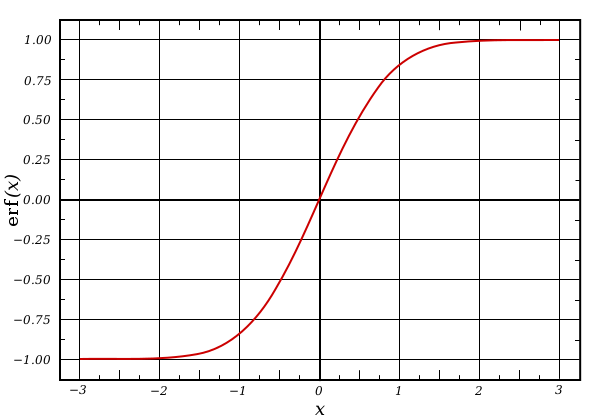
**CMATH Library- #include <cmath>**

**erf** - returns the *error function* value for x

* 
* **Code Snippet:**

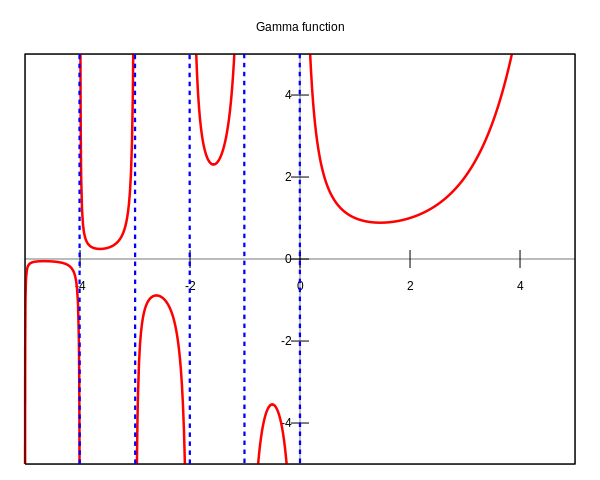
|  |
| --- |
| **double num = 0;**    **do {**  **cout << "Enter a number to compute the erf function for: " << flush;**  **cin >> num;**  **cout << "The answer is: " << erf(num) << endl;**  **}while (true);** |

* **Output:**

|  |
| --- |
| Enter a number to compute the erf function for: 1  The Answer is: 0.842701  Enter a number to compute the erf function for: .62  The Answer is: 0.619411  Enter a number to compute the erf function for: .126  The Answer is: 0.141427 |

* This function would be extremely handy when doing computations involving a normal distribution plot, or something else involving the erf function

**tgamma -** returns the *gamma function* of x

* 
* **Code Snippet:**

|  |
| --- |
| **#include <iostream>**  **#include <cmath>**  **using namespace std;**  **int main() {**  **double num = 1;**  **do {**  **cout << "Enter a number to compute the gamma function for: " << flush;**  **cin >> num;**  **cout << "The Answer is: " << tgamma(num) << endl;**  **} while (true);**  **return 0;**  **}** |

* **Output:**

|  |
| --- |
| **Enter a number to compute the gamma function for: 1**  **The Answer is: 1**  **Enter a number to compute the gamma function for: .5**  **The Answer is: 1.77245**  **Enter a number to compute the gamma function for: .25**  **The Answer is: 3.62561**  **Enter a number to compute the gamma function for: -.25**  **The Answer is: -4.90167** |

* This function would be handy in many different fields of statistics and mathematics, or some other place where a person wants to find a smooth curve that connect two points (x,y) given by y=(x-1)!

**CCTYPE Library- #include <cctype>**

**isgraph -** checks if a character has a graphical representation

* Code snippet

|  |
| --- |
| #include <iostream>  #include <cctype>  #include <iomanip>    using namespace std;    int main() {  double num = 1.02;  bool mexico = true;  int banana = 15;  char character = 'C';  cout << "This will tell you if the following variables have graphical components"  << endl;  cout << "Is a double number graphical (1.02): " << flush;  if (isgraph(num) == 0)  cout << "No\n";  else  cout << "Yes\n";  cout << "Is a boolean variable graphical (true/false): " << flush;  if (isgraph(mexico) == 0)  cout << "No\n";  else  cout << "Yes\n";  cout << "Is an integer graphical? " << flush;  if (isgraph(banana) == 0)  cout << "No\n";  else  cout << "Yes\n";  cout << "What about characters? are they graphical?: " << flush;  if (isgraph(character) == 0)  cout << "No" << endl;  else  cout << "Yes" << endl;  return 0;  } |

* Output

|  |
| --- |
| This will tell you if the following variables have graphical components  Is a double number graphical (1.02): No  Is a boolean variable graphical (true/false): No  Is an integer graphical? No  What about characters? are they graphical?: Yes |

* This is convenient for determining if a character in a file has a graphical component, this way you could add information to a file and later only print the graphical stuff, meaning you could potentially add a lot more information to the file and it wouldn’t impact the user’s experience

**isxdigit -** checks if a character is a hexadecimal digit

* Code snippet

|  |
| --- |
| #include <iostream>  #include <iomanip>  #include <cctype>  #include <cstdlib>  #include <cstring>  #include <cmath>  using namespace std;  int main() {  char string[80];  long int number = 0;  bool truthfull = true;  int length = 0;  int i = 0;  int temp = 0;  length = 0;  cout << "Enter a hexadecimal number: " << flush;  cin.get(string,80);  length = strlen(string);  number = strtoul(string, 0, 16);  i=0;  truthfull = true;  do {  if (!isxdigit(string[i]))  truthfull = false;  i++;  } while (i < length && truthfull);  if (truthfull) {  for (i; i > 0; i--) {  switch (string[i]) {  case '0':  temp = 0;  case '1':  temp = 1;  case '2':  temp = 2;  case '3':  temp = 3;  case '4':  temp = 4;  case '5':  temp = 5;  case '6':  temp = 6;  case '7':  temp = 7;  case '8':  temp = 8;  case '9':  temp = 9;  case 'a':  case 'A':  temp = 10;  case 'b':  case 'B':  temp = 11;  case 'c':  case 'C':  temp = 12;  case 'd':  case 'D':  temp = 13;  case 'e':  case 'E':  temp = 14;  case 'f':  case 'F':  temp = 15;  }  number += temp\*pow(16,i-length);  temp = 0;  }  cout << "The hexadecimal number for " << string << " is " << number << endl;  }  else {  cout << "Sorry. but that is not a hexadecimal number" << endl;  }  return 0;  } |

* Output

|  |
| --- |
| steven@maackia-amurensis:~$ ./test  Enter a hexadecimal number: test  Sorry. but that is not a hexadecimal number  steven@maackia-amurensis:~$ ./test  Enter a hexadecimal number: ffff  The hexadecimal number for ffff is 65535  steven@maackia-amurensis:~$ ./test  Enter a hexadecimal number: 0000  The hexadecimal number for 0000 is 0  steven@maackia-amurensis:~$ ./test  Enter a hexadecimal number: beef  The hexadecimal number for beef is 48879 |

* This function is handy in determining if a certain character is within the hexadecimal range, but it sort of sucks because there are less options for converting a number into or out of hexadecimal

**LIMITS Library- #include <limits>**

**is\_integer -** returns true if type is an integer

* Code snippet

|  |
| --- |
| #include <limits>  #include <iostream>  #include <iomanip>  using namespace std;  int main() {  int integer\_num = 15;  char character = 'A';  bool yes\_or\_no = true;  double aRealNumber = 1.001;  cout << "Is an int like '" << integer\_num << "' an integer? ";  if (numeric\_limits<int>::is\_integer)  cout << "\nYes\n";  else  cout << "\nNo\n";  cout << "What about a character like '" << character << "'? ";  if (numeric\_limits<char>::is\_integer)  cout << "\nYes\n";  else  cout << "\nNo\n";  cout << "What about boolean characters? ";  if (numeric\_limits<bool>::is\_integer)  cout << "\nYes\n";  else  cout << "\nNo\n";  cout << "What about double numbers like '" << aRealNumber << "'? ";  if (numeric\_limits<double>::is\_integer)  cout << "\nYes" << endl;  else  cout << "\nNo" << endl;  return 0;  } |

* Output

|  |
| --- |
| steven@maackia-amurensis:~$ ./test  Is an int like '15' an integer?  Yes  What about a character like 'A'?  Yes  What about boolean characters?  Yes  What about double numbers like '1.001'?  No |

* This function would be very convenient to determine if a class you wrote gave an output that was in the form of an integer

**max\_digits10** - displays the number of digits (in decimal base) required to ensure that values that differ are always differentiated

* Code snippet

|  |
| --- |
| #include <iostream>  #include <iomanip>  #include <limits>  using namespace std;  int main() {  cout << "How many digits does it take to uniquely"  << " represent all distinct value for bool?\n"  << numeric\_limits<bool>::max\_digits10  << "\nWhat about char?\n"  << numeric\_limits<char>::max\_digits10  << "\nWhat about int?\n"  << numeric\_limits<int>::max\_digits10  << "\nWhat about double?\n"  << numeric\_limits<double>::max\_digits10  << "\nWhat about long double?\n"  << numeric\_limits<long double>::max\_digits10  << "\nWhat about float?\n"  << numeric\_limits<float>::max\_digits10  << endl;  return 0;  } |

* Output

|  |
| --- |
| How many digits does it take to uniquely represent all distinct value for bool?  0  What about char?  0  What about int?  0  What about double?  17  What about long double?  21  What about float?  9 |

* This function would be convenient to determine the max number of unique digits needed to fill an particular class you may have written, and then determining the max number of digits able to be entered by the user so you don’t allocate 80 or so digits for a c-string, which is going to be later converted to a a double or something, instead you only need to allocate the number given by this function. This could potentially save a lot of space.