APPENDIX 4

Some Library Functions

The following lists are organized according to what the function is used for, rather than what library it is in. The function declaration gives the number and types of arguments as well as the type of the value returned. In most cases, the function declarations give only the type of the parameter and do not give a parameter name. (See the section "Alternate Form for Function Declarations" in Chapter 4 for an explanation of this kind of function declaration.)

Arithmetic Functions

Function Declaration	Description	Header File
`int abs(int);`	Absolute value	`cstdlib`
`long labs(long);`	Absolute value	`cstdlib`
`double fabs(double);`	Absolute value	`cmath`
`double sqrt(double);`	Square root	`cmath`
`double pow(double, double);`	Returns the first argument raised to the power of the second argument.	`cmath`
`double exp(double);`	Returns e (base of the natural logarithm) to the power of its argument.	`cmath`
`double log(double);`	Natural logarithm (ln)	`cmath`
`double log10(double);`	Base 10 logarithm	`cmath`
`double ceil(double);`	Returns the smallest integer that is greater than or equal to its argument.	`cmath`
`double floor(double);`	Returns the largest integer that is less than or equal to its argument.	`cmath`

Input and Output Member Functions

Form of a Function Call	Description	Header File
`Stream_Var.open(External_File_Name);`	Connects the file with the *External_File_Name* to the stream named by the *Stream_Var*. The *External_File_Name* is a string value.	`fstream`
`Stream_Var.fail( );`	Returns *true* if the previous operation (such as open) on the stream *Stream_Var* has failed.	`fstream` or `iostream`
`Stream_Var.close( );`	Disconnects the stream *Stream_Var* from the file it is connected to.	`fstream`
`Stream_Var.bad( );`	Returns *true* if the stream *Stream_Var* is corrupted.	`fstream` or `iostream`
`Stream_Var.eof( );`	Returns *true* if the program has attempted to read beyond the last character in the file connected to the input stream *Stream_Var*. Otherwise, it returns *false*.	`fstream` or `iostream`
`Stream_Var.get(Char_Variable);`	Reads one character from the input stream *Stream_Var* and sets the *Char_Variable* equal to this character. Does *not* skip over whitespace.	`fstream` or `iostream`
`Stream_Var.getline(String_Var, Max_Characters +1);`	One line of input from the stream *Stream_Var* is read, and the resulting string is placed in *String_Var*. If the line is more than *Max_Characters* long, only the first *Max_Characters are* read. The declared size of the *String_Var* should be *Max_Characters* +1 or larger.	`fstream` or `iostream`
`Stream_Var.peek( );`	Reads one character from the input stream *Stream_Var* and returns that character. But the character read is *not* removed from the input stream; the next read will read the same character.	`fstream` or `iostream`

Input and Output Member Functions *(continued)*

Form of a Function Call	Description	Header File
`Stream_Var.put (Char_Exp);`	Writes the value of the *Char_Exp* to the output stream *Stream_Var*.	`fstream` or `iostream`
`Stream_Var.putback (Char_Exp);`	Places the value of *Char_Exp* in the input stream *Stream_Var* so that that value is the next input value read from the stream. The file connected to the stream is not changed.	`fstream` or `iostream`
`Stream_Var.precision (Int_Exp);`	Specifies the number of digits output after the decimal point for floating-point values sent to the output stream *Stream_Var*.	`fstream` or `iostream`
`Stream_Var.width (Int_Exp);`	Sets the field width for the next value output to the stream *Stream_Var*.	`fstream` or `iostream`
`Stream_Var.setf(Flag);`	Sets flags for formatting output to the stream *Stream_Var*. See Display 6.5 for the list of possible flags.	`fstream` or `iostream`
`Stream_Var.unsetf(Flag);`	Unsets flags for formatting output to the stream *Stream_Var*. See Display 6.5 for the list of possible flags.	`fstream` or `iostream`

Character Functions

For all of these the actual type of the argument is `int`, but for most purposes you can think of the argument type as `char`. If the value returned is a value of type `int`, you must perform an explicit or implicit typecast to obtain a `char`.

Function Declaration	Description	Header File
`bool isalnum(char);`	Returns *true* if its argument satisfies either isalpha or isdigit. Otherwise, returns *false*.	cctype
`bool isalpha(char);`	Returns *true* if its argument is an upper- or lowercase letter. It may also return *true* for other arguments. The details are implementation dependent. Otherwise, returns *false*.	cctype
`bool isdigit(char);`	Returns *true* if its argument is a digit. Otherwise, returns *false*.	cctype
`bool ispunct(char);`	Returns *true* if its argument is a printable character that does not satisfy isalnum and is not whitespace. (These characters are considered punctuation characters.) Otherwise, returns *false*.	cctype
`bool isspace(char);`	Returns *true* if its argument is a whitespace character (such as blank, tab, or new line). Otherwise, returns *false*.	cctype
`bool iscntrl(char);`	Returns *true* if its argument is a control character. Otherwise, returns *false*.	cctype
`bool islower(char);`	Returns *true* if its argument is a lowercase letter. Otherwise, returns *false*.	cctype
`bool isupper(char);`	Returns *true* if its argument is an uppercase letter. Otherwise, returns *false*.	cctype
`int tolower(char);`	Returns the lowercase version of its argument. If there is no lowercase version, returns its argument unchanged.	cctype
`int toupper(char);`	Returns the uppercase version of its argument. If there is no uppercase version, returns its argument unchanged.	cctype

String Functions

Function Declaration	Description	Header File
`int atoi(const char a[]);`	Converts a string of characters to an integer.	`cstdlib`
`long atol(const char a[]);`	Converts a string of characters to a *long* integer.	`cstdlib`
`double atof(const char a[]);`	Converts a string of characters to a *double*.	`cstdlib`[1]
`strcat(`*String_Variable*`,` *String_Expression*`);`	Appends the value of the *String_Expression* to the end of the string in the *String_Variable*.	`cstring`
`strcmp(`*String_Exp1*`,` *String_Exp2*`)`	Returns *true* if the values of the two string expressions are different; otherwise, returns *false*.[2]	`cstring`
`strcpy(`*String_Variable*`,` *String_Expression*`);`	Changes the value of the *String_Variable* to the value of the *String_Expression*.	`cstring`
`strlen(`*String_Expression*`)`	Returns the length of the *String_Expression*.	`cstring`
`strncat(`*String_Variable*`,` *String_Expression*`,` *Limit*`);`	Same as `strcat` except that at most *Limit* characters are appended.	`cstring`
`strncmp(`*String_Exp1*`,` *String_Exp2*`,` *Limit*`)`	Same as `strcmp` except that at most *Limit* characters are compared.	`cstring`
`strncpy(`*String_Variable*`,` *String_Expression*`,` *Limit*`);`	Same as `strcpy` except that at most *Limit* characters are copied.	`cstring`
`strstr(`*String_Expression*`,` *Pattern*`)`	Returns a pointer to the first occurrence of the string *Pattern* in *String_Expression*. Returns the NULL pointer if the *Pattern* is not found.	`cstring`
`strchr(`*String_Expression*`,` *Character*`)`	Returns a pointer to the first occurrence of the *Character* in *String_Expression*. Returns the NULL pointer if *Character* is not found.	`cstring`
`strrchr(`*String_Expression*`,` *Character*`)`	Returns a pointer to the last occurrence of the *Character* in *String_Expression*. Returns the NULL pointer if *Character* is not found.	`cstring`

[1] Some implementations place it in `cmath`.
[2] Returns an integer that is less than zero, zero, or greater than zero according to whether *String_Exp1* is less than, equal to, or greater than *String_Exp2*, respectively. The ordering is lexicographic ordering.

Random Number Generator

Function Declaration	Description	Header File
`int random(int);`	The call `random(n)` returns a pseudorandom integer greater than or equal to 0 and less than or equal to n–1. (Not available in all implementations. If not available, then you must use rand.)	`cstdlib`
`int rand();`	The call `rand( )` returns a pseudorandom integer greater than or equal to 0 and less than or equal to RAND_MAX. RAND_MAX is a predefined integer constant that is defined in `cstdlib`. The value of RAND_MAX is implementation dependent but will be at least 32767.	`cstdlib`
`void srand(unsigned int);` `void srandom(unsigned int);` (The type `unsigned int` is an integer type that only allows nonnegative values. You can think of the argument type as `int` with the restriction that it must be nonnegative.)	Reinitializes the random number generator. The argument is the seed. Calling `srand` multiple times with the same argument will cause `rand` or `random` (whichever you use) to produce the same sequence of pseudorandom numbers. If `rand` or `random` is called without any previous call to `srand`, the sequence of numbers produced is the same as if there had been a call to `srand` with an argument of 1.	`cstdlib`

Trigonometric Functions

These functions use radians, not degrees.

Function Declaration	Description	Header File
`double acos(double);`	Arc cosine	cmath
`double asin(double);`	Arc sine	cmath
`double atan(double);`	Arc tangent	cmath
`double cos(double);`	Cosine	cmath
`double cosh(double);`	Hyperbolic cosine	cmath
`double sin(double);`	Sine	cmath
`double sinh(double);`	Hyperbolic sine	cmath
`double tan(double);`	Tangent	cmath
`double tanh(double);`	Hyperbolic tangent	cmath