 1;

    cout << “Enter 5 numbers:”;

    while (i <= 5) {

        cin >> input;

        sum = sum + input;

        i = i + 1;

    }

    cout << sum;

}
```

SUMS 50 NUMBERS:

```
#include <iostream>

using namespace std;


int main() {

    int input, sum = 0, i = 1;

    cout << “Enter 50 numbers:”;

    while (i <= 50) {

        cin >> input;

        sum = sum + input;

        i = i + 1;

    }

    cout << sum;

}
```

PROGRAM EXECUTION

- Sequential: program executes line by line.
- Conditional: program executes **conditionally** (if a logical expression evaluates to **true**).

PROGRAM EXECUTION

- **Sequential**: program executes line by line.
- **Conditional**: program executes **conditionally** (if a logical expression evaluates to **true**).
- **Iterative**: program repeats execution until a condition is met.

ITERATIVE EXECUTION

- while is used to control iterative execution

```
while ( condition )
{
    // do some stuff repeatedly, as long as condition is met
}
```

- Note: iterative execution happens 0 (condition not met) or more times.

ELEMENTS OF AN ITERATIVE STATEMENT

- **Initialization**: what are the variables' values before the loop starts?
- **Exit condition**: when does the loop quit?
- **Update**: how are the variables changed within the loop?

EXAMPLE

```
1    #include <iostream>
2    using namespace std;
3
4    int main() {
5        int input, sum = 0, i = 1;
6        cout << “Enter 3 numbers:”;
7        while (i <= 3) {
8            cin >> input;
9            sum = sum + input;
10           i = i + 1;
11       }
12       cout << sum;
13   }
```

INITIALIZATION

```
1    #include <iostream>
2    using namespace std;
3
4    int main() {
5        int input, sum = 0, i = 1;
6        cout << “Enter 3 numbers:”;
7        while (i <= 3) {
8            cin >> input;
9            sum = sum + input;
10           i = i + 1;
11       }
12       cout << sum;
13   }
```

EXIT CONDITION

```
1    #include <iostream>
2    using namespace std;
3
4    int main() {
5        int input, sum = 0, i = 1;
6        cout << “Enter 3 numbers:”;
7        while (i <= 3) {
8            cin >> input;
9            sum = sum + input;
10           i = i + 1;
11       }
12       cout << sum;
13   }
```

UPDATE

```
#include <iostream>
using namespace std;

int main() {
    int input, sum = 0, i = 1;
    cout << “Enter 3 numbers:”;
    while (i <= 3) {
        cin >> input;
        sum = sum + input;
        i = i + 1;
    }
    cout << sum;
}
```

EXERCISE

- Write down the values of `i`, `j` at the beginning of each iteration of the following loop:

```
int i = 0, j = 0;

while ( i < 5 ) {
    j = j + 10;
    ++i;
}
```

EXERCISE

- With a partner, write a program that:
  - gets **five** integers from the user,
  - computes their **average**,
  - and prints the result to the screen.

EXERCISE

- Individually, write what gets printed to the screen when the user inputs:
  - `1, 2, 3, 4, 5`
  - `5, 5, 10, 5, 5`
  - `5, 4, 3, 2, 1`