NIU CS240
Computer Programming In CPP

Nathan Warner

Northern Illinois
University

Computer Science
Northern Illinois University
August 28, 2023 United States

Computer Programming In CPP

1 Basic Language Features

1.1 Data

- There are several data types (numbers, characters, etc)
- individual data items must be declared and named - this is known as creating a variable
- values that are put into variables can come from
  - program instructions
  - user input
  - files
- program instructions can alter these values
- original or newly computer values can go to
  - screen
  - printer
  - disk

Instructions:

- for data input (from keyboard, disk)
- for data output (to screen, printer, disk)
- computation of new values
- program control (decisions, repetition)
- modularization (putting a sequence of instructions into a package called a function)

1.2 The Language

The C++ language is made up of

- keywords/reserved words (if, while, int, etc.)
- symbols: { } = | <= ! [ ] * & (and more)
- programmer-defined names for variables and functions

These programmer-defined names:

- 1 - 31 chars long; use letters, digits, _ (underscore)
- start with a letter or _
- are case-sensitive: *Num* is **different** than *num*
- should be meaningful: *studentCount* is better than s or sc

2 Primitive Data Types

2.1 Data Types

Each data item has a type and a name chosen by the programmer. The type determines the range of possible values it can hold as well as the operations that can be used on it. For example, you can add a number to a numeric data type, but you cannot add a number to someone's name. (What would "Joe" + 1 mean?)

Figure:

Type	Keyword
Boolean	bool
Character	char
String	string
Integer	int
Floating point	float
Double floating point	double
Valueless	void

Note:-

Floating point numbers have a limit of 6 significant figures and doubles have a limit of 12 characters.

2.2 Integers

- no decimal point, comma, or $
- leading + - allowed
- range (on our system): ± 2 billion
- int constants are written as: 1234 −3 43

2.3 Integer declaration

```
int x; // gives type and name; no value yet
int x,y; //declares 2 ints; no values yet
int population = 160000; //declares & sets the initial value
```

Note:-

It is critically importannt that variables have values before they are used in a program.

2.4 Float and Double

These data types are also commonly referred to as **real** variables (following mathematics usage) when it's not important which one we mean and we don't want to always have to say "float or double"

- has decimal point
- leading ± allowed
- no comma, $
- range (float) ± 10 to 38^{th} (limited to 66 sig figs)
- range (double) ± 10 to 308^{th} (limited to 12 sig figs)

2.5 float and double declaration

```
float  pi = 3.1416;   //declares, names, sets init value

double x = 3.5,              //note comma here
       y = 1245.6543; //can use > 6 digits

OR

double x   = 3.5;
double y   = 1245.6543;

float  big = 5.28e3;  // exponential notation: 5280
```

2.6 Char

- can hold just one character
- char constants are written with single quotes 'a'

2.7 Char declarations

```
char ch;
char choice = 'q';
```

2.8 String

- can hold 0 to many characters
- string constants are written with double quotes: "Hello, world"

2.9 String declarations

```
string s;
string MyName = "Amy';
```

3 Arithemetic Operators

The arithmetic operators are:

- + addition
- - subtraction or unary negation (-5)
- * multiplication
- / division (see special notes on division below)
- % modulus division -- integer remainder of integer division

Note:-

There is no exponential operator

In C++, a division with 2 int operands has an int resulting value, But with 1 or 2 float/double operands, the resulting value is a float or double. Be aware. Forgetting this can easily cause an error.

3.1 Arithmetic Expressions

Arithmetic Expressions are formed by operands and operators. Operands are the values used, operators are the things done to the numbers. Parts of arithmetic expressions are often grouped by () for clarity or to affect the meaning of the expression:

Expression	Value	Notes
$x + y$	13	
$x \times y$	22	
$x \times y + x$	33	
$x - y$	9	
$-x + y$	-9	Unary negation: "minus x"
$\frac{x}{y}$	5	int since both ops are int
$x\%y$	1	rem when x divided by y
$\frac{x}{\text{realnum}}$	5.5	one op is real so result is real

Note:-

Note: The expressions above by themselves are not valid C++ statements - they would be part of a statement. But each expression (if it is in a valid C++ statement) has a value. Also - the spaces written between operands and operators are not required by C++, but do improve legibility.

More complex expressions are evaluated by C++ using Rules of Precedence (like algebra)

1. sub-expressions in ()
2. unary -
3. * and / - left to right
4. + and - - left to right

4 Assignment Statement/Operation

The symbol "=" is a **command**. It means: "evaluate the **expression** on the right and store this **value** in the **variable** on the left"

General Syntax:

```
Identifier = Expression;
```

Examples:

```
x = y * 3;
x = x + y;
```

Note:-

Notice that there is always exactly one variable on the left to receive the value of the expression on the right. This is **NOT** like algebra; it is **NOT** an equation. All variables in expressions on the right must have defined values or you get random results. This is an example of what I meant before when I said variables must have values before they are used.

4.1 Multiple assignments

You can do multiple assignments on one line:

```
x = y = z = 3;  //all get value 3
```

4.2 Naming Variables

Use meaningful names that explain themselves to reader of the program. (You, me, the maintainer of the program after you go to your next job...)