0.0.1 GiNaC

GiNaC is a relatively new library that was designed to overcome certain design problems encountered and accumulated during the development of `xloops` [1], a package for automatic calculation of Feynman diagrams in quantum field theories. GiNaC (which stands for *GiNaC is Not a CAS*) is entirely written in C++ and provides classes for efficient handling of symbolic expressions. The user interacts with it in compiled C++-programs but other interfaces are easily conceivable, like an interactive shell (a sample implementation is actually included with the sources) or even an interface to the C++ interpreter `Cint` by Masaharu Goto [2].

GiNaC provides methods for the efficient handling of multivariate polynomials, quotients thereof (heuristic and subresultant PRS GCD algorithms), truncated power series, some symbolic linear algebra and special functions as well as algebras needed in high energy physics.

The philosophy behind GiNaC is driven by the observation that sometimes large-scale scientific applications call for a combined approach: computer algebraic methods drive some numerical simulation the results of which get displayed graphically. It is not uncommon to see the whole process itself being controlled via a graphical user interface. The design of such combined symbolical/numerical/graphical systems is largely complicated by technical odds and ends. While many of the more advanced computer algebra systems have powerful built-in graphical capabilities, numerically they cannot compete with compiled code thus establishing a need to use a second system. The interfaces are frequently constructed using fragile pipes, shell scripts, etc. The situation doesn't get any better when one considers the linguistic capabilities of computer algebra systems. To give an idea of what we mean we refer to the section called "Mathematics versus Computer Science" in Wester's chapter "A Critique of the Mathematical Abilities of CA Systems" in [3]: it has an interesting section where some violations of scope are illustrated.

Here is a complete example how symbolic computation works in C++ using the GiNaC framework:

```
#include <ginac/ginac.h>
using namespace GiNaC;

ex HermitePoly(const symbol & x, int n)
{
    const ex HKer = exp(-pow(x,2));
    // uses the identity H_n(x) == (-1)^n exp(x^2) (d/dx)^n exp(-x^2)
    return normal(pow(-1,n) * diff(HKer, x, n) / HKer);
}

int main(int argc, char **argv)
{
    if (argc != 2) {
        cout << "usage:\n" << argv[0] << " <degree>" << endl;
        return -1;
    }
```

```
    int degree = atoi(argv[1]);
    symbol z("z");
    cout << "H_" << degree << "(z)=="
         << HermitePoly(z,degree) << endl;
    return 0;
}
```

The compiled program takes an integer argument n from the command line and computes the nth Hermite polynomial in z using straightforward differentiation. The `normal()` function therein cancels the generators `HKer` in numerator and denominator. The result is returned and then printed like this: "`H_5(z)==120*z-160*z^3+32*z^5`". Note that operators `+`, `-`, `*` and `/` have been overloaded to allow construction of composite expressions while `pow(`b`,`e`)` has to be used for exponentiation (overloading operator `^` in C++ would lead to surprises, since it has lower precedence than `*`). Generally, in order to be easy to understand, GiNaC tries to follow the spirit of the standardized language C++ as far as possible.

As a package for long integers and arbitrary precision arithmetic GiNaC uses Bruno Haible's extraordinary CLN library (cf. 4.6.11). It is compatible in the design since it is also written in C++ (and it is compatible in the license).

GiNaC is free software, covered by the GNU General Public License. It is available from `http://www.ginac.de/`. It should run on any machine where CLN has successfully been ported to (we have positive results from Linux, Solaris and Tru64 using GCC as C++ compiler).

Christian Bauer, Alexander Frink, Richard Kreckel (Mainz)

References

1. Lars Brücher, Johannes Franzkowski, and Dirk Kreimer. XLoops: Automated Feynman diagram calculation. *Comput. Phys. Commun.*, 115:140–160, 1998.
2. Masaharu Goto. The CINT C/C++ Interpreter, 2000. `http://root.cern.ch/root/Cint.html`.
3. Wester, Michael J. *Computer Algebra Systems*. John Wiley, Chichester, 1999.

This article was processed using the LaTeX macro package with LMAMULT style