Об авторе

Герберт Шилдт (Herbert Schildt) — выдающийся автор книг по программированию, общепризнанный авторитет в области программирования на языках C, C++, Java и приложений для Windows. Тираж его книг, переведенных на многие языки мира, составляет более 2,5 миллионов экземпляров. Он является автором многочисленных бестселлеров, среди которых наиболее известны такие издания, как *Полный справочник по C++, Teach Yourself C, Teach Yourself C++, C++ from the Ground Up, Windows 2000 Programming from the Ground Up, Java: The Complete Reference*. Г. Шилдт имеет степень магистра наук в области вычислительной математики, присвоенную ему Иллинойским университетом (University of Illinois).

Ждем ваших отзывов!

Вы, читатель этой книги, и есть главный ее критик и комментатор. Мы ценим ваше мнение и хотим знать, что было сделано нами правильно, что можно было сделать лучше и что еще вы хотели бы увидеть изданным нами. Нам интересно услышать и любые другие замечания, которые вам хотелось бы высказать в наш адрес.

Мы ждем ваших комментариев и надеемся на них. Вы можете прислать нам бумажное или электронное письмо либо просто посетить наш Web-сервер и оставить свои замечания там. Одним словом, любым удобным для вас способом дайте нам знать, нравится или нет вам эта книга, а также выскажите свое мнение о том, как сделать наши книги более интересными для вас.

Посылая письмо или сообщение, не забудьте указать название книги и ее авторов, а также ваш обратный адрес. Мы внимательно ознакомимся с вашим мнением и обязательно учтем его при отборе и подготовке к изданию последующих книг. Наши координаты:

E-mail: `info@williamspublishing.com`
WWW: `http://www.williamspublishing.com`

Информация для писем:

из России: 115419, Москва, а/я 783
из Украины: 03150, Киев, а/я 152

Содержание

Предисловие ... 26
Часть I. Основы языка C ... 29
Глава 1. Обзор возможностей языка C ... 31
Краткая история развития языка C ... 32
C – язык среднего уровня ... 32
Язык C хорошо структурирован ... 34
Язык C создан для программистов ... 35
Компиляторы и интерпретаторы ... 36
Структура программы на языке C ... 37
Библиотеки и компоновка ... 38
Раздельная компиляция ... 39
Компиляция программы на языке C ... 39
Карта памяти программы на языке C ... 39
Сравнительная характеристика языков C и C++ ... 40
Словарь терминов ... 41
Глава 2. Выражения ... 43
Базовые типы данных ... 44
Модификация базовых типов ... 45
Имена переменных ... 46
Переменные ... 47
Где объявляются переменные ... 47
Локальные переменные ... 47
Формальные параметры функции ... 50
Глобальные переменные ... 51
Четыре типа областей видимости ... 52
Квалификаторы типа ... 52
Квалификатор `const` ... 53
Квалификатор `volatile` ... 54
Спецификаторы класса памяти ... 54
Спецификатор `extern` ... 55
Спецификатор `static` ... 57
Спецификатор `register` ... 59
Инициализация переменных ... 60
Константы ... 60
Шестнадцатеричные и восьмеричные константы ... 61
Строковые константы ... 61
Специальные символьные константы ... 62
Операции ... 62
Оператор присваивания ... 63
Арифметические операции ... 65
Операции увеличения (инкремента) и уменьшения (декремента) ... 66
Операции сравнения и логические операции ... 67
Поразрядные операции ... 69
Операция ? ... 72
Операции получения адреса (&) и раскрытия ссылки (*) ... 73

Операция определения размера sizeof ... 74
Оператор последовательного вычисления: оператор “запятая” ... 75
Оператор доступа к члену структуры (оператор . (точка)) и оператор доступа через указатель -> (оператор стрелка) ... 75
Операторы [ ] и ( ) ... 76
Сводка приоритетов операций ... 76
Выражения ... 77
Порядок вычислений ... 77
Преобразование типов в выражениях ... 77
Явное преобразование типов: операция приведения типов ... 78
Пробелы и круглые скобки ... 79
Глава 3. Операторы ... 81
Логические значения ИСТИНА (True) и ЛОЖЬ (False) в языке C ... 82
Условные операторы ... 82
Оператор `if` ... 82
Вложенные условные операторы if ... 84
Лестница if-else-if ... 85
Оператор “`?`”, альтернативный условному ... 86
Условное выражение ... 88
Оператор выбора — `switch` ... 89
Вложенные операторы `switch` ... 92
Операторы цикла ... 92
Цикл `for` ... 92
Варианты цикла `for` ... 93
Бесконечный цикл ... 96
Цикл `for` без тела цикла ... 97
Объявление переменных внутри цикла ... 97
Цикл `while` ... 98
Цикл `do-while` ... 100
Операторы перехода ... 101
Оператор `return` ... 101
Оператор `goto` ... 101
Оператор `break` ... 102
Функция `exit()` ... 103
Оператор `continue` ... 104
Оператор-выражение ... 105
Блок операторов ... 105
Глава 4. Массивы и строки ... 107
Одномерные массивы ... 108
Создание указателя на массив ... 109
Передача одномерного массива в функцию ... 110
Строки ... 110
Двухмерные массивы ... 112
Массивы строк ... 115
Многомерные массивы ... 116
Индексация указателей ... 117
Инициализация массивов ... 118
Инициализация безразмерных массивов ... 120
Массивы переменной длины ... 121

Приемы использования массивов и строк на примере игры в крестики-нолики ... 121
Глава 5. Указатели ... 125
Что такое указатели ... 126
Указательные переменные ... 126
Операции для работы с указателями ... 127
Указательные выражения ... 127
Присваивание указателей ... 127
Преобразование типа указателя ... 128
Адресная арифметика ... 129
Сравнение указателей ... 130
Указатели и массивы ... 132
Массивы указателей ... 133
Многоуровневая адресация ... 134
Инициализация указателей ... 135
Указатели на функции ... 137
Функции динамического распределения ... 140
Динамическое выделение памяти для массивов ... 141
Указатели с квалификатором restrict ... 143
Трудности при работе с указателями ... 144
Глава 6. Функции ... 147
Общий вид функции ... 148
Что такое область действия функции ... 148
Аргументы функции ... 149
Вызовы по значению и по ссылке ... 149
Вызов по ссылке ... 150
Вызов функций с помощью массивов ... 152
Аргументы функции main(): argv и argc ... 154
Оператор return ... 157
Возврат из функции ... 157
Возврат значений ... 158
Возвращаемые указатели ... 160
Функции типа void ... 161
Что возвращает функция main()? ... 161
Рекурсия ... 161
Прототипы функций ... 163
Старомодные объявления функций ... 165
Прототипы стандартных библиотечных функций ... 166
Объявление списков параметров переменной длины ... 166
Правило “неявного int” ... 166
Старомодные и современные объявления параметров функций ... 167
Ключевое слово inline ... 168
Глава 7. Структуры, объединения, перечисления и декларация typedef ... 169
Структуры ... 170
Доступ к членам структуры ... 172
Присваивание структур ... 172
Массивы структур ... 173
Пример со списком рассылки ... 173
Передача структур функциям ... 179
Передача членов структур функциям ... 179
Передача целых структур функциям ... 179

Указатели на структуры ... 181
Объявление указателя на структуру ... 181
Использование указателей на структуры ... 181
Массивы и структуры внутри структур ... 183
Объединения ... 184
Битовые поля ... 187
Перечисления ... 188
Важное различие между C и C++ ... 190
Использование sizeof для обеспечения переносимости ... 191
Средство typedef ... 192
Глава 8. Ввод/вывод на консоль ... 195
Чтение и запись символов ... 197
Трудности использования getchar() ... 197
Альтернативы getchar() ... 198
Чтение и запись строк ... 199
Форматный ввод/вывод на консоль ... 201
printf() ... 201
Вывод символов ... 202
Вывод чисел ... 202
Отображение адреса ... 204
Спецификатор преобразования %n ... 204
Модификаторы формата ... 205
Модификатор минимальной ширины поля ... 205
Модификатор точности ... 206
Выравнивание вывода ... 207
Обработка данных других типов ... 207
Модификаторы * и # ... 208
scanf() ... 208
Спецификаторы преобразования ... 209
Ввод чисел ... 209
Ввод целых значений без знака ... 210
Чтение одиночных символов с помощью scanf() ... 210
Чтение строк ... 210
Ввод адреса ... 211
Спецификатор %n ... 211
Использование набора сканируемых символов ... 211
Пропуск лишних разделителей ... 212
Символы в управляющей строке, не являющиеся разделителями ... 212
Функции scanf() необходимо передавать адреса ... 213
Модификаторы формата ... 213
Подавление ввода ... 214
Глава 9. Файловый ввод/вывод ... 215
Файловый ввод/вывод в C и C++ ... 216
Файловый ввод/вывод в стандартном C и UNIX ... 216
Потоки и файлы ... 216
Потоки ... 217
Файлы ... 217
Основы файловой системы ... 218
Указатель файла ... 219
Открытие файла ... 219
Закрытие файла ... 220
Запись символа ... 221

Чтение символа ... 221
Использование fopen(), getc(), putc() и fclose() ... 222
Использование feof() ... 223
Ввод/вывод строк: fputs() и fgets() ... 224
Функция rewind() ... 225
Функция ferror() ... 226
Стирание файлов ... 227
Дозапись потока ... 228
Функции fread() и fwrite() ... 228
Использование fread() и fwrite() ... 228
Пример со списком рассылки ... 229
Ввод/вывод при прямом доступе: функция fseek() ... 234
Функции fprintf() и fscanf() ... 235
Стандартные потоки ... 236
Связь с консольным вводом/выводом ... 238
Перенаправление стандартных потоков: функция freopen() ... 238
Глава 10. Препроцессор и комментарии ... 241
Препроцессор ... 242
Директива #define ... 242
Определение макросов с формальными параметрами ... 243
Директива #error ... 244
Директива #include ... 245
Директивы условной компиляции ... 245
Директивы #if, #else, #elif и #endif ... 245
Директивы #ifdef и #ifndef ... 247
Директива #undef ... 248
Использование defined ... 249
Директива #line ... 249
Директива #pragma ... 250
Операторы препроцессора # и ## ... 250
Имена предопределенных макрокоманд ... 251
Комментарии ... 251
Однострочные комментарии ... 252
Часть II. Стандарт C99 ... 253
Глава 11. C99 ... 255
Сравнение C99 с C89. Общее впечатление ... 256
Новые возможности ... 256
Удаленные средства ... 257
Измененные средства ... 257
Указатели, определенные с квалификаторами типа restrict ... 257
Ключевое слово inline ... 258
Новые встроенные типы данных ... 259
_Bool ... 259
_Complex и _Imaginary ... 260
Типы целых данных long long ... 260
Расширение массивов ... 260
Массивы переменной длины ... 261
Использование квалификаторов типов в объявлении массива ... 261
Однострочные комментарии ... 262
Распределение кода и объявлений ... 262
Изменения препроцессора ... 263

Переменные списки аргументов ... 263
Оператор _Pragma ... 263
Встроенные прагмы ... 263
Новые встроенные макросы ... 264
Объявление переменных внутри цикла for ... 264
Составные литералы ... 265
Массивы с переменными границами в качестве членов структур ... 266
Назначенные инициализаторы ... 266
Новые возможности семейства функций printf() и scanf() ... 267
Новые библиотеки C99 ... 267
Зарезервированный идентификатор __func__ ... 268
Расширение граничных значений трансляции ... 268
Неявный int больше не поддерживается ... 269
Удалены неявные объявления функций ... 270
Ограничения на return ... 270
Расширенные целые типы ... 270
Изменения в правилах продвижения целых типов ... 271

Часть III. Стандартная библиотека ... 273

Глава 12. Редактирование связей, использование библиотек и заголовков ... 275

Редактор связей ... 276
Раздельная компиляция ... 276
Переместимые коды и абсолютные коды ... 277
Редактирование связей с оверлеями ... 277
Связывание с динамически подсоединяемыми библиотеками (DLL) ... 278
Стандартная библиотека C ... 279
Библиотечные файлы и объектные файлы ... 279
Заголовки ... 279
Макросы в заголовках ... 281
Переопределение библиотечных функций ... 281

Глава 13. Функции ввода/вывода ... 283

Функция clearerr ... 284
Пример ... 284
Зависимые функции ... 285
Функция fclose ... 285
Пример ... 286
Зависимые функции ... 286
Функция feof ... 286
Пример ... 286
Зависимые функции ... 286
Функция ferror ... 287
Пример ... 287
Зависимые функции ... 287
Функция fflush ... 287
Пример ... 287
Зависимые функции ... 288
Функция fgetc ... 288
Пример ... 288
Зависимые функции ... 288
Функция fgetpos ... 289
Пример ... 289
Зависимые функции ... 289

Функция fgets ... 289
Пример ... 289
Зависимые функции ... 290
Функция fopen ... 290
Пример ... 291
Зависимые функции ... 291
Функция fprintf ... 291
Пример ... 292
Зависимые функции ... 292
Функция fputc ... 292
Пример ... 293
Зависимые функции ... 293
Функция fputs ... 293
Пример ... 293
Зависимые функции ... 293
Функция fread ... 293
Пример ... 294
Зависимые функции ... 294
Функция freopen ... 294
Пример ... 295
Зависимые функции ... 295
Функция fscanf ... 295
Пример ... 296
Зависимые функции ... 296
Функция fseek ... 296
Пример ... 296
Зависимые функции ... 297
Функция fsetpos ... 297
Пример ... 297
Зависимые функции ... 297
Функция ftell ... 298
Пример ... 298
Зависимые функции ... 298
Функция fwrite ... 298
Пример ... 298
Зависимые функции ... 299
Функция getc ... 299
Пример ... 299
Зависимые функции ... 300
Функция getchar ... 300
Пример ... 300
Зависимые функции ... 300
Функция gets ... 300
Пример ... 301
Зависимые функции ... 301
Функция perror ... 301
Пример ... 302
Функция printf ... 302
Модификаторы формата функции printf(), добавленные стандартом C99 ... 304
Пример ... 305
Зависимые функции ... 305
Функция putc ... 305
Пример ... 305

Зависимые функции ... 305
Функция putchar ... 305
Пример ... 306
Зависимые функции ... 306
Функция puts ... 306
Пример ... 306
Зависимые функции ... 306
Функция remove ... 307
Пример ... 307
Зависимые функции ... 307
Функция rename ... 307
Пример ... 307
Зависимые функции ... 308
Функция rewind ... 308
Пример ... 308
Зависимые функции ... 308
Функция scanf ... 308
Модификаторы формата, добавленные к функции scanf() Стандартом C99 ... 311
Пример ... 311
Зависимые функции ... 312
Функция setbuf ... 312
Пример ... 312
Зависимые функции ... 312
Функция setvbuf ... 312
Пример ... 313
Зависимые функции ... 313
Функция snprintf ... 313
Зависимые функции ... 313
Функция sprintf ... 313
Пример ... 314
Зависимые функции ... 314
Функция sscanf ... 314
Пример ... 314
Зависимые функции ... 315
Функция tmpfile ... 315
Пример ... 315
Зависимые функции ... 315
Функция tmpnam ... 315
Пример ... 316
Зависимые функции ... 316
Функция ungetc ... 316
Пример ... 316
Зависимые функции ... 317
Функции vprintf, vfprintf, vsprintf и vsnprintf ... 317
Пример ... 317
Зависимые функции ... 318
Функции vscanf, vfscanf и vsscanf ... 318
Зависимые функции ... 318
Глава 14. Строковые и символьные функции ... 319
Функция isalnum ... 320
Пример ... 320
Зависимые функции ... 321

Функция isalpha 321
Пример 321
Зависимые функции 321
Функция isblank 322
Пример 322
Зависимые функции 322
Функция iscntrl 322
Пример 322
Зависимые функции 323
Функция isdigit 323
Пример 323
Зависимые функции 323
Функция isgraph 324
Пример 324
Зависимые функции 324
Функция islower 324
Пример 324
Зависимые функции 325
Функция isprint 325
Пример 325
Зависимые функции 325
Функция ispunct 325
Пример 326
Зависимые функции 326
Функция isspace 326
Пример 326
Зависимые функции 327
Функция isupper 327
Пример 327
Зависимые функции 327
Функция isxdigit 327
Пример 328
Зависимые функции 328
Функция memchr 328
Пример 328
Еще один пример 329
Зависимые функции 329
Функция memcmp 329
Пример 330
Зависимые функции 330
Функция memcpy 330
Пример 331
Зависимые функции 331
Функция memmove 331
Пример 331
Зависимые функции 332
Функция memset 332
Пример 332
Зависимые функции 332
Функция strcat 332
Пример 332
Зависимые функции 333
Функция strchr 333

Пример ... 333
Зависимые функции ... 333
Функция strcmp ... 333
Пример ... 334
Зависимые функции ... 334
Функция strcoll ... 334
Пример ... 334
Зависимые функции ... 334
Функция strcpy ... 335
Пример ... 335
Зависимые функции ... 335
Функция strcspn ... 335
Пример ... 335
Зависимые функции ... 335
Функция strerror ... 336
Пример ... 336
Функция strlen ... 336
Пример ... 336
Зависимые функции ... 336
Функция strncat ... 336
Пример ... 337
Зависимые функции ... 337
Функция strncmp ... 337
Пример ... 337
Зависимые функции ... 338
Функция strncpy ... 338
Пример ... 338
Зависимые функции ... 338
Функция strpbrk ... 339
Пример ... 339
Зависимые функции ... 339
Функция strrchr ... 339
Пример ... 339
Зависимые функции ... 340
Функция strspn ... 340
Пример ... 340
Зависимые функции ... 340
Функция strstr ... 340
Пример ... 340
Зависимые функции ... 341
Функция strtok ... 341
Пример ... 341
Зависимые функции ... 342
Функция strxfrm ... 342
Пример ... 342
Зависимые функции ... 342
Функция tolower ... 342
Пример ... 342
Зависимые функции ... 343
Функция toupper ... 343
Пример ... 343
Зависимые функции ... 343

Глава 15. Математические функции.....345
Семейство функций acos.....347
Пример.....348
Зависимые функции.....348
Семейство функций acosh.....348
Зависимые функции.....348
Семейство функций asin.....348
Пример.....349
Зависимые функции.....349
Семейство функций asinh.....349
Зависимые функции.....349
Семейство функций atan.....349
Пример.....350
Зависимые функции.....350
Семейство функций atanh.....350
Зависимые функции.....350
Семейство функций atan2.....350
Пример.....351
Зависимые функции.....351
Семейство функций cbrt.....351
Пример.....351
Зависимые функции.....351
Семейство функций ceil.....351
Пример.....352
Зависимые функции.....352
Семейство функций copysign.....352
Зависимые функции.....352
Семейство функций cos.....352
Пример.....352
Зависимые функции.....353
Семейство функций cosh.....353
Пример.....353
Зависимые функции.....353
Семейство функций erf.....353
Зависимые функции.....354
Семейство функций erfc.....354
Зависимые функции.....354
Семейство функций exp.....354
Пример.....355
Зависимые функции.....355
Семейство функций exp2.....355
Зависимые функции.....355
Семейство функций expm1.....355
Зависимые функции.....355
Семейство функций fabs.....355
Пример.....356
Зависимые функции.....356
Семейство функций fdim.....356
Зависимые функции.....356
Семейство функций floor.....356
Пример.....356
Зависимые функции.....357

Семейство функций fma ... 357
Зависимые функции ... 357
Семейство функций fmax ... 357
Зависимые функции ... 357
Семейство функций fmin ... 357
Зависимые функции ... 357
Семейство функций fmod ... 358
Пример ... 358
Зависимые функции ... 358
Семейство функций frexp ... 358
Пример ... 359
Зависимые функции ... 359
Семейство функций hypot ... 359
Зависимые функции ... 359
Семейство функций ilogb ... 359
Зависимые функции ... 359
Семейство функций ldexp ... 359
Пример ... 360
Зависимые функции ... 360
Семейство функций lgamma ... 360
Зависимые функции ... 360
Семейство функций llrint ... 360
Зависимые функции ... 360
Семейство функций llround ... 361
Зависимые функции ... 361
Семейство функций log ... 361
Пример ... 361
Зависимые функции ... 361
Семейство функций log1p ... 362
Зависимые функции ... 362
Семейство функций log10 ... 362
Пример ... 362
Зависимые функции ... 362
Семейство функций log2 ... 363
Зависимые функции ... 363
Семейство функций logb ... 363
Зависимые функции ... 363
Семейство функций lrint ... 363
Зависимые функции ... 363
Семейство функций lround ... 364
Зависимые функции ... 364
Семейство функций modf ... 364
Пример ... 364
Зависимые функции ... 364
Семейство функций nan ... 364
Зависимые функции ... 365
Семейство функций nearbyint ... 365
Зависимые функции ... 365
Семейство функций nextafter ... 365
Зависимые функции ... 365
Семейство функций nexttoward ... 365
Зависимые функции ... 365
Семейство функций pow ... 366

Пример ... 366
Зависимые функции ... 366
Семейство функций remainder ... 366
Зависимые функции ... 366
Семейство функций remquo ... 367
Зависимые функции ... 367
Семейство функций rint ... 367
Зависимые функции ... 367
Семейство функций round ... 367
Зависимые функции ... 367
Семейство функций scalbln ... 368
Зависимые функции ... 368
Семейство функций scalbn ... 368
Зависимые функции ... 368
Семейство функций sin ... 368
Пример ... 368
Зависимые функции ... 369
Семейство функций sinh ... 369
Пример ... 369
Зависимые функции ... 369
Семейство функций sqrt ... 370
Пример ... 370
Зависимые функции ... 370
Семейство функций tan ... 370
Пример ... 370
Зависимые функции ... 371
Семейство функций tanh ... 371
Пример ... 371
Зависимые функции ... 371
Семейство функций tgamma ... 371
Зависимые функции ... 371
Семейство функций trunc ... 372
Зависимые функции ... 372
Глава 16. Функции времени, даты и локализации ... 373
Функция asctime ... 374
Пример ... 374
Зависимые функции ... 375
Функция clock ... 375
Пример ... 375
Зависимые функции ... 375
Функция ctime ... 375
Пример ... 376
Зависимые функции ... 376
Функция difftime ... 376
Пример ... 376
Зависимые функции ... 376
Функция gmtime ... 377
Пример ... 377
Зависимые функции ... 377
Функция localeconv ... 377
Пример ... 379
Родственная функция ... 379

Функция localtime ... 379
Пример ... 380
Зависимые функции ... 380
Функция mktime ... 380
Пример ... 380
Зависимые функции ... 381
Функция setlocale ... 381
Пример ... 381
Зависимые функции ... 382
Функция strftime ... 382
Пример ... 383
Зависимые функции ... 384
Функция time ... 384
Пример ... 384
Зависимые функции ... 384
Глава 17. Функции динамического распределения памяти ... 385
Функция calloc ... 386
Пример ... 386
Зависимые функции ... 386
Функция free ... 387
Пример ... 387
Зависимые функции ... 387
Функция malloc ... 387
Пример ... 388
Зависимые функции ... 388
Функция realloc ... 388
Пример ... 389
Зависимые функции ... 389
Глава 18. Служебные функции ... 391
Функция abort ... 392
Пример ... 392
Зависимые функции ... 392
Функция abs ... 393
Пример ... 393
Зависимая функция ... 393
Функция-макрос assert ... 393
Пример ... 393
Зависимая функция ... 394
Функция atexit ... 394
Пример ... 394
Зависимые функции ... 394
Функция atof ... 394
Пример ... 395
Зависимые функции ... 395
Функция atoi ... 395
Пример ... 395
Зависимые функции ... 396
Функция atol ... 396
Пример ... 396
Зависимые функции ... 396
Функция atoll ... 396
Зависимые функции ... 396

Функция bsearch 397
Пример 397
Зависимая функция 398
Функция div 398
Пример 398
Зависимые функции 398
Функция exit 398
Пример 399
Зависимые функции 399
Функция _Exit 399
Зависимые функции 399
Функция getenv 399
Пример 400
Зависимая функция 400
Функция labs 400
Пример 400
Зависимые функции 400
Функция llabs 400
Зависимые функции 401
Функция ldiv 401
Пример 401
Зависимые функции 401
Функция lldiv 401
Зависимые функции 401
Функция longjmp 402
Пример 402
Зависимая функция 403
Функция mblen 403
Пример 403
Зависимые функции 403
Функция mbstowcs 403
Пример 403
Зависимые функции 403
Функция mbtowc 404
Пример 404
Зависимые функции 404
Функция qsort 404
Пример 405
Зависимая функция 405
Сортировка в убывающем порядке 405
Функция raise 405
Зависимая функция 406
Функция rand 406
Пример 406
Зависимая функция 406
Функция setjmp 406
Зависимая функция 407
Функция signal 407
Зависимая функция 407
Функция srand 407
Пример 408
Зависимая функция 408
Функция strtod 408

Пример ... 409
Зависимые функции ... 409
Функция strtof ... 409
Зависимые функции ... 409
Функция strtol ... 410
Пример ... 410
Зависимые функции ... 410
Функция strtold ... 411
Зависимые функции ... 411
Функция strtoll ... 411
Зависимые функции ... 411
Функция strtoul ... 411
Пример ... 412
Зависимые функции ... 412
Функция strtoull ... 412
Зависимые функции ... 412
Функция system ... 413
Пример ... 413
Зависимая функция ... 413
Функции-макросы va_arg, va_start, va_end и va_copy ... 413
Пример ... 414
Зависимая функция ... 415
Функция wcstombs ... 415
Зависимые функции ... 415
Функция wctomb ... 415
Зависимые функции ... 415
Глава 19. Функции обработки двухбайтовых символов ... 417
Функции классификации двухбайтовых символов ... 418
Функции ввода-вывода двухбайтовых символов ... 420
Функции для операций над строками двухбайтовых символов ... 421
Преобразование строк двухбайтовых символов ... 422
Функции для обработки массивов двухбайтовых символов ... 423
Функции для преобразования многобайтовых и двухбайтовых символов ... 424
Глава 20. Библиотечные средства, добавленные в версии C99 ... 425
Библиотека поддержки арифметических операций с комплексными числами ... 426
Библиотека поддержки среды вычислений с плавающей точкой ... 430
Заголовок <stdint.h> ... 431
Функции для преобразования формата целочисленных значений ... 432
Математические макросы обобщенного типа ... 433
Заголовок <stdbool.h> ... 434
Часть IV. Алгоритмы и приложения ... 435
Глава 21. Сортировка и поиск ... 437
Сортировка ... 438
Классы алгоритмов сортировки ... 439
Оценка алгоритмов сортировки ... 439
Пузырьковая сортировка ... 440
Сортировка посредством выбора ... 443
Сортировка вставками ... 444
Улучшенные алгоритмы сортировки ... 445
Сортировка Шелла ... 446
Быстрая сортировка ... 448

Выбор метода сортировки ... 451
Сортировка других структур данных ... 451
Сортировка строк ... 451
Сортировка структур ... 453
Сортировка дисковых файлов с произвольной выборкой ... 454
Поиск ... 457
Методы поиска ... 457
Последовательный поиск ... 457
Двоичный поиск ... 458
Глава 22. Очереди, стеки, связанные списки и деревья ... 459
Очереди ... 460
Циклическая очередь ... 464
Стеки ... 467
Связанные списки ... 471
Односвязные списки ... 471
Двусвязные списки ... 476
Пример списка рассылки ... 479
Двоичные деревья ... 484
Глава 23. Разреженные массивы ... 493
Зачем нужны разреженные массивы? ... 494
Представление разреженного массива в виде связанного списка ... 495
Анализ метода представления в виде связанного списка ... 498
Представление разреженного массива в виде двоичного дерева ... 498
Анализ метода представления в виде двоичного дерева ... 500
Представление разреженного массива в виде массива указателей ... 500
Анализ метода представления разреженного массива в виде массива указателей ... 502
Хэширование ... 503
Анализ метода хэширования ... 506
Выбор метода ... 507
Глава 24. Синтаксический разбор и вычисление выражений ... 509
Выражения ... 510
Разбиение выражения на лексемы ... 511
Разбор выражений ... 514
Простая программа синтаксического анализа выражений ... 515
Работа с переменными в анализаторе ... 520
Проверка синтаксиса в рекурсивном нисходящем анализаторе ... 526
Глава 25. Решение задач с помощью искусственного интеллекта ... 529
Представление и терминология ... 530
Комбинаторные взрывы ... 532
Методы поиска ... 534
Оценка поиска ... 535
Представление в виде графа ... 536
Поиск в глубину ... 537
Анализ поиска в глубину ... 545
Полный перебор, или поиск в ширину ... 546
Анализ поиска в ширину ... 547
Добавление эвристики ... 548
Поиск методом наискорейшего подъема ... 548
Анализ наискорейшего подъема ... 554
Поиск с использованием частичного пути минимальной стоимости ... 554

Анализ поиска с использованием частичного пути минимальной стоимости.. 555
Выбор метода поиска ... 556
Поиск нескольких решений ... 556
Удаление путей ... 557
Удаление вершин ... 557
Поиск “оптимального” решения ... 562
И снова возвращаемся к поиску потерянных ключей ... 567
Часть V. Разработка программ с помощью C ... 571
Глава 26. Создание скелета приложения для Windows 2000 ... 573
Общая картина специфики программирования для Windows 2000 ... 574
Модель рабочего стола ... 575
Мышь ... 575
Пиктограммы, растровые изображения и другая графика ... 575
Меню, средства управления и диалоговые окна ... 576
Интерфейс прикладного программирования Win32 ... 576
Компоненты окна ... 577
Взаимодействие прикладных программ с Windows ... 578
Базовые концепции функционирования приложений для Windows 2000 ... 578
WinMain() ... 579
Процедура окна ... 579
Классы окон ... 579
Цикл обработки сообщений ... 580
Типы данных Windows ... 580
Скелет программы для Windows 2000 ... 580
Определение класса окна ... 583
Создание окна ... 585
Цикл обработки сообщений ... 587
Функция окна ... 588
Файл описания больше не нужен ... 589
Соглашения об именовании ... 589
Глава 27. Проектирование программ с помощью C ... 591
Проектирование сверху вниз ... 592
Структурирование программы ... 592
Выбор структуры данных ... 593
“Пуленепробиваемые” функции ... 594
Использование программы MAKE ... 597
Использование макросов в MAKE ... 600
Применение интегрированной среды разработки ... 601
Глава 28. Производительность, переносимость и отладка ... 603
Эффективность ... 604
Операции увеличения и уменьшения ... 604
Применение регистровых переменных ... 605
Указатели вместо индексации массива ... 606
Применение функций ... 606
Перенос программ ... 610
Использование #define ... 610
Зависимость от операционной системы ... 611
Различия в размерах данных ... 611
Отладка ... 611
Ошибки очередности вычисления ... 611

Проблемы с указателями ... 612
Интерпретация синтаксических ошибок ... 614
Ошибки, вызванные "потерей" единицы ... 615
Ошибки из-за нарушения границ ... 616
Пропуск прототипов функций ... 617
Ошибки при задании аргументов ... 618
Переполнение стека ... 619
Применение отладчика ... 619
Теория отладки в общих чертах ... 619

Часть VI. Интерпретатор языка C ... 621

Глава 29. Интерпретатор языка C ... 623

Практическое значение интерпретаторов ... 624
Определение языка Little C ... 625
Ограничения языка Little C ... 626
Интерпретация структурированного языка ... 628
Неформальная теория языка C ... 628
Выражения языка C ... 629
Определение значения выражения ... 630
Синтаксический анализатор выражений ... 631
Синтаксический разбор исходного текста программы ... 631
Рекурсивный нисходящий синтаксический анализатор Little C ... 636
Интерпретатор Little C ... 647
Предварительный проход интерпретатора ... 648
Функция main() ... 650
Функция interp_block() ... 651
Обработка локальных переменных ... 664
Вызов функций, определенных пользователем ... 665
Присваивание значений переменным ... 668
Выполнение оператора if ... 669
Обработка цикла while ... 670
Обработка цикла do-while ... 671
Цикл for ... 671
Библиотечные функции Little C ... 672
Компиляция и компоновка интерпретатора Little C ... 675
Демонстрация Little C ... 675
Усовершенствование интерпретатора Little C ... 679
Расширение Little C ... 680
Добавление новых средств в язык Little C ... 680
Создание дополнительных средств программирования ... 680
Предметный указатель ... 681

Предисловие

Это четвертое издание книги *C: The Complete Reference (Полный справочник по C)*. Со времен третьего издания в области программирования произошло много прогрессивных изменений, в частности, получили широкое распространение Internet и World Wide Web, был изобретен язык Java, а C++ был стандартизирован. Также был создан новый стандарт C, названный C99. Несмотря на то, что Стандарт C99 пока не завоевал всеобщего признания, его создание стало одним из самых выдающихся событий в области программирования за последние пять лет. В условиях стремительного развития компьютерных технологий порой нелегко сразу определить фундаментальные элементы, на которых строится будущее этих технологий. Именно таким основополагающим элементом является язык C. Во всем мире значительная часть текстов программ написана на этом языке. На его основе построен язык C++, а синтаксис языка C является фундаментом языка Java. Если бы язык C был всего лишь отправной точкой для других языков программирования, это был бы интересный, но мертвый язык. К счастью, это не так. Сегодня язык C не менее актуален, чем во время своего создания. Как будет видно из дальнейшего изложения, Стандарт C99 содержит такие новые перспективные конструкции, благодаря которым язык C по праву считается одним из самых прогрессивных в области программирования. Несмотря на большое распространение "потомков" языка C (Java и C++), значение самого C попрежнему остается первостепенным.

В создании Стандарта C99 участвовали наиболее выдающиеся специалисты по языкам программирования, среди них такие, как Рекс Джашке (Rex Jaeschke), Джим Томас (Jim Thomas), Том МакДональд (Tom MacDonald) и Джон Бенито (John Benito). Как член комиссии по стандартизации я наблюдал процесс создания стандарта и участвовал в дискуссиях по поводу каждого нововведения. В результате ежедневного обмена идеями и мнениями по электронной почте между участниками этого процесса, находящимися практически во всех странах мира, удалось выработать единую концепцию, что и привело в конечном итоге к значительному усовершенствованию языка C.

Надо признать, что, работая над первым изданием этой книги, я не предполагал столь быстрого роста достижений в области программирования, хотя некоторые из них, например, большой успех C++, были предопределены уже с самого начала. Но язык C я считаю одним из лучших среди всех языков программирования. Это изящный, элегантный, логичный и, что особенно важно, мощный язык. Его столь успешное развитие и распространение очень меня радует и вдохновляет.

Книга для всех программистов

Эта книга задумана как справочник для всех программистов, работающих на языке C, независимо от уровня их подготовки. Предполагается, что читатель уже имеет некоторое представление об основах языка C и может написать на нем хотя бы простейшую программу. Однако, если читатель только начал изучать C, эта книга послужит отличным дополнением к любому учебнику по C, так как в ней можно будет найти ответы на многие трудные вопросы, возникающие в процессе изучения.

Книга также будет полезна в качестве подробного справочника по основам C++, который, как известно, является объектно-ориентированным расширением языка C, т.е. она пригодится любому программисту, пишущему программы на C или C++.

Новое в четвертом издании

По сравнению с тремя предыдущими изданиями структура книги в основном осталась неизменной. Большинство изменений определено новыми возможностями языка, появившимися после введения Стандарта C99. Все эти новые возможности подробно описаны в части II книги, в предыдущих изданиях эта информация отсутствует. Часть III, в которой описывается библиотека стандартных функций, в этом издании значительно дополнена, в нее включено описание многих новых библиотечных функций, введенных Стандартом C99. Но, конечно, не изменено полное описание Стандарта C89, который, как известно, очень важен, ведь именно на его основе создан C++. Кроме того, большинство программистов работают с версией C89, потому что до настоящего времени фактически все еще нет общедоступного компилятора, поддерживающего Стандарт C99.

Книга в целом была значительно переработана с целью ознакомления с новыми характеристиками трансляторов, среды программирования и операционных систем, использующихся в настоящее время.

О чем эта книга

В книге подробно описаны все аспекты языка C и его библиотеки стандартных функций. Главный акцент сделан на Стандарте ANSI/ISO этого языка. Дано описание как Стандарта C89, так и C99.

Книга состоит из следующих шести частей:

- Основополагающие элементы языка C, определенные в C89
- Расширение C99
- Библиотеки стандартных функций C
- Распространенные алгоритмы и приложения
- Среда программирования C
- Создание интерпретатора C

В части I подробно представлены все средства языка C, т.е. его ключевые слова, инструкции препроцессора и другие. В этой части в основном описывается Стандарт C89, а также упоминаются некоторые новые свойства, введенные Стандартом C99.

В части II подробно рассматриваются новые возможности языка, введенные стандартом C99. Есть два аргумента в пользу отдельного описания стандартов C89 и C99. Во-первых, подавляющее большинство программистов используют сегодня C89. Эта версия языка C воспринимается ими как “собственно C”. К тому же это самый распространенный в мире язык программирования. Существенно также и то, что C89 является подмножеством C++. Поэтому версия C89 крайне актуальна как сегодня, так и в обозримом будущем. По этим причинам в книге должно быть сделано четкое разграничение между этими версиями языка C. Во-вторых, многие читатели этой книги уже хорошо знакомы с версией C89, и им будет значительно легче найти новый для себя материал, если новые свойства C99 будут изложены в отдельной части книги.

В части III дается описание библиотеки стандартных функций C. Рассматриваются как функции Стандарта C89, так и функции Стандарта C99, причем особо выделены функции, введенные Стандартом C99.

В части IV можно ознакомиться с наиболее важными и распространенными алгоритмами и приложениями, необходимыми для каждого программиста. Здесь рассматриваются методы искусственного интеллекта и их применение, а также программирование для Windows 2000.

В части V вы узнаете много интересного о среде программирования C, здесь обсуждаются вопросы эффективности, переносимости и отладки программ.

В части VI возможности языка C демонстрируются на примере разработки его интерпретатора. Это наиболее увлекательная и даже забавная часть книги. Поэкспериментировать с этим интерпретатором будет истинным наслаждением для любого программиста! Нам кажется, это самый лучший способ для того, чтобы по достоинству оценить чистоту и элегантность языка C.

Тексты программ в Web

Тексты программ, рассматриваемых в этой книге, можно бесплатно получить по адресу **www.williamspublishing.com**

HS
21 марта 2000 г.
Магомет (Mahomet), штат Иллинойс

Материал для дальнейшего изучения

Эта книга Герберта Шилдта — лишь одна из многих его книг по программированию. Ниже приведены другие книги, представляющие интерес для программистов.

Изучающим программирование под Windows мы рекомендуем следующие книги:
Windows 2000 Programming from the Ground Up
Windows 98 Programming from the Ground Up
Windows NT 4 Programming from the Ground Up
The Windows Programming Annotated Archives

Для изучающих язык C будут интересны такие книги:
Teach Yourself C
C/C++ Annotated Archives

Следующие книги будут полезны тем, кто изучает C++:
Полный справочник по C++ (ИД "Вильямс", 2003 г.)
Teach Yourself C++
C++ from the Ground Up
Expert C++
C/C++ Annotated Archives

Изучающим язык Java мы рекомендуем прочесть:
Java: The Complete Reference (Полный справочник по Java, изд. "Диалектика").

Если вам нужна квалифицированная консультация, обращайтесь к Герберту Шилдту, общепризнанному авторитету в области программирования

Полный справочник по

Глава 15

Математические функции

В версии C99 математическая библиотека была значительно пополнена; при этом число ее функций увеличилось более чем в три раза (стандарт C89 определял всего лишь 22 математические функции). Одной из основных целей комитета по версии C99 было повышение применимости языка C для численных расчетов. Теперь с уверенностью можно сказать, что эта цель достигнута!

Для использования математических функций в программу необходимо включить заголовок <math.h>. Помимо объявления математических функций, этот заголовок определяет один или несколько макросов. В версии C89 заголовком <math.h> определяется только макрос HUGE_VAL, который представляет собой значение типа double, сигнализирующее о возникшем переполнении. В версии C99 кроме него определены следующие макросы.

HUGE_VALF	версия макроса HUGE_VAL с типом float
HUGE_VALL	версия макроса HUGE_VAL с типом long double
INFINITY	Значение, представляющее бесконечность
math_errhandling	Содержит макросы MATH_ERRNO и/или MATH_ERREXCEPT
MATH_ERRNO	Встроенная глобальная переменная errno, используемая для вывода сообщений об ошибках
MATH_ERREXCEPT	Исключение, возбуждаемое при выполнении операций над вещественными числами, с целью вывода сообщения об ошибках
NAN	Не число

В версии C99 определены следующие макросы (подобные функциям), классифицирующие значение.

int fpclassify(*fpval*)	В зависимости от значения аргумента *fpval* возвращает FP_INFINITY, FP_NAN, FP_NORMAL, FP_SUBNORMAL или FP_ZERO. Эти макросы определяются заголовком <math.h>
int isfinite(*fpval*)	Возвращает ненулевое значение, если *fpval* конечное
int isinf(*fpval*)	Возвращает ненулевое значение, если *fpval* представляет бесконечность
int isnan(*fpval*)	Возвращает ненулевое значение, если *fpval* — не является числом
int isnormal(*fpval*)	Возвращает ненулевое значение, если *fpval* представляет собой нормализованное число
int signbit(*fpval*)	Возвращает ненулевое значение, если *fpval* отрицательно (т.е. установлен его знаковый разряд)

В версии C99 определены следующие макросы сравнения, аргументы которых (*a* и *b*) должны иметь числовые значения в формате с плавающей точкой.

int isgreater(*a, b*)	Возвращает ненулевое значение, если *a* больше *b*
int isgreaterequal(*a, b*)	Возвращает ненулевое значение, если *a* больше или равно *b*
int isless(*a, b*)	Возвращает ненулевое значение, если *a* меньше *b*
int islessequal(*a, b*)	Возвращает ненулевое значение, если *a* меньше или равно *b*
int islessgreater(*a, b*)	Возвращает ненулевое значение, если *a* больше или меньше *b*
int isunordered(*a, b*)	Возвращает 1, если *a* и *b* не упорядочены одно по отношению к другому Возвращает 0, если *a* и *b* упорядочены

Эти макросы введены, так как они прекрасно обрабатывают значения, которые не являются числами, не вызывая при этом исключений вещественного типа.

Макросы EDOM и ERANGE также используются математическими функциями. Эти макросы определены в заголовке <errno.h>.

Ошибки в версиях C89 и C99 обрабатываются по-разному. Так, в версии C89, если аргумент математической функции не попадает в область определения, возвращается

некоторое значение, зависящее от конкретной реализации, а встроенная глобальная целая переменная errno устанавливается равной значению EDOM. В версии C99 нарушение области определения также приводит к возврату значения, зависящего от конкретной реализации. Однако по значению math_errhandling можно судить о выполнении других действий. Если math_errhandling содержит значение MATH_ERRNO, то встроенная глобальная целая переменная errno устанавливается равной значению EDOM. Если же math_errhandling содержит значение MATH_ERREXCEPT, возбуждается исключение вещественного типа.

В версии C89, если функция генерирует результат, который слишком велик и потому не может быть представлен в машинном формате, то происходит переполнение. В этом случае функция возвращает значение HUGE_VAL, а переменная errno устанавливается равной значению ERANGE, сигнализирующему о выходе за пределы диапазона. При потере значимости функция возвращает нуль и устанавливает переменную errno равной значению ERANGE. В версии C99 ошибка переполнения также приводит к тому, что функция возвращает значение HUGE_VAL, а при потере значимости — нуль. Если math_errhandling содержит значение MATH_ERRNO, глобальная переменная errno устанавливается равной значению ERANGE, свидетельствующему об ошибке диапазона. Если же math_errhandling содержит значение MATH_ERREXCEPT, возбуждается исключение вещественного типа.

В версии C89 аргументами математических функций должны быть значения типа double, причем значения, возвращаемые функциями, тоже имеют тип double. В версии C99 добавлены варианты этих функций, работающие с типами float и long double. В этих функциях используются суффиксы f и l соответственно. Например, в версии C89 функция sin() определена следующим образом.

```
double sin(double arg);
```

Версия C99 поддерживает приведенное выше определение функции sin(), но в ней добавлены еще две ее модификации — sinf() и sinl().

```
float sinf(float arg);
long double sinl(long double arg);
```

Операции, выполняемые всеми тремя функциями, одинаковы; различаются лишь типы данных, над которыми выполняются эти операции. Добавление модификаций f и l математических функций позволяет использовать версию, которая наиболее точно соответствует данным, с которыми работают функции.

Поскольку в версии C99 добавлено так много новых функций, стоит отдельно перечислить те из них, которые поддерживаются версией C89. Это следующие функции.

acos	cos	fmod	modf	tan
asin	cosh	frexp	pow	tanh
atan	exp	ldexp	sin	
atan2	fabs	log	sinh	
ceil	floor	log10	sqrt	

И последнее замечание: все углы задаются в радианах.

Семейство функций acos

```
#include <math.h>
float acosf(float arg);
double acos(double arg);
long double acosl(long double arg);
```

Функции `acosf()` и `acosl()` добавлены в версии C99.

Каждая функция семейства `acos()` возвращает значение арккосинуса от аргумента *arg*. Значение аргумента должно находиться в диапазоне от -1 до 1; в противном случае возникает ошибка из-за выхода за пределы области допустимых значений (ошибка из-за нарушения области определения).

Пример

Данная программа выводит значения арккосинусов последовательности аргументов, лежащих в интервале от -1 до 1 и увеличивающихся с шагом одна десятая, т.е. программа составляет таблицу арккосинуса.

```
#include <math.h>
#include <stdio.h>

int main(void)
{
  double val = -1.0;

  do {
    printf("Арккосинус %f равен %f.\n", val, acos(val));
    val += 0.1;
  } while(val<=1.0);

  return 0;
}
```

Зависимые функции

`asin()`, `atan()`, `atan2()`, `sin()`, `cos()`, `tan()`, `sinh()`, `cosh()` и `tanh()`

Семейство функций acosh

```
#include <math.h>
float acoshf(float arg);
double acosh(double arg);
long double acoshl(long double arg);
```

Функции `acosh()`, `acoshf()` и `acoshl()` добавлены в версии C99.

Каждая функция семейства `acosh()` возвращает значение гиперболического арккосинуса от аргумента *arg*. Значение аргумента должно быть больше или равно нулю; в противном случае возникает ошибка из-за выхода за пределы области допустимых значений (ошибка из-за нарушения области определения).

Зависимые функции

`asinh()`, `atanh()`, `sinh()`, `cosh()` и `tanh()`

Семейство функций asin

```
#include <math.h>
float asinf(float arg);
double asin(double arg);
long double asinl(long double arg);
```

Функции `asinf()` и `asinl()` добавлены в версии C99.

Каждая функция семейства `asin()` возвращает значение арксинуса от аргумента *arg*. Значение аргумента должно находиться в диапазоне от -1 до 1; в противном случае возникает ошибка из-за выхода за пределы области допустимых значений (ошибка из-за нарушения области определения).

Пример

Данная программа выводит значения арксинусов последовательности аргументов, лежащих в интервале от -1 до 1 и увеличивающихся с шагом одна десятая, т.е. составляет таблицу арксинуса.

```
#include <math.h>
#include <stdio.h>

int main(void)
{
  double val = -1.0;

  do {
    printf("Арксинус %f равен %f.\n", val, asin(val));
    val += 0.1;
  } while(val<=1.0);

  return 0;
}
```

Зависимые функции

`acos()`, `atan()`, `atan2()`, `sin()`, `cos()`, `tan()`, `sinh()`, `cosh()` и `tanh()`

Семейство функций asinh

```
#include <math.h>
float asinhf(float arg);
double asinh(double arg);
long double asinhl(long double arg);
```

Функции `asinh()`, `asinhf()` и `asinhl()` добавлены в версии C99.

Каждая функция семейства `asinh()` возвращает значение гиперболического арксинуса от аргумента *arg*.

Зависимые функции

`acosh()`, `atanh()`, `sinh()`, `cosh()` и `tanh()`

Семейство функций atan

```
#include <math.h>
float atanf(float arg);
double atan(double arg);
long double atanl(long double arg);
```

Функции `atanf()` и `atanl()` добавлены в версии C99.

Каждая функция семейства `atan()` возвращает значение арктангенса от аргумента *arg*.

Пример

Данная программа выводит значения арктангенсов последовательности аргументов, лежащих в интервале от -1 до 1 и увеличивающихся с шагом одна десятая, т.е. составляет таблицу арктангенса.

```
#include <math.h>
#include <stdio.h>

int main(void)
{
  double val = -1.0;

  do {
    printf("Арктангенс %f равен %f.\n", val, atan(val));
    val += 0.1;
  } while(val<=1.0);

  return 0;
}
```

Зависимые функции

`asin()`, `acos()`, `atan2()`, `tan()`, `cos()`, `sin()`, `sinh()`, `cosh()` и `tanh()`

Семейство функций atanh

```
#include <math.h>
float atanhf(float arg);
double atanh(double arg);
long double atanhl(long double arg);
```

Функции `atanh()`, `atanhf()` и `atanhl()` добавлены в версии C99.

Каждая функция семейства `atanh()` возвращает значение гиперболического арктангенса от аргумента *arg*. Значение аргумента должно находиться в диапазоне от -1 до 1 (не включая границы); в противном случае возникает ошибка из-за выхода за пределы области допустимых значений (ошибка из-за нарушения области определения). Если *arg* равен 1 или -1, возможен выход за пределы допустимого диапазона.

Зависимые функции

`acosh()`, `asinh()`, `sinh()`, `cosh()` и `tanh()`

Семейство функций atan2

```
#include <math.h>
float atan2f(float a, float b);
double atan2(double a, double b);
long double atan2l(long double a, long double b);
```

Функции `atan2f()` и `atan2l()` добавлены в версии C99.

Каждая функция семейства `atan2()` возвращает значение арктангенса отношения *a/b*. Для вычисления квадранта возвращаемого значения используются знаки аргументов функции.

Пример

Данная программа выводит значения арктангенсов последовательности аргументов *y*, лежащих в интервале от -1 до 1 и увеличивающихся с шагом одна десятая, т.е. составляет таблицу арктангенса.

```
#include <math.h>
#include <stdio.h>

int main(void)
{
  double val = -1.0;

  do {
    printf("Арктангенс %f равен %f.\n", val, atan2(val, 1.0));
    val += 0.1;
  } while(val<=1.0);

  return 0;
}
```

Зависимые функции

`asin()`, `acos()`, `atan()`, `tan()`, `cos()`, `sin()`, `sinh()`, `cosh()` и `tanh()`

Семейство функций cbrt

```
#include <math.h>
float cbrtf(float num);
double cbrt(double num);
long double cbrtl(long double num);
```

Функции `cbrt()`, `cbrtf()` и `cbrtl()` добавлены в версии C99.

Каждая функция семейства `cbrt()` возвращает значение кубического корня от аргумента *num*.

Пример

Данный фрагмент программы выводит на экран число `2`.

```
printf("%f", cbrt(8));
```

Зависимые функции

`sqrt()`

Семейство функций ceil

```
#include <math.h>
float ceilf(float num);
```

```
double ceil(double num);
long double ceill(long double num);
```

Функции `ceilf()` и `ceill()` добавлены в версии C99.

Каждая функция семейства `ceil()` возвращает наименьшее целое (представленное в виде значения с плавающей точкой), которое больше значения аргумента *num* или равно ему. Например, если *num* равно 1.02, функция `ceil()` вернет значение 2.0, а при *num*, равном -1.02, — значение -1.

Пример

Данный фрагмент программы выводит на экран число `10`.

```
printf("%f", ceil(9.9));
```

Зависимые функции

`floor()` и `fmod()`

Семейство функций copysign

```
#include <math.h>
float copysignf(float val, float signval);
double copysign(double val, double signval);
long double copysignl(long double val, long double signval);
```

Функции `copysign()`, `copysignf()` и `copysignl()` добавлены в версии C99.

Каждая функция семейства `copysign()` наделяет аргумент *val* знаком, который имеет аргумент *signval*, и возвращает полученный результат. Таким образом, возвращаемое значение имеет величину, равную величине аргумента *val*, а его знак совпадает со знаком аргумента *signval*.

Зависимые функции

`fabs()`

Семейство функций cos

```
#include <math.h>
float cosf(float arg);
double cos(double arg);
long double cosl(long double arg);
```

Функции `cosf()` и `cosl()` добавлены в версии C99.

Каждая функция семейства `cos()` возвращает значение косинуса аргумента *arg*. Значение аргумента должно быть выражено в радианах.

Пример

Данная программа выводит значения косинусов последовательности аргументов, лежащих в интервале от −1 до 1 и увеличивающихся с шагом одна десятая, т.е. составляет таблицу косинуса.

```
#include <math.h>
#include <stdio.h>

int main(void)
{
  double val = -1.0;

  do {
    printf("Косинус %f равен %f.\n", val, cos(val));
    val += 0.1;
  } while(val<=1.0);

  return 0;
}
```

Зависимые функции

asin(), acos(), atan2(), atan(), tan(), sin(), sinh(), cos() и tanh()

Семейство функций cosh

```
#include <math.h>
float coshf(float arg);
double cosh(double arg);
long double coshl(long double arg);
```

Функции coshf() и coshl() добавлены в версии C99.

Каждая функция семейства cosh() возвращает значение гиперболического косинуса аргумента *arg*.

Пример

Данная программа выводит значения гиперболических косинусов последовательности аргументов, лежащих в интервале от −1 до 1 и увеличивающихся с шагом одна десятая, т.е. составляет таблицу гиперболического косинуса.

```
#include <math.h>
#include <stdio.h>

int main(void)
{
  double val = -1.0;

  do {
    printf("Гиперболический косинус %f равен %f.\n", val, cosh(val));
    val += 0.1;
  } while(val<=1.0);

  return 0;
}
```

Зависимые функции

asin(), acos(), atan2(), atan(), tan(), sin() и tanh()

Семейство функций erf

```
#include <math.h>
float erff(float arg);
double erf(double arg);
long double erfl(long double arg);
```

Функции `erf()`, `erff()` и `erfl()` добавлены в версии C99.

Каждая функция семейства `erf()` возвращает значение функции ошибок[1] от аргумента *arg*.

Зависимые функции

`erfc()`

Семейство функций erfc

```
#include <math.h>
float erfcf(float arg);
double erfc(double arg);
long double erfcl(long double arg);
```

Функции `erfc()`, `erfcf()` и `erfcl()` добавлены в версии C99.

Каждая функция семейства `erfc()` возвращает функцию ошибок дополнительную[2] от аргумента *arg*.

Зависимые функции

`erf()`

Семейство функций exp

```
#include <math.h>
float expf(float arg);
double exp(double arg);
long double expl(long double arg);
```

Функции `expf()` и `expl()` добавлены в версии C99.

---

[1] Интеграл (вероятности) ошибок:

$$erf(x)=\frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt\ .$$

Иногда называется просто *интегралом ошибок* или *интегралом вероятности*. В теории вероятности чаще используется не интеграл вероятности, а *интеграл вероятности Гаусса*, или *функция нормального распределения*

$$\Phi(x)=\frac{1}{2}[1+erf(\frac{x}{\sqrt{2}})]\ .$$ — *Прим. ред.*

[2] *Дополнительный интеграл вероятности*:

$$\mathrm{erfc}(x)=\frac{2}{\sqrt{\pi}}\int_x^\infty e^{-t^2}dt=1\text{-}\mathrm{erf}(x).$$ — *Прим. ред.*

Каждая функция семейства `exp()` возвращает значение экспоненты от аргумента *arg* (число *e*, возведенное в степень, которая равна значению аргумента *arg*).

Пример

Данный фрагмент программы выводит число e, округленное до значения 2.718282.

```
printf("e, возведенное в первую степень, приблизительно равно: %f.",
exp(1.0));
```

Зависимые функции

`exp2()` и `log()`

Семейство функций exp2

```
#include <math.h>
float exp2f(float arg);
double exp2(double arg);
long double exp2l(long double arg);
```

Функции `exp2()`, `exp2f()` и `exp2l()` добавлены в версии C99.

Каждая функция семейства `exp2()` возвращает число 2, возведенное в степень *arg*.

Зависимые функции

`exp()` и `log()`

Семейство функций expm1

```
#include <math.h>
float expm1f(float arg);
double expm1(double arg);
long double expm1l(long double arg);
```

Функции `expm1()`, `expm1f()` и `expm1l()` добавлены в версии C99.

Каждая функция семейства `expm1()` возвращает уменьшенное на единицу значение числа *e*, возведенного в степень *arg* (т.е. возвращаемое значение равно $e^{arg}-1$).

Зависимые функции

`exp()` и `log()`

Семейство функций fabs

```
#include <math.h>
float fabsf(float num);
double fabs(double num);
long double fabsl(long double num);
```

Функции `fabsf()` и `fabsl()` добавлены в версии C99.

Каждая функция семейства `fabs()` возвращает абсолютное значение аргумента *num*.

Пример

Данная программа дважды выводит на экран число `1.0`.

```
#include <math.h>
#include <stdio.h>

int main(void)
{
  printf("%1.1f %1.1f", fabs(1.0), fabs(-1.0));

  return 0;
}
```

Зависимые функции

`abs()`

Семейство функций fdim

```
#include <math.h>
float fdimf(float a, float b);
double fdim(double a, double b);
long double fdiml(long double a, long double b);
```

Функции `fdim()`, `fdimf()` и `fdiml()` добавлены в версии C99.

Каждая функция семейства `fdim()` возвращает нуль, если значение аргумента *a* меньше значения аргумента *b* или равно ему. В противном случае возвращается результат вычисления разности *a-b*.

Зависимые функции

`remainder()` и `remquo()`

Семейство функций floor

```
#include <math.h>
float floorf(float num);
double floor(double num);
long double floorl(long double num);
```

Функции `floorf()` и `floorl()` добавлены в версии C99.

Каждая функция семейства `floor()` возвращает наибольшее целое (представленное в виде значения с плавающей точкой), которое меньше значения аргумента *num* или равно ему. Например, при *num*, равном 1.02, функция `floor()` вернет значение 1.0, а при *num*, равном −1.02, — значение −2.0.

Пример

Данный фрагмент программы выводит на экран число `10`.

```
printf("%f", floor(10.9));
```

Зависимые функции

`ceil()` и `fmod()`

Семейство функций fma

```
#include <math.h>
float fmaf(float a, float b, float c);
double fma(double a, double b, double c);
long double fmal(long double a, long double b, long double c);
```

Функции `fma()`, `fmaf()` и `fmal()` определены в версии C99.

Каждая функция семейства `fma()` возвращает значение выражения *a*b+c*. Округление выполняется только один раз, после завершения всей операции.

Зависимые функции

`round()`, `lround()` и `llround()`

Семейство функций fmax

```
#include <math.h>
float fmaxf(float a, float b);
double fmax(double a, double b);
long double fmaxl(long double a, long double b);
```

Функции `fmax()`, `fmaxf()` и `fmaxl()` определены в версии C99.

Каждая функция семейства `fmax()` возвращает больший из аргументов *a* и *b*.

Зависимые функции

`fmin()`

Семейство функций fmin

```
#include <math.h>
float fminf(float a, float b);
double fmin(double a, double b);
long double fminl(long double a, long double b);
```

Функции `fmin()`, `fminf()` и `fminl()` определены в версии C99.

Каждая функция семейства `fmin()` возвращает меньший из аргументов *a* и *b*.

Зависимые функции

`fmax()`

Семейство функций fmod

```
#include <math.h>
float fmodf(float a, float b);
double fmod(double a, double b);
long double fmodl(long double a, long double b);
```

Функции `fmodf()` и `fmodl()` определены в версии C99.

Каждая функция семейства `fmod()` возвращает остаток от деления аргументов *a/b*.

Пример

Данная программа выводит на экран число `1.0`, являющееся остатком деления 10/3.

```
#include <math.h>
#include <stdio.h>

int main(void)
{
  printf("%1.1f", fmod(10.0,3.0));

  return 0;
}
```

Зависимые функции

`ceil()`, `floor()` и `fabs()`

Семейство функций frexp

```
#include <math.h>
float frexpf(float num, int *exp);
double frexp(double num, int *exp);
long double frexpl(long double num, int *exp);
```

Функции `frexpf()` и `frexpl()` добавлены в версии C99.

Каждая функция семейства `frexp()` разбивает число *num* на мантиссу *mantissa*, значение которой удовлетворяет неравенствам $0.5 \le mantissa < 1$, и целый показатель степени числа 2 (он обозначен через *exp*), притом числа *mantissa* и *exp* выбираются так, чтобы выполнялось равенство $num = mantissa * 2^{exp}$. Значение мантиссы возвращается функцией, а значение показателя[1] присваивается переменной, адресуемой указателем *exp*.

[1] Напомним, что представление числа *num* в виде $num = mantissa * b^{exp}$ (здесь *b* — основание системы счисления) называется *представлением с плавающей точкой* (*запятой*) или *полулогарифмическим представлением*, и что целая часть логарифма называется характеристикой. Так что $exp = \chi_2(num)+1$, где $\chi_2(num) = \lfloor \log_2(num) \rfloor$ — характеристика двоичного логарифма. Число *exp* часто называется *порядком* числа *num* (в нормализованном представлении). Заметим также, что терминам *мантисса* и *характеристика* часто придается и иной смысл. Так, по историческим причинам под *мантиссой* часто подразумевают *дробную часть логарифма*; иногда ее называют также *мантиссой логарифма*. Что же касается *характеристики*, то под ней иногда понимают просто *число, которое представляет порядок в представлении с плавающей запятой*. (В этом смысле в большинстве машин характеристика равна порядку, если он положительный; отличия между ними, как правило, обусловлены тем, что представление порядка, который может быть также и неположительным числом, при реализации операций над числами в полулогарифмическом представлении рассматривают как представление неотрицательного числа.) Так что можно сказать, что характеристика в этом смысле — машинное представление порядка числа. *Порядок* в этом контексте называется также иногда *экспонентой*. (Не путайте с экспонентой-функцией!) — *Прим. ред.*

Пример

Данный фрагмент программы выводит число `0.625` в качестве мантиссы и число `4` — в качестве показателя степени.

```
int e;
double f;

f = frexp(10.0, &e);
printf("%f %d", f, e);
```

Зависимые функции

`ldexp()`

Семейство функций hypot

```
#include <math.h>
float hypotf(float side1, float side2);
double hypot(double side1, double side2);
long double hypotl(long double side1, long double side2);
```

Функции `hypot()`, `hypotf()` и `hypotl()` определены в версии C99.

Каждая функция семейства `hypot()` возвращает длину гипотенузы при заданных длинах двух катетов (т.е. функция возвращает значение квадратного корня из суммы квадратов значений аргументов *side1* и *side2*)[1].

Зависимые функции

`sqrt()`

Семейство функций ilogb

```
#include <math.h>
int ilogbf(float num);
int ilogb(double num);
int ilogbl(long double num);
```

Функции `ilogb()`, `ilogbf()` и `ilogbl()` добавлены в версии C99.

Каждая функция семейства `ilogb()` возвращает порядок аргумента *num*. Возвращаемое значение имеет тип `int`.

Зависимые функции

`logb()`

Семейство функций ldexp

```
#include <math.h>
float ldexpf(float num, int exp);
```

[1] Или расстояние точки с координатами (*side1; side2*) от начала координат. — *Прим. ред.*

```
double ldexp(double num, int exp);
long double ldexpl(long double num, int exp);
```

Функции `ldexpf()` и `ldexpl()` добавлены в версии C99.

Каждая функция семейства `ldexp()` возвращает значение выражения *num* * 2^{exp}.

Пример

Данная программа выводит число `4`.

```
#include <math.h>
#include <stdio.h>

int main(void)
{
  printf("%f", ldexp(1, 2));

  return 0;
}
```

Зависимые функции

`frexp()` и `modf()`

Семейство функций lgamma

```
#include <math.h>
float lgammaf(float arg);
double lgamma(double arg);
long double lgammal(long double arg);
```

Функции `lgamma()`, `lgammaf()` и `lgammal()` добавлены в версии C99.

Каждая функция семейства `lgamma()` вычисляет абсолютное значение *гамма-функции*[1] от аргумента *arg* и возвращает ее натуральный логарифм.

Зависимые функции

`tgamma()`

Семейство функций llrint

```
#include <math.h>
long long int llrintf(float arg);
long long int llrint(double arg);
long long int llrintl(long double arg);
```

Функции `llrint()`, `llrintf()` и `llrintl()` добавлены в версии C99.

Каждая функция семейства `llrint()` возвращает значение аргумента *arg*, округленного до ближайшего целого, которое имеет тип `long long int`.

Зависимые функции

`lrint()` и `rint()`

[1] Другие названия: *Г-функция*, *Г-функция Эйлера*, *эйлеров интеграл второго рода*. — *Прим. ред.*

Семейство функций llround

```
#include <math.h>
long long int llroundf(float arg);
long long int llround(double arg);
long long int llroundl(long double arg);
```

Функции `llround()`, `llroundf()` и `llroundl()` добавлены в версии C99.

Каждая функция семейства `llround()` возвращает значение аргумента *arg*, округленное до ближайшего целого, которое имеет тип `long long int`. Значения, отстоящие от большего и меньшего целых на одинаковую величину (например, число 3.5), округляются в сторону большего целого.

Зависимые функции

`lround()` и `round()`

Семейство функций log

```
#include <math.h>
float logf(float num);
double log(double num);
long double logl(long double num);
```

Функции `logf()` и `logl()` добавлены в версии C99.

Каждая функция семейства `log()` возвращает значение натурального логарифма от аргумента *num*. Если значение аргумента *num* отрицательно, возникает ошибка из-за выхода за пределы области допустимых значений (ошибка из-за нарушения области определения). Если же значение *num* равно нулю, возможна ошибка из-за выхода за пределы диапазона представимых значений.

Пример

Следующая программа выводит на экран значения натуральных логарифмов чисел от 1 до 10 (с шагом 1), т.е. составляет таблицу натуральных логарифмов целых чисел от 1 до 10.

```
#include <math.h>
#include <stdio.h>

int main(void)
{
  double val = 1.0;

  do {
    printf("%f %f\n", val, log(val));
    val++;
  } while (val<11.0);

  return 0;
}
```

Зависимые функции

`log10()` и `log2()`

Семейство функций log1p

```
#include <math.h>
float log1pf(float num);
double log1p(double num);
long double log1pl(long double num);
```

Функции `log1p()`, `log1pf()` и `log1pl()` добавлены в версии C99.

Каждая функция семейства `log1p()` возвращает значение натурального логарифма от аргумента *num+1*. Если значение аргумента *num* отрицательно, возникает ошибка из-за выхода за пределы области допустимых значений (ошибка из-за нарушения области определения). Если же значение *num* равно −1, возможна ошибка из-за выхода за пределы диапазона представимых значений.

Зависимые функции

`log()`

Семейство функций log10

```
#include <math.h>
float log10f(float num);
double log10(double num);
long double log10l(long double num);
```

Функции `log10f()` и `log10l()` добавлены в версии C99.

Каждая функция семейства `log10()` возвращает значение логарифма по основанию 10 от аргумента *num*. Если значение аргумента *num* отрицательно, возникает ошибка из-за выхода за пределы области допустимых значений (ошибка из-за нарушения области определения). Если же значение *num* равно нулю, возможна ошибка из-за выхода за пределы диапазона представимых значений.

Пример

Данная программа выводит значение десятичных логарифмов чисел, изменяющихся от 1 до 10 с шагом 1, т.е. составляет таблицу десятичных логарифмов целых чисел от 1 до 10.

```
#include <math.h>
#include <stdio.h>

int main(void)
{
  double val = 1.0;

  do {
    printf("%f %f\n", val, log10(val));
    val++;
  } while (val<11.0);

  return 0;
}
```

Зависимые функции

`log()` и `log2()`

Семейство функций log2

```
#include <math.h>
float log2f(float num);
double log2(double num);
long double log2l(long double num);
```

Функции `log2()`, `log2f()` и `log2l()` добавлены в версии C99.

Каждая функция семейства `log2()` возвращает значение логарифма по основанию 2 от аргумента *num*. Если значение аргумента *num* отрицательно, возникает ошибка из-за выхода за пределы области допустимых значений (ошибка из-за нарушения области определения). Если же значение *num* равно нулю, возможна ошибка из-за выхода за пределы диапазона представимых значений[1].

Зависимые функции

`log()` и `log10()`

Семейство функций logb

```
#include <math.h>
float logbf(float num);
double logb(double num);
long double logbl(long double num);
```

Функции `logb()`, `logbf()` и `logbl()` добавлены в версии C99.

Каждая функция семейства `logb()` возвращает показатель аргумента *num*. Возвращаемое значение является числом с плавающей точкой. Если значение аргумента *num* равно нулю, возможна ошибка из-за выхода за пределы диапазона представимых значений.

Зависимые функции

`ilogb()`

Семейство функций lrint

```
#include <math.h>
long int lrintf(float arg);
long int lrint(double arg);
long int lrintl(long double arg);
```

Функции `lrint()`, `lrintf()` и `lrintl()` добавлены в версии C99.

Каждая функция семейства `lrint()` возвращает значение аргумента *arg*, округленное до ближайшего целого, которое имеет тип `long int`.

Зависимые функции

`llrint()` и `rint()`

[1] Как известно, в нуле логарифм не определен, но из-за трудностей представления близких к нулю положительных чисел автор придерживается столь осторожных формулировок. — *Прим. ред.*

Семейство функций lround

```
#include <math.h>
long int lroundf(float arg);
long int lround(double arg);
long int lroundl(long double arg);
```

Функции `lround()`, `lroundf()` и `lroundl()` добавлены в версии C99.

Каждая функция семейства `lround()` возвращает значение аргумента *arg*, округленное до ближайшего целого, которое имеет тип `long int`. Значения, отстоящие от большего и меньшего целых на одинаковую величину (например, число 3.5), округляются в сторону большего целого.

Зависимые функции

`llround()` и `round()`

Семейство функций modf

```
#include <math.h>
float modff(float num, float *i);
double modf(double num, double *i);
long double modfl(long double num, long double *i);
```

Функции `modff()` и `modfl()` добавлены в версии C99.

Каждая функция семейства `modf()` разбивает аргумент *num* на целую и дробную части. Функция возвращает дробную часть и размещает целую часть в переменной, адресуемой параметром *i*.

Пример

Данный фрагмент программы выводит на экран числа `10` и `0.123`.

```
double i;
double f;

f = modf(10.123, &i);
printf("%f %f", i, f);
```

Зависимые функции

`frexp()` и `ldexp()`

Семейство функций nan

```
#include <math.h>
float nanf(const char *content);
double nan(const char *content);
long double nanl(const char *content);
```

Функции `nan()`, `nanf()` и `nanl()` добавлены в версии C99.

Каждая функция семейства `nan()` возвращает значение, которое не является числом и которое содержит строку, адресуемую параметром *content*.

Зависимые функции

`isnan()`

Семейство функций nearbyint

```
#include <math.h>
float nearbyintf(float arg);
double nearbyint(double arg);
long double nearbyintl(long double arg);
```

Функции `nearbyint()`, `nearbyintf()` и `nearbyintl()` добавлены в версии C99.

Каждая функция семейства `nearbyint()` возвращает значение аргумента *arg*, округленное до ближайшего целого. Однако возвращаемое число представлено в формате с плавающей точкой.

Зависимые функции

`rint()` и `round()`

Семейство функций nextafter

```
#include <math.h>
float nextafterf(float from, float towards);
double nextafter(double from, double towards);
long double nextafterl(long double from, long double towards);
```

Функции `nextafter()`, `nextafterf()` и `nextafterl()` добавлены в версии C99.

Каждая функция семейства `nextafter()` возвращает значение, следующее после аргумента *from*, причем выбор следующего значения осуществляется в направлении, задаваемом аргументом *towards*.

Зависимые функции

`nexttoward()`

Семейство функций nexttoward

```
#include <math.h>
float nexttowardf(float from, long double towards);
double nexttoward(double from, long double towards);
long double nexttowardl(long double from, long double towards);
```

Функции `nexttoward()`, `nexttowardf()` и `nexttowardl()` добавлены в версии C99.

Каждая функция семейства `nexttoward()` возвращает значение, следующее после аргумента *from*, причем выбор следующего значения осуществляется в направлении, задаваемом аргументом *towards*. Действие этих функций аналогично действию функций семейства `nextafter()` за исключением того, что параметр всех трех функций *towards* имеет тип `long double`.

Зависимые функции

`nextafter()`

Семейство функций pow

```
#include <math.h>
float powf(float base, float exp);
double pow(double base, double exp);
long double powl(long double base, long double exp);
```

Функции `powf()` и `powl()` добавлены в версии C99.

Каждая функция семейства `pow()` возвращает значение аргумента *base*, возведенное в степень *exp*, т.е. в результате получается $base^{exp}$. Если значение аргумента *base* равно нулю, а *exp* меньше или равно нулю, возможна ошибка из-за выхода за пределы области допустимых значений (ошибка из-за нарушения области определения). Она произойдет также в том случае, если *base* отрицательно, а *exp* не является целым числом. При этом также может возникнуть ошибка из-за выхода за пределы диапазона представимых значений.

Пример

Следующая программа выводит первые десять степеней числа 10, т.е. составляет таблицу степеней числа 10.

```
#include <math.h>
#include <stdio.h>

int main(void)
{
  double x = 10.0, y = 0.0;

  do {
    printf("%f\n", pow(x, y));
    y++;
  } while(y<11.0);

  return 0;
}
```

Зависимые функции

`exp()`, `log()` и `sqrt()`

Семейство функций remainder

```
#include <math.h>
float remainderf(float a, float b);
double remainder(double a, double b);
long double remainderl(long double a, long double b);
```

Функции `remainder()`, `remainderf()` и `remainderl()` определены в версии C99.

Каждая функция семейства `remainder()` возвращает остаток от деления значений аргументов *a/b*.

Зависимые функции

`remquo()`

Семейство функций remquo

```
#include <math.h>
float remquof(float a, float b, int *quo);
double remquo(double a, double b, int *quo);
long double remquol(long double a, long double b, int *quo);
```

Функции `remquo()`, `remquof()` и `remquol()` определены в версии C99.

Каждая функция семейства `remquo()` возвращает остаток от деления значений аргументов *a/b*. При этом целое, адресуемое параметром *quo*, будет содержать частное.

Зависимые функции

`remainder()`

Семейство функций rint

```
#include <math.h>
float rintf(float arg);
double rint(double arg);
long double rintl(long double arg);
```

Функции `rint()`, `rintf()` и `rintl()` добавлены в версии C99.

Каждая функция семейства `rint()` возвращает значение аргумента *arg*, округленное до ближайшего целого. Однако возвращаемое число представлено в формате с плавающей точкой. Может возникнуть исключение вещественного типа.

Зависимые функции

`nearbyint()` и `round()`

Семейство функций round

```
#include <math.h>
float roundf(float arg);
double round(double arg);
long double roundl(long double arg);
```

Функции `round()`, `roundf()` и `roundl()` добавлены в версии C99.

Каждая функция семейства `round()` возвращает значение аргумента *arg*, округленное до ближайшего целого. Однако возвращаемое число представлено в формате с плавающей точкой. Значения, отстоящие от большего и меньшего целого на одинаковую величину (например, число 3.5), округляются в сторону большего целого.

Зависимые функции

`lround()` и `llround()`

Семейство функций scalbln

```
#include <math.h>
float scalblnf(float val, long int exp);
double scalbln(double val, long int exp);
long double scalblnl(long double val, long int exp);
```

Функции `scalbln()`, `scalblnf()` и `scalblnl()` добавлены в версии C99.

Каждая функция семейства `scalbln()` возвращает произведение параметра *val* и значения `FLT_RADIX`, возведенного в степень, которая равна значению параметра *exp*, т.е. в результате получается *val* * FLT_RADIX^{exp}.

Макрос `FLT_RADIX` определен в заголовке `<float.h>`, и его значение равно основанию системы счисления, используемой для представления вещественных чисел.

Зависимые функции

`scalbn()`

Семейство функций scalbn

```
#include <math.h>
float scalbnf(float val, int exp);
double scalbn(double val, int exp);
long double scalbnl(long double val, int exp);
```

Функции `scalbn()`, `scalbnf()` и `scalbnl()` добавлены в версии C99.

Каждая функция семейства `scalbn()` возвращает произведение параметра *val* и значения `FLT_RADIX`, возведенного в степень *exp*, т.е. в результате получается *val* * FLT_RADIX^{exp}.

Макрос `FLT_RADIX` определен в заголовке `<float.h>`, и его значение равно основанию системы счисления, используемой для представления вещественных чисел.

Зависимые функции

`scalbln()`

Семейство функций sin

```
#include <math.h>
float sinf(float arg);
double sin(double arg);
long double sinl(long double arg);
```

Функции `sinf()` и `sinl()` добавлены в версии C99.

Каждая функция семейства `sin()` возвращает значение синуса от аргумента *arg*. Значение аргумента должно быть задано в радианах.

Пример

Данная программа выводит синусы последовательности значений, лежащих в пределах от −1 до 1 и изменяющихся с шагом одна десятая, т.е. составляет таблицу синусов чисел от −1 до 1.

```
#include <math.h>
#include <stdio.h>

int main(void)
{
  double val = -1.0;

  do {
    printf("Синус %f равен %f.\n", val, sin(val));
    val += 0.1;
  } while(val<=1.0);

  return 0;
}
```

Зависимые функции

`asin()`, `acos()`, `atan2()`, `atan()`, `tan()`, `cos()`, `sinh()`, `cosh()` и `tanh()`

Семейство функций sinh

```
#include <math.h>
float sinhf(float arg);
double sinh(double arg);
long double sinhl(long double arg);
```

Функции `sinhf()` и `sinhl()` добавлены в версии C99.

Каждая функция семейства `sinh()` возвращает значение гиперболического синуса от аргумента *arg*.

Пример

Данная программа выводит гиперболические синусы последовательности значений, лежащих в пределах от -1 до 1 и изменяющихся с шагом одна десятая, т.е. составляет таблицу гиперболических синусов чисел от -1 до 1.

```
#include <math.h>
#include <stdio.h>

int main(void)
{
  double val = -1.0;

  do {
    printf("Гиперболический синус %f равен %f.\n", val, sinh(val));
    val += 0.1;
  } while(val<=1.0);

  return 0;
}
```

Зависимые функции

`asin()`, `acos()`, `atan2()`, `atan()`, `tan()`, `cos()`, `cosh()` и `sin()`

Семейство функций sqrt

```
#include <math.h>
float sqrtf(float num);
double sqrt(double num);
long double sqrtl(long double num);
```

Функции `sqrtf()` и `sqrtl()` добавлены в версии C99.

Каждая функция семейства `sqrt()` возвращает значение квадратного корня от аргумента *num*. Если значение аргумента отрицательно, возникает ошибка из-за выхода за пределы области допустимых значений (ошибка из-за нарушения области определения).

Пример

Данный фрагмент программы выводит на экран число `4`.

```
printf("%f", sqrt(16.0));
```

Зависимые функции

`exp()`, `log()` и `pow()`

Семейство функций tan

```
#include <math.h>
float tanf(float arg);
double tan(double arg);
long double tanl(long double arg);
```

Функции `tanf()` и `tanl()` добавлены в версии C99.

Каждая функция семейства `tan()` возвращает значение тангенса от аргумента *arg*. Значение аргумента должно быть задано в радианах.

Пример

Данная программа выводит тангенсы последовательности значений, лежащих в пределах от -1 до 1 и изменяющихся с шагом одна десятая, т.е. составляет таблицу тангенсов чисел от -1 до 1.

```
#include <math.h>
#include <stdio.h>

int main(void)
{
  double val = -1.0;

  do {
    printf("Тангенс %f равен %f.\n", val, tan(val));
    val += 0.1;
  } while(val<=1.0);

  return 0;
}
```

Зависимые функции

acos(), asin(), atan(), atan2(), cos(), sin(), sinh(), cosh() и tanh()

Семейство функций tanh

```
#include <math.h>
float tanhf(float arg);
double tanh(double arg);
long double tanhl(long double arg);
```

Функции tanhf() и tanhl() добавлены в версии C99.

Каждая функция семейства tanh() возвращает значение гиперболического тангенса от аргумента *arg*.

Пример

Данная программа выводит гиперболические тангенсы последовательности значений, лежащих в пределах от -1 до 1 и изменяющихся с шагом одна десятая, т.е. составляет таблицу гиперболических тангенсов чисел от -1 до 1.

```
#include <math.h>
#include <stdio.h>

int main(void)
{
  double val = -1.0;

  do {
    printf("Гиперболический тангенс %f равен %f.\n", val, tanh(val));
    val += 0.1;
  } while(val<=1.0);

  return 0;
}
```

Зависимые функции

acos(), asin(), atan(), atan2(), cos(), sin(), cosh(), sinh() и tan ()

Семейство функций tgamma

```
#include <math.h>
float tgammaf(float arg);
double tgamma(double arg);
long double tgammal(long double arg);
```

Функции tgamma(), tgammaf() и tgammal() добавлены в версии C99.

Каждая функция семейства tgamma() возвращает значение гамма-функции от аргумента *arg*.

Зависимые функции

lgamma()

Семейство функций trunc

```
#include <math.h>
float truncf(float arg);
double trunc(double arg);
long double truncl(long double arg);
```

Функции `trunc()`, `truncf()` и `truncl()` добавлены в версии C99.

Каждая функция семейства `trunc()` возвращает усеченное значение аргумента *arg*, т.е. значение, в котором отброшена дробная часть[1].

Зависимые функции

`nearbyint()`

[1] Иногда говорят, что это округленное значение аргумента *arg*, причем округление в данном случае выполняется отбрасыванием дробной части. — *Прим. ред.*