CSCI 1370

FEBRUARY 1, 2016

REVIEW

Data types tell the computer how the data should be processed.

The **type** assigned to a variable also tells the computer how much space in memory to reserve (in response to a variable declaration).

COMPUTER MEMORY

A **bit** is the smallest piece of memory. It is short for **b**inary dig**it** (as it can have values of 0 or 1).

A **byte** is eight consecutive bits (e.g., 0000 0001).

DATA STORAGE

All data is stored as bits in main memory. The meaning of a set of bits depends on the *encoding* (type).

BINARY REPRESENTATION	INTEGER	CHARACTER
1000 0001	65	'A'
1000 0010	66	'B'
1000 0011	67	'C'

DATA TYPE: `int`

Declaration of a variable of type `int` tells the computer to reserve ~**4 bytes** of space.

Example:

```
int x = 1;
```

DATA TYPE: `float`

Declaration of a `float` reserves ~**4 bytes** of space.

Example:

```
float x = 1;
```

DATA TYPE: `double`

Declaration of a `double` reserves ~**8 bytes** of space.

Example:

```
double x = 1;
```

DATA TYPE: `char`

Declaration of a variable of type `char` tells the computer to reserve **1 byte** of space. This type is used for *characters*: letters, digits, and special symbols.

Example:

```
char x = '1';
```

DATA TYPE: `string`

To store a *set of ordered characters*, we need to declare a variable of type string.
String *literals* use as many bytes as there are characters, *plus one*...
String *variables* reserve ~**32 bytes** of space.

Example:

```
string x = “1”;
```

OVERFLOW

Overflow occurs when we assign a value that is too large to be stored in the respective variable.

As a result, the variable will be assigned a value that is “wrapped around” the set of possible values.

EXAMPLE: OVERFLOW

In the code below, we are declaring an integer of type ***short int*** and ***initializing it to the largest value it can hold***. When we subsequently increment its value, we run into **overflow**.

```
short num = 32767;
cout << num;
num = num + 1;
cout << num;      //prints -32768
```

FUNCTIONS

Programs are divided up into **functions**.

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

```
int main()
{
   cout << “Hello world!”;
}
```

THE `main` FUNCTION

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

int main()
{
   cout << “Hello world!”;
}
```

THE `main` FUNCTION

Every C++ must have a `main` function.
When a C++ program executes, it starts by ***calling*** `main`.

PROGRAM EXECUTION

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

int main()
{
    cout << “Hello world!”;
}
```

EXAMPLE: A PROGRAM WITH TWO FUNCTIONS

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

void anotherFunction()
{
   cout << “Goodbye!”;
}

int main()
{
   cout << “Hello world!“;
}
```

PROGRAM EXECUTION

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

void anotherFunction()
{
   cout << “Goodbye!”;
}

int main()
{
   cout << “Hello world!“;
}
```

USING FUNCTIONS

To use a function, we must make a **function *call***.

To *call* our function (named “`anotherFunction`”) we need to add the following statement to our code:

```
anotherFunction();
```

EXAMPLE: A PROGRAM WITH TWO FUNCTIONS

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

void anotherFunction()
{
   cout << “Goodbye!”;
}

int main()
{
   cout << “Hello world!“;
}
```

EXAMPLE: A PROGRAM WITH TWO FUNCTIONS

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

void anotherFunction()
{
   cout << “Goodbye!”;
}

int main()
{
   cout << “Hello world!“;
   anotherFunction();
}
```

PROGRAM EXECUTION

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

void anotherFunction()
{
    cout << “Goodbye!”;
}

int main()
{
    cout << “Hello world!“;
    anotherFunction();
}
```

USING FUNCTIONS (CONT.)

To use a function, we must make a **function *call***. Note: any function can call another function at any time.

When a function (let's refer to the caller as A) *calls* another function (B), the computer jumps to executing function B. When the computer finishes executing B, it returns to where it left off with A.

PROGRAM EXECUTION

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

void anotherFunction()
{
    cout << “Goodbye!”;
}

int main()
{
    cout << “Hello world!“;
    anotherFunction();
}
```

ANOTHER EXAMPLE

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

void anotherFunction()
{
    cout << “world”;
}

int main()
{
    cout << “Hello “;
    anotherFunction();
    cout << “!”;
}
```

WRITING FUNCTIONS

To write a function, you must specify a **header** and a **body**.

The **header** has three parts:
a *return type* (e.g., `int`),
a name (e.g., `main`), and
a *list of parameters* between parentheses.

The **body** is a sequence of statements placed between two curly brackets (`{}`).

THE `main` FUNCTION

```
int main()
{
   //statements to execute
}
```

EXERCISE

With a partner, write statements in the `main` function that call `anotherFunction` and complete the output to get “Hello world!” printed to the console.

```
void anotherFunction()
{
   cout << “world”;
}

int main()
{
   // Your statements here.
}
```

EXERCISE

With a partner, now add a *third* function (call it `functionC`), which prints out “glorious”. Modify your code in `main` such that the phrase, “Hello glorious world!” is printed in the console.

```
void anotherFunction()
{
   cout << “world”;
}

int main()
{
   cout << “Hello “;
   anotherFunction();
   cout << “!”;
}
```

ADMINISTRATIVIA

WEEK 3 PARTICIPATION ACTIVITIES

Wednesday:
labs 2a–e (by 9:25am)
exercises 3a-e (by 10pm)

Thursday:
second group meeting (by 8pm)

Sunday:
exercises 3f-i (by 10pm)

QUIZ #4