Program provides integer-to-binary conversion

BERT ERICKSON, FAYETTEVILLE, NY

Binary numbers rarely appear in applications of C or C++ programs, so any reference to converting from an integer to a binary number is usually relegated to a few simple examples in the appendix. However, when you re working with codes for communication systems, terms such as parity, checksum, distance, weight, and block codes are much easier to verify with a check solution when they are in binary form. C and C++ statements do use integers for manipulations that have binary implications. However, when the analysis gets down to the binary-number level, the conversion from integers is hard to find in the libraries supplied with the compiler. The cintbin and classicC functions in Listings 1 and 2 convert an integer in the main function to a binary number that remains available in the main function.

The ones and zeros in the elements of the array correspond to the location of bits in the customary binary number. You can compile the C++ cintbin version as listed. Readers who have an ANSI C compiler can use the program preceded by // in Listing 2. For long integers, refer to the revised edition of Microsoft C Programming for the PC by Robert Lafore. The first part of the listing is only a driver that has a call to the function and a printout for the binary number. You can use the bits in the binary number in any additional statements.

The first argument in the call should be 31 or less to provide some leading zeros but large enough to make sure the most significant bit is included. The temporary variable z and the return value provide some assurance that the result is valid. The statements in both functions are self-explanatory, so the only thing left to do is to compile one of the programs and enter 31 4,294,967,295 with a space after 31 to verify the 32-bit binary number 1111 1111 1111 1111 1111 1111 1111 1111. You can download the listings and the executable cintbin file from EDN s Web site, www.ednmag.com. At the registered-user area, go into the Software Center to download the files from DI-SIG, #2156. (DI #2156)

To Vote For This Design, Circle No. 349

LISTING 1—C++ INTEGER-TO-BINARY CONVERSION ROUTINE

```
// cintbin.cpp            a C++ function
// Converts an integer to a binary number
#include <iostream.h>
#include <math.h>
 int c[32];
main() {
 int n; unsigned long int x;
 unsigned long int cintbin(int, unsigned long int);   //Declare

 cout << "\n\tEnter max power of 2 desired (31 or less) and "
      << "the integer number\n" << "\t(4,294,967,295 or less) "
      << "with a space between them  ";
 cin >> n >> x;
 cout << "\tReturn x = " << cintbin(n, x)
      << " should equal the input value shown above\n   Bin # =";
 for (int k = n; k>= 0; k--) cout << ' ' << c[k];
  return 0;
 }

 unsigned long int cintbin(int n, unsigned long int x) //Define
     {
     unsigned long int y, z; z = x;
     for (int i = 0; i < n+1; i++) c[i] = 0;

     for (int j = n; j >= 0; j--)
  { y = int(pow(2,j)); if (x >= y)
    { c[j] = 1; x = x - y; }
  }
 return z;
```

LISTING 2—CLASSICC INTEGER-TO-BINARY-CONVERSION ROUTINE

```
//// classicC.cpp   a C function
//// Converts an integer to a binary number
//#include <stdio.h>
//#include <math.h>
// int c[32];
//main()  {
// int n; unsigned long int x;
// unsigned long int classicC(int, unsigned long int);   //Declare
//
// printf("\n\n\tEnter max power of 2 desired (31 or less) and ");
// printf("the integer number\n");
// printf("\t(4,294,967,295 or less) with a space between them  ");
// scanf("%d %lu",&n,&x);
// printf("\tReturn x = %lu",classicC(n,x));
// printf(" should equal the input value shown above\n   Bin # =");
// for (int k = n; k >= 0; k--)
//  printf(" %d",c[k]);
// return 0;
// }
//
// unsigned long classicC(int n, unsigned long x) //Define classicC
// {
//  unsigned long y, z;
//  int i, j;
//  z = x;
//  for (i = 0; i<n+1; i++)
//    c[i] = 0;
//  for (j=n; j>=0; j--)
//   {
//    y=pow(2,j);
//    if (x >= y)
//    {
//     c[j] = 1;
//     x = x - y;
//    }
//   }
//  return z;
 }
```