Syntax and Semantics

Outline

- C++ Structure
- Data types

Terminology

- A programming language is a set of rules, symbols, and special words used to construct a program.
- Syntax (grammar) is a set of rules that precisely states how valid instructions to be constructed in C++.
- Semantics (meaning) is the correctness of instructions written in C++.

C++ Structure

- All C++ programs have
  - header file(s)
  - a function called main.
- Function is a subprogram in C++.
- Begin { and end } markers: to indicate the beginning and ending of a block of statements to be executed.
Identifiers

An **identifier** is a name associated with a function or data object (variable, data type).

- Combinations of letters (A…Z, a…z), digits (0…9), and underscore ( _ )
- Must begin with a letter or underscore
- No special characters such as +, $, ., *, ', etc.
- Case sensitive

Examples

- **Valid identifier:**
  - Value2, Sum, Integer1, Product, Total_Income
- **Invalid Identifier:**
  - Number 1, 2Data, First-Initial, Cost_in_$, float

Reserved Words

A **reserved word** is a predefined word with a special meaning in C++ (appendix A-1), e.g., int, if, else, for, switch...

Data and Data Types

- Each data in C++ has a type associated with it.
- A **data type** is a set of data together with a set of operations on the data values, e.g., char, short, int, float, long...

Classifications of Data types

- **Integral type:**
  - An integral type is a data type possesses integer values, e.g., char, short, int, and long.
- **floating type:**
  - A floating type is a data type with a set of operations on real numbers, e.g., float, double (double precision), and long double.

C++ Simple Data Types

```
Simple types
  Integral
    char  short  int  long  enum
  Floating
    float  double  long double
  unsigned
```
Integral Types

- **int**
  - 123, +123, -123, 22333
- **char**
  - ‘S’, ‘c’, ‘4’, ‘@’

<table>
<thead>
<tr>
<th>Type</th>
<th>Size in Bytes</th>
<th>Minimum Value</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>1</td>
<td>-128</td>
<td>127</td>
</tr>
<tr>
<td>unsigned char</td>
<td>1</td>
<td>0U</td>
<td>255U</td>
</tr>
<tr>
<td>short</td>
<td>2</td>
<td>-32,768</td>
<td>32,767</td>
</tr>
<tr>
<td>unsigned short</td>
<td>2</td>
<td>0U</td>
<td>65,535U</td>
</tr>
<tr>
<td>int</td>
<td>4</td>
<td>-2,147,483,648</td>
<td>2,147,483,647</td>
</tr>
<tr>
<td>unsigned int</td>
<td>4</td>
<td>0U</td>
<td>4,294,967,295U</td>
</tr>
<tr>
<td>long</td>
<td>8</td>
<td>-9,223,372,036</td>
<td>9,223,372,036U</td>
</tr>
<tr>
<td>unsigned long</td>
<td>8</td>
<td>0UL</td>
<td>18,446,744,073</td>
</tr>
</tbody>
</table>

Floating-Point Types

A real number or a floating number has an integer part and a fractional part, with a decimal point in between e.g., 18.0, 127.54, 0.57, 8.17456E-12, 3.652442E4, 7E20.

<table>
<thead>
<tr>
<th>Type</th>
<th>Size in Bytes</th>
<th>Minimum Value</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td>4</td>
<td>3.4E-38</td>
<td>3.4E+38</td>
</tr>
<tr>
<td>double</td>
<td>8</td>
<td>1.7E-308</td>
<td>1.7E+308</td>
</tr>
<tr>
<td>long double</td>
<td>16</td>
<td>3.4E-4932</td>
<td>1.1E+4932</td>
</tr>
</tbody>
</table>

Variables

- A **variable** is an identifier associated with a memory location that used to store data, e.g.,
  ```
  Data1 (memory location: 1110101101)
  ```

Declaration

- A statement that associates an identifier with a data object, a function, or a data type is called declaration statement.

```
- data type Identifier, Identifier, ...;
- int Number1;
- int main();
- char FirstInitial;
- float Wages;
```

Constants

A **constant variable** is an identifier with a fixed value.

```
- const data type Identifier = literal value
- const float PI = 3.14159;
- const int MaxHours = 40;
- const char Blank = ' ';```
**Assignment Statement**

• An assignment statement is a statement that assigns the value of an expression into a variable.

```
Variable = Expression;
```

*Expression*: an arithmetic expression, another identifier or a literal value

---

**Examples**

- `Count = 153;`
- `Count = Count1;`
- `Num1 = Num2*Num3/Num4;`
- `MidInit = 'A';`

---

**More Examples**

```cpp
...
int Number1;
char FirstInitial;
float HourRate = 5.65;
float Wages;
int Hours;
Number1 = 65;
FirstInitial = 'a';
Wages = HourRate*Hours;
...
```

---

**C++ Preprocessor**

Numerous libraries of functions are included as part of any C++ programming package. The actual code for these libraries has been already compiled and to be added to the program during the compiling phase of program construction. We use compiler directive `#include` to inform the compiler which libraries are needed or to be included in the source program.

```cpp
#include <iostream>
#include <iomanip>
#include <cmath>
#include <string>
```

*`iostream` is called a header file whose contents are inserted in place of the `#include` line during compilation. These header files can be found in `c:\Program Files\Microsoft Visual Studio\Vc98\Include`*

---

**Namespace**

• A namespace contains identifiers that used or declared in header files.
• The namespace for identifiers in these header files is called `std`
• The purpose of a namespace is to provide a mechanism that minimizes the possibility of accidentally duplicating names in various parts of a program.


**Namespace directive**

```cpp
using namespace std;
```

using directive in the line indicates that there are identifiers in the namespace called std will be used in the program. Another way to reference identifiers in the namespace std is by using a qualified name

```cpp
std :: cout
```

---

**Mathematical Operations**

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Unary plus</td>
</tr>
<tr>
<td>-</td>
<td>Unary minus</td>
</tr>
<tr>
<td>+</td>
<td>Addition</td>
</tr>
<tr>
<td>–</td>
<td>Subtraction</td>
</tr>
<tr>
<td>*</td>
<td>Multiplication</td>
</tr>
<tr>
<td>/</td>
<td>Division</td>
</tr>
<tr>
<td>%</td>
<td>Modulus</td>
</tr>
</tbody>
</table>

A *unary operator* is an operator with one operand. A *binary operator* is an operator with two operands.

---

**Precedence of C++ Operators**

- `* / %` *first*  
- `+ –` *last*

---

**Examples**

```cpp
Sum = Num1 + Num2;
Average = (Num1 + Num2)/2;
W = X * Y / X + Y;
Delta = B * B – 4 * A * C;
Quotient = X / Y;
Remainder = X % Y;
```

---

**Type of Arithmetic Expressions**

- combining float with either integer or float
  ⇒ float
- combining integers or floats with `/`
  ⇒ integer or float
- combining integers with either `+`, `–`, `*`
  ⇒ integer
- `%` can be used with only integer

---

**Modulus %**

The modulus operator `%` yields the *integer remainder* of the result of dividing its first operand by its second. If either one of the operands is negative, the sign of the result is machine-dependent.
Examples

14 % 3 = 2
-14 % 3 = 2 or -2
14 % -3 = 2 or -2
-14 % -3 = 2 or -2

Increment and Decrement Operators

++ Increment
-- decrement
Examples:
Num ++ or ++ Num ⇔ Num = Num + 1