The Elements of C++ Programs

- In C ++, all subprograms are called functions.
- Every C program must have a function named `main`.
- Other functions are called by `main`.
- Program execution returns to `main` when the callee function completes execution.

Example C++ Program

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

int Square(int);
int Cube(int);

int main()
{
   cout << "The square of 27 is" << Square(27) << endl;
   cout << "and the cube of 27 is" << Cube(27) << endl;
   return 0;
}
```


Example C++ Program (continued)

```
int Square(int n)
{
   return n*n;
}

int Cube(int n)
{
   return n*n*n;
}
```

The left brace({) and right brace (}) mark the beginning and end of the statements to be executed.

Notes on Program Execution

- The first statement in main is `cout << The square of 27 is" << Square(27) << endl;`
- During the execution of this statement, `Square` is called, returning a value of 729, which is printed out, followed by a new line character.
- Then, the statement `cout << "and the cube of 27 is" << Cube(27) << endl;` is executed which includes a call of `Cube.` The value 19683 (the cube of 27) is displayed.
- Both `Square` and `Cube` are examples of *value-returning* functions, i.e. `int Square(int n).`
- `main` is also a value-returning function, 0 usually indicates normal (error-free) execution has occurred.
- `main` is called by the operating system.

Example C++ Program

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

int Square(int);
int Cube(int);

int main()
{
   cout << "The square of 27 is" << Square(27) << endl;
   cout << "and the cube of 27 is" << Cube(27) << endl;
   return 0;
}
```


Example C++ Program (continued)

```
int Square(int n)
{
   return n*n;
}

int Cube(int n)
{
   return n*n*n;
}
```

Example C++ Program

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

int Square(int);
int Cube(int);

int main()
{
   cout << "The square of 27 is" << Square(27) << endl;
   cout << "and the cube of 27 is" << Cube(27) << endl;
   return 0;
}
```


Example C++ Program (continued)

```
int Square(int n)
{
   return n*n;
}

int Cube(int n)
{
   return n*n*n;
}
```


Example C++ Program

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

int Square(int);
int Cube(int);

int main()
{
   cout << "The square of 27 is" << Square(27) << endl;
   cout << "and the cube of 27 is" << Cube(27) << endl;
   return 0;
}
```


Example C++ Program (continued)

```
int Square(int n)
{
   return n*n;
}

int Cube(int n)
{
   return n*n*n;
}
```


Example C++ Program

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

int Square(int);
int Cube(int);

int main()
{
   cout << "The square of 27 is" << Square(27) << endl;
   cout << "and the cube of 27 is" << Cube(27) << endl;
   return 0;
}
```


Example C++ Program (continued)

```
int Square(int n)
{
   return n*n;
}

int Cube(int n)
{
   return n*n*n;
}
```

Example C++ Program

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

int Square(int);
int Cube(int);

int main()
{
   cout << "The square of 27 is" << Square(27) << endl;
   cout << "and the cube of 27 is" << Cube(27) << endl;
   return 0;
}
```


Example C++ Program (continued)

```
int Square(int n)
{
   return n*n;
}

int Cube(int n)
{
   return n*n*n;
}
```


Example C++ Program

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

int Square(int);
int Cube(int);

int main()
{
   cout << "The square of 27 is" << Square(27) << endl;
   cout << "and the cube of 27 is" << Cube(27) << endl;
   return 0;
}
```


Example C++ Program (continued)

```
int Square(int n)
{
   return n*n;
}

int Cube(int n)
{
   return n*n*n;
}
```


Example C++ Program

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

int Square(int);
int Cube(int);

int main()
{
   cout << "The square of 27 is" << Square(27) << endl;
   cout << "and the cube of 27 is" << Cube(27) << endl;
   return 0;
}
```


Example C++ Program (continued)

```
int Square(int n)
{
   return n*n;
}

int Cube(int n)
{
   return n*n*n;
}
```

Example C++ Program

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

int Square(int);
int Cube(int);

int main()
{
   cout << "The square of 27 is" << Square(27) << endl;
   cout << "and the cube of 27 is" << Cube(27) << endl;
   return 0;
}
```


Example C++ Program (continued)

```
int Square(int n)
{
   return n*n;
}

int Cube(int n)
{
   return n*n*n;
}
```


Syntax and Semantics

- Syntax consists of the formal rules governing how valid instructions are written in a programming language. (grammar)
- Semantics consists of the set of rules that determines the meaning of instructions written in a programming language. (meaning)
- A metalanguage is a language that is used to write the syntax rules for another language.
- In this book, we write the syntax rules for C++ using a metalanguage called a syntax template.

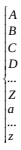
Syntax Templates

- A syntax template is a generic example of the C++ construct being defined.
- Graphic conventions show which portions are optional and which can be repeated.
  - A boldface word or symbol is a literal word or symbol in the C++ language.
  - A nonboldface word can be replaced by another template.
  - A curly brace is used to indicate a list of items, from which one item can be chosen.
  - Shading indicates an optional part of the definition.

Syntax Template Example

- Identifier

- Letter



Digit

Naming Program Elements

- Identifiers are used in C++ to name things – identifiers are made up of letters, digits, and the underscore character (_), and must begin with a letter or an underscore.
- Valid identifiers

```
sum_of_squares
J9
box_22A
GetData
Bin3D4
count
```

Invalid identifiers

```
40hours
Get Data
box-22
cost_in_$
int
```

Data and Data Types

- A data type is a specific set of data values, along with a set of operations on those values.
- Data is stored in the computer's memory.
- Data is like post office boxes, some are larger than others.
- Two data types for now:
  - The `char` Data Type – describes data consisting of one alphanumeric character. We enclose them in single quotation marks, e.g., 'A' 'a' '_' '8'
  - The `string` Data Type – describes data consisting of a sequence of characters. We enclose them in double quotation marks, e.g., "Hello" "My name is Rhonda"

Naming Elements: Declaration

- A declaration tells the computer what an identifier represents. (object, function, or type)
- In C++, you must declare every identifier before it is used.
- A memory location which can change is known by a variable name.
- VariableDeclaration

  DataType Identifer, Identifier...;
- A memory location(s) whose value never changes is known by a constant name
- ConstantDeclaration

  `const` DataType Identifier = LiteralValue;

Declaration Examples

- Variables

```
char letter, middleInitial, ch;
float payRate;                  // Employee's pay rate
float hours;                    // Hours worked
```

- Constants

```
const string STARS = "*********";
const char BLANK = ' ';
const string BOOK_TITLE = "Programming and Problem
  Solving with C++";
const string MESSAGE = "Error Condition";
```


Matters of Style

- Warning: C++ is case-sensitive – `Style` and `style` are not the same
- Capitalization Conventions
  - Identifiers representing variables begin with a lower-case letter and have each successive English word capitalized, e.g., `lengthInYards`
  - Names of programmer-written functions and programmer-defined data types begin with capital letters and have each successive English word capitalized, e.g., `CalcPay(payRate, hours, wages)`
  - Identifiers representing constants are all capitals, with words separated by underscores, e.g., `MAX_LENGTH`

Executable Statements

- The value of a variable can be set or changed through an assignment statement,

  e.g., `lastName = "Lincoln";`
- AssignmentStatement

  Variable = Expression;
- An expression is an arrangement of identifiers, literals, and operators that can be evaluated to compute a value of a given type.

Valid and Invalid Assignments

- Given the declarations

```
string firstName;
string middleName;
string lastName;
char middleInitial;
char letter;
string title;
```

- Valid Assignments

```
firstName = "Abraham";
middleName = firstName;
middleName = " ";
lastName = "Lincoln";
title = "President";
middleInitial = ' ';
letter = middleInitial;
```

Invalid Assignments

```
middleInitial = "A.";
letter = firstName;
firstName = Thomas;
"Edison" = lastName;
lastName = ;
```