C++

Canonical name	C
Date of creation	2013-03-22 16:51:37
Last modified on	2013-03-22 16:51:37
Owner	PrimeFan (13766)
Last modified by	PrimeFan (13766)
Numerical id	12
Author	PrimeFan (13766)
Entry type	Definition
Classification	msc 68N15
Synonym	C with Classes
Related topic	SimpleAnalyticDiscussionOfTheCubicEquation

$++$ is a compiled multi-paradigm programming language based on C that supports procedural, object-oriented and generic programming paradigms. It was developed by Bjarne Stroustrup at Bell Labs in 1983, the same year that the ANSI C standard was ratified. Working in tandem, the ANSI and ISO ratified standard C++ in 1998.

The following C++ program takes two integers as inputs and outputs their greatest common divisor using Euclid's algorithm.

```cpp
#include <iostream>

template<typename T>
T gcd(T x, T y) {
  while ( y != 0 ) {
    T z = x % y;
    x = y;
    y = z;
  }
  return x;
}

int main() {
  int inputX, inputY;

  std::cout << "Please enter x: ";
  std::cin >> inputX;
  std::cout << "Please enter y: ";
  std::cin >> inputY;

  std::cout << "The GCD of " << inputX
            << " and " << inputY
            << " is " << gcd(inputX,inputY) << std::endl;

  return 0;
}
```

This program works pretty much the same as `http://planetmath.org/CProgrammingLanguage` corresponding C example. Instead of C's standard I/O library, the program uses C++'s `iostream` library. A substantial difference between the two programs is that the C++ version of `gcd` function is more general: not only

1

can it work with integers of type `int`, the same code can work[1] with *any* type that has division with remainder defined. The division with remainder corresponds to the operator `%` in C++.

Note that the algorithm, for integers, may return negative numbers for the greatest common divisor, if an input contains a negative. The algorithm is still correct, however, if the "greatest common divisor" is suitably defined, as in modern abstract algebra, to be unique only up to units in a ring. See [?] — which is where the present example comes from — for an interesting discussion around this point.

We could define another GCD function with the same name but that took three integers as input and returned the GCD of the trio. Or we could create our own complex integer object and then write a GCD function with the same name that took complex integers as input. Furthermore, we could define the operators so as to enable a programmer to write things like `w = u + v;` with complex integers rather something like `w.setVal(complexAddition(u, v));`. The programmer's imagination is the limit: the addition operator for a set object could be defined to either add up each element in one set to an element in a second set, or to perform a union of the two sets. Even in our simple GCD example, C++'s `iostream` library has overloaded the bitwise operators to perform concatenation.

References

[Stepanov] Alexander Stepanov. `http://www.stepanovpapers.com/MusserForeward.pdf`Forewor
to *STL Tutorial and Reference Guide*: C++ Programming with the Standard Template Library, second ed. Addison-Wesley, 2001.

---

[1]Moreover, unlike most other object-oriented languages, this abtraction imposes absolutely no performance penalty at run-time.