

Jarosław **Stańczyk**

NOWOCZESNY C

Przegląd **C23** z przykładami



Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz wydawca dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz wydawca nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Redaktor prowadzący: Małgorzata Kulik

Projekt okładki: Studio Gravite/Olsztyn
Obarek, Pokoński, Pazdrijowski, Zaprucki

Materiały graficzne na okładce zostały wykorzystane za zgodą Shutterstock.

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 230 98 63
e-mail: helion@helion.pl
WWW: <https://helion.pl> (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
<https://helion.pl/user/opinie/nowocp>
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

ISBN: 978-83-283-9995-2

Copyright © Helion S.A. 2023

Printed in Poland.

- Kup książkę
- Poleć książkę
- Oceń książkę

- Księgarnia internetowa
- Lubię to! » Nasza społeczność

Spis treści

Wstęp	17
Krótka historia języka C	18
Nowości i zmiany w C23	19
Jak zorganizowana jest ta książka	21
Kody źródłowe	21
W ramach rozrywki	21
I. Podstawy	
1. Warsztat	25
1.1. Narzędzia i środowisko pracy	25
1.2. Uruchamianie przykładów	26
1.2.1. Pierwszy program	26
1.2.2. Eksperymenty	29
2. Kod źródłowy	30
2.1. Struktura kodu źródłowego	31
2.1.1. Przykładowa struktura	32
2.2. Ograniczenia translacji	34
II. Koncepcja języka	
3. Elementy leksykalne	37
3.1. Zestawy znaków	37
3.1.1. Znaki wielobajtowe i rozszerzone	38
3.1.2. Sekwencje trój- i dwuznakowe	39
3.2. Komentarze	40
3.3. Identyfikatory	41
3.3.1. Słowa kluczowe	41
3.3.2. Kategorie identyfikatorów	42
3.3.3. Zakres identyfikatorów	43
3.3.4. Łączenie identyfikatorów	43
3.3.5. Identyfikator <code>__func__</code>	44
3.4. Deklaracje	44
3.4.1. Składnia deklaracji	45
3.4.2. Deklaratory	45
3.4.3. Inicjatory	47
3.4.4. Diagnostyka: <code>static_assert</code>	49

4. Podstawowe typy danych	50
4.1. Klasyfikacja typów	50
4.2. Typ <code>bool</code>	52
4.3. Typ <code>char</code>	53
4.4. Typy całkowitoliczbowe	53
4.4.1. Standardowe typy całkowitoliczbowe	53
4.4.2. Typ <code>_BitInt(N)</code>	56
4.4.3. Rozszerzone typy całkowitoliczbowe	57
4.5. Typy zmiennoprzecinkowe	58
4.5.1. Standardowe typy zmiennoprzecinkowe	58
4.5.2. Zmiennoprzecinkowe typy dziesiętne	60
4.5.3. Zmiennoprzecinkowe liczby zespolone	61
4.6. Typ <code>void</code>	62
5. Typy wyliczeniowe i typy pochodne	64
5.1. Typy wyliczeniowe	64
5.2. Tablice	67
5.2.1. Tablice o stałej długości	68
5.2.2. Tablice o zmiennej długości	71
5.2.3. Tablice wielowymiarowe	72
5.2.4. Łańcuchy znaków	73
5.3. Wskaźniki	75
5.3.1. Arytmetyka wskaźników	78
5.3.2. Tablice a wskaźniki	80
5.3.3. Wskaźniki do funkcji	80
5.4. Struktury, unie i pola bitowe	81
5.4.1. Struktury	81
5.4.2. Unie	87
5.4.3. Pola bitowe	88
6. Jeszcze o typach	91
6.1. Deklaracja <code>typedef</code>	91
6.2. Czas trwania obiektów	92
6.2.1. Klasy pamięci	93
6.3. Specyfikatory typów	98
6.4. Kwalifikatory typów	99
6.4.1. Kwalifikator <code>const</code>	100
6.4.2. Kwalifikator <code>volatile</code>	102
6.4.3. Kwalifikator <code>restrict</code>	102
6.4.4. Kwalifikator <code>_Atomic</code>	104
6.5. Specyfikator wyrównania	105
6.5.1. Specyfikator <code>alignas</code>	106
6.6. Specyfikatory atrybutów	107
6.6.1. Atrybuty standardowe	108
6.6.2. Testowanie atrybutów	112
6.7. Typy niekompletne	113
6.8. Typy zgodne	114

6.9.	Konwersja typów	115
6.9.1.	Konwersje arytmetyczne	116
6.9.2.	Konwersje typów w przypisaniach i wskaźnikach	119
7.	Stałe	121
7.1.	Stałe całkowitoliczbowe	121
7.1.1.	Rozszerzone typy całkowitoliczbowe	123
7.1.2.	Separator cyfr	123
7.2.	Stałe zmiennoprzecinkowe	124
7.2.1.	Ułamki dziesiętne	124
7.2.2.	Szesnastkowe stałe zmiennoprzecinkowe	125
7.3.	Stałe wyliczeniowe	125
7.4.	Stałe znakowe	126
7.4.1.	Znaki sterujące	127
7.4.2.	Uniwersalna nazwa znaku	128
7.5.	Literały łańcuchowe	129
7.6.	Stałe predefiniowane	129
8.	Wyrażenia i operatory	131
8.1.	Przetwarzanie wyrażeń	131
8.1.1.	Klasyfikatory wyrażeń	131
8.1.2.	Wybór ogólny	132
8.2.	Operatory	134
8.2.1.	Operatory arytmetyczne	136
8.2.2.	Operatory przypisania	137
8.2.3.	Operatory relacji	138
8.2.4.	Operatory logiczne	139
8.2.5.	Operatory bitowe	140
8.2.6.	Alternatywna notacja operatorów logicznych i bitowych	142
8.2.7.	Operator warunkowy <code>?:</code>	142
8.2.8.	Operator sekwencji <code>,</code>	144
8.2.9.	Operatory dostępu do pamięci	144
8.2.10.	Operator wywołania funkcji <code>()</code>	146
8.2.11.	Operator rzutowania <code>(typ)</code>	146
8.2.12.	Literały złożone <code>(typ){}</code>	147
8.2.13.	Operator <code>sizeof</code>	148
8.2.14.	Operator <code>alignof</code>	148
8.2.15.	Operatory typowania <code>typeof</code> i <code>typeof_unqual</code>	150
9.	Instrukcje	152
9.1.	Instrukcja prosta	152
9.2.	Bloki	152
9.3.	Instrukcje sterujące	153
9.3.1.	Instrukcje warunkowe	154
9.3.2.	Pętle	156
9.3.3.	Skoki bezwarunkowe	158

10. Funkcje	161
10.1. Funkcja <code>main</code>	161
10.1.1. Uruchamianie programu	161
10.1.2. Zakończenie programu	162
10.2. Prototypy funkcji	163
10.3. Definicje funkcji	164
10.3.1. Definicje w starym stylu	165
10.3.2. Funkcje anonimowe	166
10.4. Specyfikatory funkcji	166
10.4.1. Specyfikator <code>inline</code>	166
10.4.2. Specyfikator <code>_Noreturn</code>	167
10.5. Wywołanie funkcji, operator <code>()</code>	168
11. Dyrektywy preprocesora	169
11.1. Dołączanie plików	169
11.1.1. Dyrektywa <code>#include</code>	169
11.1.2. Makro <code>__has_include</code>	170
11.1.3. Dyrektywa <code>#embed</code>	171
11.1.4. Makro <code>__has_embed</code>	174
11.2. Makrodefinicje	175
11.2.1. Dyrektywa <code>#define</code>	175
11.2.2. Dyrektywa <code>#undef</code>	176
11.2.3. Operator <code>#</code>	177
11.2.4. Operator <code>##</code>	177
11.2.5. Zmienna liczba argumentów makra	178
11.3. Kompilacja warunkowa	179
11.3.1. <code>#if</code> , <code>#elif</code> , <code>#else</code> , <code>#endif</code>	179
11.3.2. Operator <code>defined</code>	180
11.3.3. <code>#ifdef</code> , <code>#elifdef</code> , <code>#ifndef</code> , <code>#elifndef</code>	181
11.4. Inne dyrektywy	182
11.4.1. Dyrektywa <code>#</code>	182
11.4.2. Dyrektywa <code>#line</code>	182
11.4.3. Dyrektywy <code>#error</code> i <code>#warning</code>	183
11.5. Predefiniowane standardowe makra	184
11.5.1. Makra wymagane przez C23	184
11.5.2. Makra opcjonalne	185
11.6. Dyrektywa <code>#pragma</code>	186
11.6.1. Operator <code>_Pragma</code>	187
 III. Biblioteka standardowa	
12. Wprowadzenie	191
13. Diagnostyka <code><assert.h></code>	193
13.1. Makra <code><assert.h></code>	193
13.1.1. Makro <code>NDEBUG</code>	193
13.1.2. Makro <code>assert</code>	193
13.2. Deklaracja <code>static_assert</code>	194

14. Arytmetyka liczb zespolonych <complex.h>	196
14.1. Konstrukcja liczb zespolonych i urojonych	197
14.1.1. Liczby zespolone	197
14.1.2. Liczby urojone	198
14.1.3. Konstrukcja liczb	199
14.2. Funkcje manipulujące liczbami zespolonymi	200
14.2.1. Operacje podstawowe	200
14.2.2. Funkcje trygonometryczne	201
14.2.3. Funkcje hiperboliczne	201
14.2.4. #pragma CX_LIMITED_RANGE	203
14.3. Dalszy rozwój <complex.h>	204
15. Klasyfikacja znaków <ctype.h>	205
15.1. Funkcje klasyfikujące	205
15.2. Funkcje konwersji znaków	206
15.3. Dalszy rozwój <ctype.h>	207
16. Obsługa błędów <errno.h>	208
16.1. Typy i makra	208
16.1.1. Typ errno_t	208
16.1.2. Makro errno	209
16.2. Dalszy rozwój <errno.h>	210
17. Środowisko zmiennoprzecinkowe <fenv.h>	211
17.1. Dyrektywy #pragma	213
17.1.1. #pragma FENV_ACCESS	213
17.1.2. #pragma FENV_ROUND	213
17.1.3. #pragma FENV_DEC_ROUND	214
17.2. Obsługa wyjątków zmiennoprzecinkowych	214
17.2.1. Makra	214
17.2.2. Funkcje obsługi wyjątków	214
17.3. Zaokrąglenia	219
17.3.1. Makra kierunku zaokrąglania	219
17.3.2. Funkcje obsługi zaokrągleń	219
17.4. Środowisko zmiennoprzecinkowe	222
17.4.1. Funkcja fegetenv	222
17.4.2. Funkcja fesetenv	222
17.4.3. Funkcja feholdexcept	224
17.4.4. Funkcja feupdateenv	224
17.5. Dalszy rozwój <fenv.h>	225
18. Charakterystyka typów zmiennoprzecinkowych <float.h>	226
18.1. Dalszy rozwój <float.h>	229

19. Konwersja formatu typów całkowitoliczbowych <inttypes.h>	230
19.1. Funkcje operujące na liczbach typu <code>intmax_t</code>	231
19.1.1. Funkcja <code>imaxabs</code>	231
19.1.2. Funkcja <code>imaxdiv</code>	231
19.1.3. Funkcje <code>strtoimax</code> i <code>strtoumax</code>	232
19.1.4. Funkcje <code>wcstoimax</code> i <code>wcstoumax</code>	232
19.2. Makra specyfikatorów formatu	232
19.3. Dalszy rozwój <inttypes.h>	233
20. Alternatywna notacja operatorów <iso646.h>	234
21. Charakterystyka typów całkowitoliczbowych <limits.h>	235
22. Ustawienia regionalne <locale.h>	238
22.1. Wprowadzenie	239
22.2. Ustawienia regionalne	239
22.2.1. Funkcja <code>setlocale</code>	239
22.2.2. Funkcja <code>localeconv</code>	241
22.3. Dalszy rozwój <locale.h>	243
23. Funkcje matematyczne <math.h>	244
23.1. Klasyfikacja i porównywanie liczb	249
23.2. Funkcje typów zmiennoprzecinkowych	251
23.2.1. Funkcje trygonometryczne i hiperboliczne	251
23.2.2. Funkcje wykładnicze i logarytmiczne	253
23.2.3. Pierwiastkowanie, potęgowanie, wartość bezwzględna	254
23.2.4. Funkcje błędu i funkcje gamma	254
23.2.5. Najbliższa wartość całkowita i zaokrąglenia	255
23.2.6. Operacja modulo	256
23.2.7. Manipulowanie wartościami	256
23.2.8. Różnica pozytywna, minimum i maksimum	257
23.2.9. Podstawowe operacje optymalizowane	258
23.3. Dodatkowe funkcje typów zmiennoprzecinkowych dziesiętnych	260
23.3.1. Funkcje kwantowe i kwantowe wykładnicze	260
23.3.2. Funkcje kodowania dziesiętnego	260
23.4. Rozszerzenia <math.h>	261
23.5. Dalszy rozwój <math.h>	262
24. Skoki odległe <setjmp.h>	263
24.1. Skoki odległe	263
24.1.1. Funkcja <code>setjmp</code>	263
24.1.2. Funkcja <code>longjmp</code>	264
24.1.3. Typ <code>jmp_buf</code>	265
25. Obsługa sygnałów <signal.h>	266
25.1. Wprowadzenie	266
25.2. Obsługa sygnałów	267
25.2.1. Funkcja <code>raise</code>	267
25.2.2. Funkcja <code>signal</code>	268

25.2.3. Typ <code>sig_atomic_t</code>	270
25.3. Dalszy rozwój <code><signal.h></code>	270
26. Wyrównywanie pamięci <code><stdalign.h></code>	271
27. Zmienna ilość argumentów funkcji <code><stdarg.h></code>	272
27.1. Funkcje o zmiennej ilości argumentów	272
27.1.1. Makro <code>va_start</code>	273
27.1.2. Makro <code>va_arg</code>	273
27.1.3. Makro <code>va_end</code>	273
27.1.4. Makro <code>va_copy</code>	274
28. Typy atomowe <code><stdatomic.h></code>	275
28.1. Wolność od blokady	277
28.1.1. Makra z rodziny <code>ATOMIC_typ_LOCK_FREE</code>	277
28.1.2. Funkcja <code>atomic_is_lock_free</code>	278
28.2. Typy atomowe i operacje na nich	278
28.2.1. Atomowe typy całkowitoliczbowe	278
28.2.2. Operacje atomowe	279
28.3. Inicjalizacja obiektów atomowych <code>atomic_init</code>	281
28.4. Szeregowanie pamięci	282
28.4.1. Typ <code>memory_order</code>	282
28.4.2. Makro <code>kill_dependency</code>	283
28.4.3. Bariery pamięci	283
28.5. Atomowe flagi i operacje na nich	284
28.5.1. Typ <code>atomic_flag</code>	284
28.5.2. Makro <code>ATOMIC_FLAG_INIT</code>	284
28.5.3. Funkcja <code>atomic_flag_test_and_set</code>	284
28.5.4. Funkcja <code>atomic_flag_clear</code>	284
28.6. Dalszy rozwój <code><stdatomic.h></code>	285
29. Narzędzia bitowe i bajtowe <code><stdbit.h></code>	286
29.1. Kolejność bajtów	287
29.2. Funkcje	287
29.2.1. Funkcje zliczające bity	288
29.2.2. Wyliczanie liczb binarnych	290
29.3. Dalszy rozwój <code><stdbit.h></code>	290
30. Wartości i typy logiczne <code><stdbool.h></code>	291
31. Kontrolowana arytmetyka liczb całkowitych <code><stdckdint.h></code>	292
31.1. Makra z rodziny <code>ckd_</code>	292
31.2. Dalszy rozwój <code><stdckdint.h></code>	293
32. Definicje wspólne <code><stddef.h></code>	294
32.1. Typy	294
32.1.1. Typ <code>max_align_t</code>	294
32.1.2. Typ <code>nullptr_t</code>	295
32.1.3. Typ <code>ptrdiff_t</code>	295
32.1.4. Typ <code>size_t</code>	296

32.1.5. Typ <code>rsize_t</code>	296
32.1.6. Typ <code>wchar_t</code>	296
32.2. Makra	298
32.2.1. Makro <code>NULL</code>	298
32.2.2. Makro <code>offsetof</code>	298
32.2.3. Makro <code>unreachable</code>	299
33. Typy całkowitoliczbowe <code><stdint.h></code>	301
33.1. Dalszy rozwój <code><stdint.h></code>	304
34. Standardowe wejście i wyjście <code><stdio.h></code>	305
34.1. Wprowadzenie	308
34.2. Funkcje obsługi błędów wejścia/wyjścia	309
34.2.1. Funkcja <code>clearerr</code>	309
34.2.2. Funkcja <code>feof</code>	310
34.2.3. Funkcja <code>ferror</code>	310
34.2.4. Funkcja <code>perror</code>	311
34.3. Operacja na plikach	311
34.3.1. Funkcja <code>remove</code>	311
34.3.2. Funkcja <code>rename</code>	311
34.3.3. Funkcje <code>tmpfile</code> , <code>tmpfile_s</code>	312
34.3.4. Funkcje <code>tmpnam</code> , <code>tmpnam_s</code>	313
34.4. Funkcje dostępu do plików	314
34.4.1. Funkcja <code>fclose</code>	314
34.4.2. Funkcja <code>fflush</code>	315
34.4.3. Funkcje <code>fopen</code> , <code>fopen_s</code>	316
34.4.4. Funkcje <code>freopen</code> , <code>freopen_s</code>	317
34.4.5. Funkcja <code>setbuf</code>	318
34.4.6. Funkcja <code>setvbuf</code>	319
34.5. Funkcje formatowanego wejścia/wyjścia	319
34.5.1. Formatowane wyjście: rodzina funkcji <code>fprintf</code>	319
34.5.2. Formatowane wejście – rodzina funkcji <code>fscanf</code>	328
34.6. Znakowe wejście/wyjście	332
34.6.1. Funkcje <code>fgetc</code> , <code>getc</code> i <code>getchar</code>	332
34.6.2. Funkcje <code>fputc</code> , <code>putc</code> i <code>putchar</code>	333
34.6.3. Funkcja <code>fgets</code>	333
34.6.4. Funkcja <code>gets_s</code>	334
34.6.5. Funkcje <code>fputs</code> i <code>puts</code>	334
34.6.6. Funkcja <code>ungetc</code>	334
34.7. Blokowe funkcje wejścia/wyjścia	335
34.7.1. Funkcja <code>fread</code>	335
34.7.2. Funkcja <code>fwrite</code>	336
34.8. Funkcje pozycjonowania plików	336
34.8.1. Funkcja <code>fgetpos</code>	336
34.8.2. Funkcja <code>fseek</code>	337
34.8.3. Funkcja <code>fsetpos</code>	337
34.8.4. Funkcja <code>ftell</code>	337

34.8.5. Funkcja <code>rewind</code>	338
34.9. Dalszy rozwój <code><stdio.h></code>	338
35. Narzędzia ogólne <code><stdlib.h></code>	339
35.1. Konwersja liczb i funkcje arytmetyczne	342
35.1.1. Konwersja numeryczna	342
35.1.2. Konwersja liczb zmiennoprzecinkowych dziesiętnych	346
35.1.3. Generowanie liczb pseudolosowych	347
35.1.4. Funkcje arytmetyczne liczb całkowitych	348
35.2. Dynamiczne zarządzanie pamięcią	349
35.2.1. Przydział pamięci	349
35.2.2. Zwalnianie pamięci	351
35.2.3. Wyrównanie pamięci	352
35.3. Komunikacja z systemem operacyjnym	352
35.3.1. Zakończenie programu	352
35.3.2. Komunikacja ze środowiskiem	356
35.4. Narzędzia do wyszukiwania i sortowania	358
35.4.1. Funkcje <code>qsort</code> , <code>qsort_s</code>	358
35.4.2. Funkcje <code>bsearch</code> , <code>bsearch_s</code>	359
35.5. Funkcje konwersji znaków oraz łańcuchów wielobajtowych i rozszerzonych	361
35.5.1. Konwersja znaków wielobajtowych i rozszerzonych	361
35.5.2. Konwersja łańcuchów wielobajtowych i rozszerzonych	363
35.6. Obsługa błędów ograniczeń zakresu	365
35.6.1. Typ <code>constraint_handler_t</code>	365
35.6.2. Funkcja <code>set_constraint_handler_s</code>	365
35.6.3. Funkcja <code>abort_handler_s</code>	366
35.7. Dalszy rozwój <code><stdlib.h></code>	367
36. Funkcje bez powrotu <code><stdnoreturn.h></code>	368
37. Obsługa łańcuchów <code><string.h></code>	369
37.1. Kopiowanie	371
37.1.1. Funkcje <code>strcpy</code> , <code>strcpy_s</code>	371
37.1.2. Funkcje <code>strncpy</code> , <code>strncpy_s</code>	372
37.1.3. Funkcja <code>strdup</code>	373
37.1.4. Funkcja <code>strndup</code>	374
37.2. Łączenie	374
37.2.1. Funkcje <code>strcat</code> , <code>strcat_s</code>	374
37.2.2. Funkcje <code>strncat</code> , <code>strncat_s</code>	376
37.3. Porównywanie	376
37.3.1. Funkcja <code>strcmp</code>	376
37.3.2. Funkcja <code>strncmp</code>	377
37.3.3. Funkcja <code>strcoll</code>	377
37.3.4. Funkcja <code>strxfrm</code>	378
37.4. Wyszukiwanie	379
37.4.1. Funkcja <code>strchr</code>	379
37.4.2. Funkcja <code>strcspn</code>	379
37.4.3. Funkcja <code>strpbrk</code>	380

37.4.4. Funkcja <code>strchr</code>	380
37.4.5. Funkcja <code>strspn</code>	380
37.4.6. Funkcja <code>strstr</code>	380
37.4.7. Funkcje <code>strtok</code> , <code>strtok_s</code>	381
37.5. Pozostałe funkcje	382
37.5.1. Funkcje <code>strerror</code> , <code>strerror_s</code>	382
37.5.2. Funkcja <code>strerrorlen_s</code>	383
37.5.3. Funkcje <code>strlen</code> , <code>strlen_s</code>	384
37.6. Obsługa bloków pamięci	384
37.6.1. Funkcja <code>memchr</code>	384
37.6.2. Funkcja <code>memcmp</code>	385
37.6.3. Funkcje <code>memcpy</code> , <code>memcpy_s</code>	385
37.6.4. Funkcja <code>memccpy</code>	386
37.6.5. Funkcje <code>memmove</code> , <code>memmove_s</code>	386
37.6.6. Funkcje <code>memset</code> , <code>memset_s</code>	386
37.7. Dalszy rozwój <code><string.h></code>	387
38. Funkcje matematyczne niezależne od typu <code><tgmath.h></code>	388
38.1. Makra wspólne dla typów rzeczywistych i zespolonych	388
38.2. Makra typów rzeczywistych	390
38.3. Makra typów zespolonych	393
38.4. Makra dziesiętnych typów zmiennoprzecinkowych	393
39. Wątki <code><threads.h></code>	394
39.1. Wątki	396
39.1.1. Obsługa wątków	396
39.1.2. Jednokrotne wywołanie funkcji	401
39.2. Dostęp do zasobów współdzielonych	402
39.2.1. Muteksy	402
39.2.2. Zmienne warunkowe	406
39.2.3. Pamięć lokalna wątku	410
39.3. Dalszy rozwój <code><threads.h></code>	413
40. Czas <code><time.h></code>	414
40.1. Typy i makra	416
40.1.1. Typ <code>clock_t</code>	416
40.1.2. Typ <code>struct timespec</code>	417
40.1.3. Typ <code>struct tm</code>	417
40.1.4. Typ <code>time_t</code>	418
40.1.5. Makro <code>CLOCKS_PER_SEC</code>	418
40.1.6. Makra z rodziny <code>TIME_</code>	419
40.2. Funkcje reprezentacji czasu	419
40.2.1. Funkcja <code>clock</code>	419
40.2.2. Funkcja <code>difftime</code>	419
40.2.3. Funkcja <code>time</code>	420
40.2.4. Funkcja <code>timespec_get</code>	421
40.2.5. Funkcja <code>timespec_getres</code>	421

40.3. Funkcje konwersji czasu	421
40.3.1. Funkcje asctime, asctime_r, asctime_s	421
40.3.2. Funkcje ctime, ctime_r, ctime_s	423
40.3.3. Funkcje gmtime, gmtime_r, gmtime_s	424
40.3.4. Funkcje localtime, localtime_r, localtime_s	424
40.3.5. Funkcje mktime, timegm	424
40.3.6. Funkcja strftime	425
40.4. Dalszy rozwój <time.h>	429
41. Obsługa znaków unicode <uchar.h>	430
41.1. Konwersja znaków UTF	431
41.1.1. Funkcje c8rtomb, c16rtomb, c32rtomb	431
41.1.2. Funkcje mbrtoc8, mbrtoc16, mbrtoc32	433
42. Obsługa znaków wielobajtowych i rozszerzonych <wchar.h>	435
42.1. Funkcje formatowanego wejścia/wyjścia	438
42.1.1. Stała WEOF	438
42.1.2. Formatowane wyjście – rodzina funkcji fwprintf	439
42.1.3. Formatowane wejście – rodzina funkcji fwscanf	441
42.2. Znakowe wejście/wyjście	442
42.2.1. Znakowe funkcje wyjścia	442
42.2.2. Znakowe funkcje wejścia	443
42.2.3. Funkcje dodatkowe	443
42.3. Manipulowanie łańcuchami	444
42.3.1. Konwersja numeryczna	444
42.3.2. Kopiowanie – wcsncpy, wcsncpy_s, wcsncpy, wcsncpy_s	445
42.3.3. Konkatenacja – wscat, wscat_s, wcsncat, wcsncat_s	446
42.3.4. Porównywanie – wcscmp, wcsncmp, wscoll, wcsxfrm	446
42.3.5. Przeszukiwanie	446
42.3.6. Pozostałe funkcje	447
42.3.7. Funkcje obsługi bloków pamięci	447
42.4. Konwersja znaków i łańcuchów	448
42.4.1. Konwersja znaków btowc, wctob	448
42.4.2. Konwersja znaków i łańcuchów wielobajtowych	449
42.5. Dalszy rozwój <wchar.h>	455
43. Klasyfikacja znaków rozszerzonych <wctype.h>	456
43.1. Funkcje klasyfikacji znaków	457
43.1.1. Funkcja iswctype	458
43.1.2. Funkcja wctype	459
43.2. Funkcje konwersji znaków	459
43.2.1. Funkcje towlower i towupper	459
43.2.2. Funkcja towctrans	459
43.2.3. Funkcja wctrans	460
43.3. Dalszy rozwój <wctype.h>	460
Literatura	461
Indeks	463

9. Instrukcje

Instrukcja określa czynność do wykonania, na przykład operację arytmetyczną lub wywołanie funkcji. Wiele instrukcji służy do sterowania przebiegiem programu poprzez definiowanie pętli i rozgałęzień. Instrukcje są przetwarzane kolejno, z wyjątkiem sytuacji, gdy instrukcje sterujące powodują skoki.

9.1. Instrukcja prosta

Każda instrukcja, która nie jest blokiem, jest zakończona średnikiem.

Składnia:

```
instrukcja(opcja);  
lub  
lista-atrybutów instrukcja;
```

Instrukcja składająca się tylko ze średnika nazywana jest *instrukcją pustą* i nie wykonuje żadnej operacji. Opcjonalnie instrukcja może być poprzedzona *listą-atrybutów* (patrz rozdział 6.). Uwaga, *lista-atrybutów* zakończona średnikiem tworzy deklarację atrybutu.

Przykład 9.1. Przykłady instrukcji prostych (fragment programu prg09-01.c)

```
1 y = x;                      // instrukcja przypisania  
2 printf("%d, %s\n", y, str); // wywołanie funkcji  
3  
4 for ( int i = 0; str[i] ; ++i )  
5 {  
6     ;    // instrukcja pusta  
7 }
```

9.2. Bloki

Blok, nazywany również *instrukcją złożoną*, grupuje kilka instrukcji w jedną. Blok może zawierać deklaracje. Deklaracje w bloku zwykle poprzedzają instrukcje. Norma zezwala jednak na umieszczanie deklaracji w dowolnym miejscu.

Składnia:

```

{
    lista deklaracji(opcja)
    lista instrukcji(opcja)
}

```

W dowolnym miejscu bloku może być definiowany nowy blok. Zwykle blok jest tworzony wszędzie tam, gdzie składnia wymaga instrukcji, natomiast program wymaga ich kilku. Dzieje się tak na przykład, gdy więcej niż jedna instrukcja ma zostać powtórzona w pętli. Blok rozpoczyna się i kończy nawiasem klamrowym (bez średnika na końcu). Instrukcje wewnątrz bloku zakończone są średnikiem.

Przykład 9.2. Instrukcje blokowe (fragment programu prg09-02.c)

```

1 {    // przykład bloku
2
3    // deklaracje
4    int i = 0;
5    static long a;
6    extern long max;
7
8    // instrukcje
9    ++a;
10   if (a >= max)
11   {
12       // blok zagnieżdżony
13       ...
14   }
15   ...
16 }
17 ...
18 {
19     // pusty blok
20 }

```

9.3. Instrukcje sterujące

Instrukcje sterujące służą do sterowania przebiegiem programu, zalicza się do nich:

— instrukcje warunkowe:

```

if ... else
switch

```

— pętle:

```

while
do ... while
for

```

— skoki:

```
goto
continue
break
return
```

Przeanalizujemy poszczególne z nich.

9.3.1. Instrukcje warunkowe

Instrukcja if

Instrukcja if powoduje *skok warunkowy*.

Składnia:

```
lista-atrybutów(opcja) if ( wyrażenie ) instrukcja
lub
lista-atrybutów(opcja) if ( wyrażenie ) instrukcja else instrukcja2
```

Pierwsze oceniane jest *wyrażenie*, które musi być typu skalarnego. Jeśli *wyrażenie* jest prawdziwe (tzn. wynik wyrażenia jest niezerowy), wykonywana jest *instrukcja*. W przeciwnym razie, jeśli występuje **else**, wykonywana jest *instrukcja2*.

Użycie **else** jest opcjonalne. Jeśli wartość *wyrażenia* wynosi 0 (*falsz*), a **else** nie występuje, wówczas realizacja programu jest kontynuowana od następnej instrukcji.

Jeśli zagnieżdżonych jest kilka instrukcji **if**, klauzula **else** zawsze należy do *ostatniej* klauzuli **if**, która nie ma jeszcze klauzuli **else**. **else** można przypisać do innego **if**, tworząc jawne bloki.

Przykład 9.3. Instrukcja **if** (fragment programu prg09-03.c)

```
1 if (x > y)
2     max = x;      // Przypisz większe z x i y do
3 else             // zmiennej max.
4     max = y;
5
6 if (n > 0)
7 {
8     if (n % 2 == 0)
9         puts ("n parzyste dodatnie");
10 }
11 else             // else należy do pierwszego if
12     puts ("n jest mniejsze od zera");
```

Pewne warunki można uprościć, na przykład zwykle stosuje się:

```
if ( wyrażenie )
```

zamiast

```
if ( wyrażenie != 0 )
```

Patrz operator warunkowy `?:` opisany w rozdziale 8.

Instrukcja `switch`

W instrukcji `switch` wartość *wyrażenia* jest porównywana ze stałymi skojarzonymi z etykietami przypadków `case`. Jeśli wynikiem wyrażenia jest stała skojarzona z etykietą przypadku, wykonywanie programu jest kontynuowane od pasującej etykiety. Jeśli nie ma pasującej etykiety, wykonanie programu przechodzi do etykiety domyślnej (`default`), jeśli jest zadeklarowana, w przeciwnym razie wykonanie jest kontynuowane od instrukcji występującej po instrukcji `switch`.

Składnia:

```
lista-atrybutów(opcja) switch ( wyrażenie ) instrukcja-switch
```

Wyrażenie jest wyrażeniem całkowitoliczbowym, a *instrukcja-switch* jest instrukcją blokową z etykietami `case` i co najwyżej jedną etykietą domyślną `default`.

Każda etykieta `case` ma postać:

```
lista-atrybutów(opcja) case wyrażenie-stałe: instrukcja-case
```

gdzie *wyrażenie-stałe* jest stałym wyrażeniem całkowitoliczbowym. Wszystkie stałe muszą się od siebie różnić.

Etykieta `default`, jeśli występuje, ma postać:

```
lista-atrybutów(opcja) default: instrukcja-default
```

Po skoku od `switch` do `case` wykonywanie programu jest kontynuowane sekwencyjnie, niezależnie od innych etykiet.

Instrukcja `break` może zostać użyta do wyjścia z bloku `switch` w dowolnym momencie. `break` jest konieczna, jeśli instrukcje następujące po innych etykietach `case` nie mają być wykonywane.

Przykład 9.4. Instrukcja `switch` (fragment programu prg09-04.c)

```

1 switch (polecenie) // w zależności od polecenia, np. od wartości
2 {                // wybranej przez użytkownika w menu
3     case 'a':
4     case 'A':     // jeśli wynikiem jest 'a' lub 'A'
5         akcja1 (); // wykonaj akcja1(),
6         break;    // a następnie opuść blok switch
7
8     case 'b':
9     case 'B':     // jeśli wynikiem jest 'b' lub 'B'
10        akcja2 (); // wykonaj akcja2()
11        break;    // i opuść blok
12
13    default:      // dla każdego innego wyniku polecenia:
14        putchar ('\a'); // wykonaj tę instrukcję
15 }
```

9.3.2. Pętle

Pętla składa się z instrukcji lub bloku zwanego *treścią pętli*, który jest wykonywany wielokrotnie, w zależności od warunku. C oferuje trzy instrukcje do tworzenia pętli:

- `while`
- `do...while`
- `for`

W każdej z tych pętli liczba wykonanych iteracji treści pętli jest określana przez *instrukcję sterującą*. Jest to wyrażenie typu skalarnego.

Instrukcja `while`

Składnia:

```
lista-atrybutów(opcja) while ( wyrażenie ) instrukcja
```

Instrukcja `while` jest pętlą *sterowaną od góry*: tzn. jako pierwszy sprawdzany jest warunek pętli (czyli *wyrażenie*). Jeśli zwróci wartość *prawda*, wykonywana jest treść pętli (czyli *instrukcja*), a następnie ponownie oceniany jest warunek pętli. Jeśli warunek pętli jest *fałszywy*, wykonywanie programu jest kontynuowane od pierwszej instrukcji występującej po treści pętli.

Przykład 9.5. Pętla `while` (fragment programu prg09-05.c)

```

1 s = str;           // niech wskaźnik s
2 while( *s != '\0' ) // wskazuje koniec łańcucha str
3     ++s;
```

Instrukcja do ... while

Składnia:

```
lista-atrybutów(opcja) do instrukcja while ( wyrażenie ) ;
```

Instrukcja **do ... while** jest pętlą *sterowaną od dołu*: najpierw wykonywana jest treść pętli, a następnie oceniane jest wyrażenie sterujące. Jest to powtarzane, dopóki wyrażenie sterujące nie będzie miało wartości *fałsz*.

Kluczową różnicą w stosunku do instrukcji **while** jest to, że treść pętli **do ... while** jest zawsze wykonywana przynajmniej raz. A pętla **while** może w ogóle nie zostać wykonana, ponieważ jej wyrażenie może już na starcie mieć wartość *false*.

Przykład 9.6. Pętla **do-while** (fragment programu prg09-06.c)

```
1 i = 0;
2 do                // kopiuj łańcuch str1
3     str2[i] = str1[i]; // do str2
4 while ( str1[i++] != '\0' );
```

Instrukcja for

Składnia:

```
lista-atrybutów(opcja)
for ( wyrażenie1(opcja); wyrażenie2(opcja); wyrażenie3(opcja) ) instrukcja
```

Typowa pętla **for** używa *wyrażenia2* jako sterującego i jest ono oceniane przed każdym wykonaniem pętli. Wyrażenie to musi być typu skalarnego. *wyrażenie3* jest wykonywane po realizacji każdej pętli. *wyrażenie1* inicjalizuje pętlę, jest wykonywane raz, przed rozpoczęciem pierwszej pętli, jeszcze przed oceną *wyrażenia2*. Jeśli *wyrażenie1* jest deklaracją, jej zakres ograniczony jest do pętli **for**. *wyrażenie1* i *wyrażenie3* mogą być dowolnymi wyrażeniami, można je pominąć. Pomińnięte *wyrażenie2* jest zastępowane przez niezerową stałą.

Pętla:

```
for ( wyrażenie1; wyrażenie2; wyrażenie3 ) instrukcja
```

jest równoważna pętli:

```
wyrażenie1
while ( wyrażenie2 )
{
    instrukcja
    wyrażenie3
}
```

Przykład 9.7. Pętla `for` i jej równoważnik `while` (fragment programu `prg09-07.c`)

```
1 for (int i = DELAY; i > 0; --i)
2     ;                // poczekaj chwilę
3
4 // równoważna pętla while:
5 int i = DELAY;       // inicjalizacja
6 while ( i > 0)        // test wyrażenia sterującego
7     --i;             // aktualizacja zmiennej i
```

Uwaga, powyższe pętle różnią się zakresem zmiennej `i`.

9.3.3. Skoki bezwarunkowe

Instrukcje skoku powodują bezwarunkowy skok do innego miejsca w kodzie.

Instrukcja `goto`

Instrukcja `goto` przeskakuje do dowolnego miejsca w funkcji, w której jest wywołana. Miejsce docelowe skoku określa nazwa etykiety.

Składnia:

```
lista-atrybutów(opcja) goto etykieta;
```

etykieta to nazwa, po której następuje dwukropek pojawiający się przed instrukcją, do której ma nastąpić skok. Ograniczeniem instrukcji `goto` jest to, że zarówno `goto`, jak i *etykieta* muszą być zawarte w tej samej funkcji.

Składnia:

```
lista-atrybutów(opcja) etykieta:
```

Przykład 9.8. Typowy skok przy użyciu `goto`

```
1 for ( ... )                // wyskok z
2     for ( ... )            // zagnieżdżonych pętli
3         if ( error )
4             goto obsluga_bledu;
5 ...
6 obsluga_bledu:             // tu zaczyna się obsługa błędów
7 ...
```

Biblioteka standardowa C dostarcza funkcje umożliwiające *skoki odległe*. Więcej o skokach odległych w rozdziale 24.

Instrukcja `continue`

Instrukcja `continue` może być używana tylko wewnątrz pętli. Powoduje ominięcie pozostałej części ciała pętli (bloku). Tak więc sterowanie wewnątrz `while` lub `do ... while` po natrafieniu na `continue` przechodzi do następnego testu *wyrażenia* sterującego, a w pętli `for` przeskakuje do oceny *wyrażenia*².

Składnia:

```
lista-atrybutów(opcja) continue;
```

Przykład 9.9. Instrukcja `continue`

```
for (i = -10; i < 10; ++i)
{
    ...
    if (i == 0)
        continue;    // dla wartości 0 pomiń resztę ciała pętli
    ...
}
```

Instrukcja `break`

Instrukcja `break` wyskakuje z pętli do następnej instrukcji za ciałem pętli lub instrukcji `switch`.

Składnia:

```
lista-atrybutów(opcja) break;
```

Przykład 9.10. Instrukcja `break`

```
1 while (1)
2 {
3     ...
4     if (polecenie == ESC)
5         break;    // jeśli polecenie ESC, zakończ pętlę
6     ...
7 }
```

Instrukcja `return`

Instrukcja `return` kończy wykonywanie bieżącej funkcji i zwraca kontrolę do obiektu wywołującego. W funkcji może pojawić się dowolna liczba instrukcji `return`. Wartość wyrażenia w instrukcji `return` jest zwracana wywołującemu jako wartość funkcji, w razie potrzeby wartość ta jest konwertowana na odpowiedni typ funkcji.

Składnia:

```
lista-atributów(opcja) return wyrażenie(opcja) ;
```

Wyrażenie w instrukcji **return** można pominąć. Ma to jednak sens tylko w funkcjach typu `void`, w takim przypadku całą instrukcję **return** można również pominąć. Wówczas funkcja zwraca kontrolę do obiektu wywołującego po zakończeniu swego bloku funkcyjnego.

Przykład 9.11. Instrukcja **return** (fragment programu `prg09-11.c`)

```
1 int max (int a, int b) // maksimum z a i b
2 {
3     return (a > b ? a : b);
4 }
```

Indeks

Atrybut

- _Noreturn, 108, 111, 368
- deprecated, 108, 109
- fallthrough, 108, 110
- maybe_unused, 108, 110
- nodiscard, 108, 111
- noreturn, 108, 111, 368
- reproducible, 108, 112
- unsequenced, 108, 112

Czas trwania obiektów, 92

- automatyczny, 92
- przydzielony, 93
- statyczny, 92
- wątkowy, 92

Definicja, 44

Deklaracja, 44–49

- _Static_assert,
 - słowo kluczowe, 42
- funkcji, 47
- prosta, 45
- static_assert,
 - słowo kluczowe, 42
- tablicy, 46
- typedef, słowo kluczowe,
 - 42, 91
- wskaźnika, 46
- złożona, 46

Digraf, sekwencja dwuznakowa, 40

Dyrektywa preprocesora, 169

- #, 182
- #define, 175
- #elif, 179
- #elifdef, 181
- #elifndef, 181
- #else, 179
- #embed, 171
- #embed
 - if.empty, 172
 - limit, 173
 - prefix, 174
 - suffix, 173
- #endif, 179
- #error, 183
- #if, 179

- #ifdef, 181

- #ifndef, 181

- #include, 169

- #line, 182

- #pragma, 186

- #pragma

- CX_LIMITED_RANGE, 196, 203

- FENV_ACCESS, 211, 213

- FENV_DEC_ROUND, 213, 214

- FENV_ROUND, 213, 219

- FP_CONTRACT, 60, 244

- #undef, 176

- #warning, 183

Funkcja

- _Exit, 341, 355

- abort, 341, 352

- abort_handler_s, 342, 366

- abs, 340, 348

- acos, 247, 251, 389

- acosd128, 247, 251

- acosd32, 247, 251

- acosd64, 247, 251

- acosh, 247, 251, 389

- acoshf, 247, 252, 389

- acoshd128, 247, 252

- acoshd32, 247, 252

- acoshd64, 247, 252

- acoshf, 247, 252, 389

- acoshl, 247, 252, 389

- acosl, 247, 251, 389

- acospi, 247, 251, 390

- acospid128, 247, 251

- acospid32, 247, 251

- acospid64, 247, 251

- acospif, 247, 251, 390

- acospil, 247, 251, 390

- aligned_alloc, 340, 349

- asctime, 415, 421

- asctime_r, 415, 421

- asctime_s, 416, 421

- asin, 247, 252, 389

- asind128, 247, 252

- asind32, 247, 252

- asind64, 247, 252

- asinf, 247, 252, 389

- asinh, 247, 252, 389

- asinhd128, 247, 252

- asinhd32, 247, 252

- asinhd64, 247, 252

- asinhf, 247, 252, 389

- asinhf, 247, 252, 389

- asinl, 247, 252, 389

- asinpif, 247, 252, 390

- asinpil, 247, 252, 390

- asinpil, 247, 252, 390

- at_quick_exit, 341, 354

- atan, 247, 252, 389

- atan2, 247, 252, 390

- atan2d128, 247, 252

- atan2d32, 247, 252

- atan2d64, 247, 252

- atan2f, 247, 252, 390

- atan2l, 247, 252, 390

- atan2pi, 247, 252, 391

- atan2pid128, 247, 252

- atan2pid32, 247, 252

- atan2pid64, 247, 252

- atan2pif, 247, 252, 391

- atan2pil, 247, 252, 391

- atand128, 247, 252

- atand32, 247, 252

- atand64, 247, 252

- atanf, 247, 252, 389

- atanh, 247, 252, 389

- atanhd128, 247, 252

- atanhd32, 247, 252

- atanhd64, 247, 252

- atanhf, 247, 252, 389

- atanhl, 247, 252, 389

- atanl, 247, 252, 389

- atanpi, 247, 252, 391

- atanpid128, 247, 252

- atanpid32, 247, 252

- atanpid64, 247, 252

- atanpif, 247, 252, 391

- atanpil, 247, 252, 391

- atexit, 341, 353, 355

- atof, 340, 342

- atoi, 340, 343

- atol, 340, 343
- atoll, 340, 343
- atomic_compare_exchange_strong, 276, 279
- atomic_compare_exchange_strong_explicit, 279
- atomic_compare_exchange_weak, 276, 279
- atomic_compare_exchange_weak_explicit, 279
- atomic_exchange, 276, 280
- atomic_exchange_explicit, 280
- atomic_fetch_add, 276, 280
- atomic_fetch_add_explicit, 280
- atomic_fetch_and, 276, 280
- atomic_fetch_and_explicit, 280
- atomic_fetch_or, 276, 280
- atomic_fetch_or_explicit, 280
- atomic_fetch_sub, 276, 280
- atomic_fetch_sub_explicit, 280
- atomic_fetch_xor, 276, 280
- atomic_fetch_xor_explicit, 280
- atomic_flag_clear, 277, 284
- atomic_flag_clear_explicit, 284
- atomic_flag_test_and_set, 277, 284
- atomic_flag_test_and_set_explicit, 284
- atomic_init, 276, 281
- atomic_is_lock_free, 276, 278
- atomic_load, 276, 281
- atomic_load_explicit, 281
- atomic_signal_fence, 276, 283
- atomic_store, 276, 281
- atomic_store_explicit, 281
- atomic_thread_fence, 276, 283
- bsearch, 341, 359
- bsearch_s, 342, 359
- btowc, 437, 448
- c16rtomb, 430, 431
- c32rtomb, 430, 431
- c8rtomb, 430, 431
- cabs, 196, 200, 389
- cabsf, 200, 389
- cabsl, 200, 389
- cacos, 196, 201, 389
- cacosf, 201, 389
- cacosh, 196, 201, 389
- cacoshf, 201, 389
- cacoshl, 201, 389
- cacosl, 201, 389
- call_once, 395, 401
- calloc, 340, 349
- canonicalize, 248, 257
- canonicalized128, 248, 257
- canonicalized32, 248, 257
- canonicalized64, 248, 257
- canonicalizef, 248, 257
- canonicalizel, 248, 257
- carg, 196, 200, 393
- cargf, 200, 393
- cargl, 200, 393
- casin, 196, 201, 389
- casinf, 201, 389
- casinh, 196, 201, 389
- casinhf, 201, 389
- casinhl, 201, 389
- casinl, 201, 389
- catan, 196, 201, 389
- catanf, 201, 389
- catanh, 196, 201, 389
- catanhf, 201, 389
- catanhl, 201, 389
- catanl, 201, 389
- cbrt, 247, 255, 391
- cbrtd128, 247, 255
- cbrtd32, 247, 255
- cbrtd64, 247, 255
- cbrtf, 247, 255, 391
- cbrtl, 247, 255, 391
- ccos, 196, 201, 389
- ccosf, 201, 389
- ccosh, 196, 201, 389
- ccoshf, 201, 389
- ccoshl, 201, 389
- ccosl, 201, 389
- ceil, 248, 255, 391
- ceild128, 248, 255
- ceild32, 248, 255
- ceild64, 248, 255
- ceilf, 248, 255, 391
- ceill, 248, 255, 391
- cexp, 196, 201, 389
- cexpf, 201, 389
- cexpl, 201, 389
- cimag, 196, 200, 393
- cimagf, 200, 393
- cimagl, 200, 393
- clearerr, 306, 309
- clock, 414, 419
- clog, 196, 201, 389
- clogf, 201, 389
- clogl, 201, 389
- cnd.broadcast, 395, 406
- cnd.destroy, 395, 407
- cnd.init, 395, 407
- cnd.signal, 395, 407
- cnd.timedwait, 395, 409
- cnd.wait, 395, 409
- compoundn, 247, 255, 391
- compoundnd128, 247, 255
- compoundnd32, 247, 255
- compoundnd64, 247, 255
- compoundnf, 247, 255, 391
- compoundnl, 247, 255, 391
- conj, 196, 201, 393
- conjf, 201, 393
- conjl, 201, 393
- copysign, 248, 257, 391
- copysignd128, 248, 257
- copysignd32, 248, 257
- copysignd64, 248, 257
- copysignf, 248, 257, 391
- copysignl, 248, 257, 391
- cos, 247, 252, 389
- cosd128, 247, 252
- cosd32, 247, 252
- cosd64, 247, 252
- cosf, 247, 252, 389
- cosh, 247, 252, 389
- coshd128, 247, 252
- coshd32, 247, 252
- coshd64, 247, 252
- coshf, 247, 252, 389
- coshl, 247, 252, 389
- cosl, 247, 252, 389
- cospi, 247, 252, 391
- cospid128, 247, 252
- cospid32, 247, 252
- cospid64, 247, 252
- cospihf, 247, 252, 391
- cospil, 247, 252, 391
- cpow, 196, 201, 389
- cpowf, 201, 389
- cpowl, 201, 389
- cproj, 196, 201, 393
- cprojf, 201, 393
- cprojl, 201, 393
- creal, 196, 200, 393
- crealf, 200, 393
- creall, 393
- csin, 196, 201, 389
- csinf, 201, 389
- csinh, 196, 201, 389
- csinhf, 201, 389
- csinhl, 201, 389
- csinl, 201, 389
- csqrt, 196, 201, 389
- csqrtf, 201, 389
- csqrtl, 201, 389
- ctan, 196, 201, 389
- ctanf, 201, 389
- ctanh, 196, 201, 389
- ctanhf, 201, 389
- ctanhl, 201, 389
- ctanl, 201, 389
- ctime, 415, 423
- ctime_r, 415, 423
- ctime_s, 416, 423
- d32addd128, 248, 258
- d32addd64, 248, 258
- d32divd128, 248, 259
- d32divd64, 248, 259
- d32fmad128, 248, 259

- d32fmad64, 248, 259
- d32muldd128, 248, 259
- d32muldd64, 248, 259
- d32sqrtd128, 248, 260
- d32sqrtd64, 248, 260
- d32subdd128, 248, 258
- d32subdd64, 248, 258
- d64adddd128, 248, 258
- d64divdd128, 248, 259
- d64fmad128, 248, 259
- d64muldd128, 248, 259
- d64sqrtd128, 248, 260
- d64subdd128, 248, 258
- daddl, 248, 258
- ddivl, 248, 259
- decodebind128, 249, 261
- decodebind32, 249, 261
- decodebind64, 249, 261
- decodedecdd128, 249, 261
- decodedecdd32, 249, 261
- decodedecdd64, 249, 261
- dfmal, 248, 259
- difftime, 414, 419
- div, 340, 348
- dmull, 248, 259
- dsqrtl, 248, 260
- dsubl, 248, 258
- encodebind128, 249, 261
- encodebind32, 249, 261
- encodebind64, 249, 261
- encodedecdd128, 249, 261
- encodedecdd32, 249, 261
- encodedecdd64, 249, 261
- erf, 248, 255, 391
- erfc, 248, 255, 391
- erfcd128, 248, 255
- erfcd32, 248, 255
- erfcd64, 248, 255
- erfcf, 248, 255, 391
- erfcl, 248, 255, 391
- erfd128, 248, 255
- erfd32, 248, 255
- erfd64, 248, 255
- erff, 248, 255, 391
- erfl, 248, 255, 391
- exit, 341, 354
- exp, 248, 253, 389
- exp10, 248, 253, 391
- exp10d128, 248, 253
- exp10d32, 248, 253
- exp10d64, 248, 253
- exp10f, 248, 253, 391
- exp10l, 248, 253, 391
- exp10m1, 248, 253, 391
- exp10m1d128, 248, 253
- exp10m1d32, 248, 253
- exp10m1d64, 248, 253
- exp10m1f, 248, 253, 391
- exp10m1l, 248, 253, 391
- exp2, 248, 253, 391
- exp2d128, 248, 253
- exp2d32, 248, 253
- exp2d64, 248, 253
- exp2f, 248, 253, 391
- exp2l, 248, 253, 391
- exp2m1, 248, 253, 391
- exp2m1d128, 248, 253
- exp2m1d32, 248, 253
- exp2m1d64, 248, 253
- exp2m1f, 248, 253, 391
- exp2m1l, 248, 253, 391
- expd128, 248, 253
- expd32, 248, 253
- expd64, 248, 253
- expf, 248, 253, 389
- expl, 248, 253, 389
- expm1, 248, 253, 391
- expm1d128, 248, 253
- expm1d32, 248, 253
- expm1d64, 248, 253
- expm1f, 248, 253, 391
- expm1l, 248, 253, 391
- fabs, 247, 255, 389
- fabsd128, 247, 255
- fabsd32, 247, 255
- fabsd64, 247, 255
- fabsf, 247, 255, 389
- fabsl, 247, 255, 389
- fadd, 248, 258
- faddl, 248, 258
- fclose, 306, 314
- fdim, 248, 257, 391
- fdimd128, 248, 257
- fdimd32, 248, 257
- fdimd64, 248, 257
- fdimf, 248, 257, 391
- fdiml, 248, 257, 391
- fdiv, 248, 259
- fdivl, 248, 259
- fe.dec.getround, 212, 219, 221
- fe.dec.setround, 212, 219, 221
- feclearexcept, 212, 215
- fegetenv, 213, 222
- fegetexceptflag, 212, 216
- fegetmode, 212, 219
- fegetround, 212, 219, 220
- feholdexcept, 213, 224
- feof, 306, 310
- feraiseexcept, 212, 217
- ferror, 306, 310
- fesetenv, 213, 222
- fesetexcept, 212, 217
- fesetexceptflag, 212, 217
- fesetmode, 212, 219
- fesetround, 212, 219, 220
- fetestexcept, 212, 218
- fetestexceptflag, 212, 218
- feupdateenv, 213, 224
- fflush, 306, 315
- ffma, 248, 259
- ffmaf, 248, 259
- fgetc, 306, 332
- fgetpos, 307, 336
- fgets, 306, 333
- fgetwc, 436, 443
- fgetws, 436, 443
- floor, 248, 255, 391
- floord128, 248, 255
- floord32, 248, 255
- floord64, 248, 255
- floorf, 248, 255, 391
- floorl, 248, 255, 391
- fma, 248, 258, 391
- fmad128, 248, 258
- fmad32, 248, 258
- fmad64, 248, 258
- fmaf, 248, 258, 391
- fmal, 248, 258, 391
- fmax, 248, 257, 391
- fmaxd128, 248, 257
- fmaxd32, 248, 257
- fmaxd64, 248, 257
- fmaxf, 248, 257, 391
- fmaximum, 248, 257, 391
- fmaximum_mag, 248, 257, 391
- fmaximum_mag_num, 248, 257, 391
- fmaximum_mag_numd128, 248, 257
- fmaximum_mag_numd32, 248, 257
- fmaximum_mag_numd64, 248, 257
- fmaximum_mag_numf, 248, 257, 391
- fmaximum_mag_numl, 248, 257, 391
- fmaximum_magd128, 248, 257
- fmaximum_magd32, 248, 257
- fmaximum_magd64, 248, 257
- fmaximum_magf, 248, 257, 391
- fmaximum_magl, 248, 257, 391
- fmaximum_num, 248, 257, 391
- fmaximum_numd128, 248, 257

- fmaximum_numd32, 248, 257
- fmaximum_numd64, 248, 257
- fmaximum_numf, 248, 257, 391
- fmaximum_numl, 248, 257, 391
- fmaximumd128, 248, 257
- fmaximumd32, 248, 257
- fmaximumd64, 248, 257
- fmaximumf, 248, 257, 391
- fmaximuml, 248, 257, 391
- fmaxl, 248, 257, 391
- fmin, 248, 257, 391
- fmind128, 248, 257
- fmind32, 248, 257
- fmind64, 248, 257
- fminf, 248, 257, 391
- fminimum, 248, 257, 391
- fminimum_mag, 248, 257, 391
- fminimum_mag_num, 248, 257, 391
- fminimum_mag_numd128, 248, 257
- fminimum_mag_numd32, 248, 257
- fminimum_mag_numd64, 248, 257
- fminimum_mag_numf, 248, 257, 391
- fminimum_mag_numl, 248, 257, 391
- fminimum_magd128, 248, 257
- fminimum_magd32, 248, 257
- fminimum_magd64, 248, 257
- fminimum_magf, 248, 257, 391
- fminimum_magl, 248, 257, 391
- fminimum_num, 248, 257, 391
- fminimum_numd128, 248, 257
- fminimum_numd32, 248, 257
- fminimum_numd64, 248, 257
- fminimum_numf, 248, 257, 391
- fminimum_numl, 248, 257, 391
- fminimumd128, 248, 257
- fminimumd32, 248, 257
- fminimumd64, 248, 257
- fminimumf, 248, 257, 391
- fminimuml, 248, 257, 391
- fminl, 248, 257, 391
- fmod, 248, 256, 391
- fmodd128, 248, 256
- fmodd32, 248, 256
- fmodd64, 248, 256
- fmodf, 248, 256, 391
- fmodl, 248, 256, 391
- fmul, 248, 259
- fmull, 248, 259
- fopen, 306, 316
- fopen.s, 307, 316
- fprintf, 306, 319
- fprintf.s, 307, 319
- fputc, 307, 333
- fputs, 307, 334
- fputwc, 436, 443
- fputws, 436, 442
- fread, 307, 335
- free, 340, 351
- free_aligned.sized, 341, 352
- free_sized, 340, 351
- freopen, 306, 317
- freopen.s, 307, 317
- frexp, 248, 253, 391
- frexpd128, 248, 253
- frexpd32, 248, 253
- frexpd64, 248, 253
- frexpf, 248, 253, 391
- frexpl, 248, 253, 391
- fromfp, 248, 256, 391
- fromfpd128, 248, 256
- fromfpd32, 248, 256
- fromfpd64, 248, 256
- fromfpf, 248, 256, 391
- fromfpl, 248, 256, 391
- fromfpx, 248, 256, 391
- fromfpxd128, 248, 256
- fromfpxd32, 248, 256
- fromfpxd64, 248, 256
- fromfpxf, 248, 256, 391
- fromfpxl, 248, 256, 391
- fscanf, 306, 328
- fscanf.s, 307, 328
- fseek, 307, 337
- fsetpos, 307, 337
- fsqrt, 248, 260
- fsqrtl, 248, 260
- fsub, 248, 258
- fsubl, 248, 258
- ftell, 307, 337
- fwide, 436, 443
- fwprintf, 436, 439
- fwprintf.s, 438, 439
- fwrite, 307, 336
- fwscanf, 436, 441
- fwscanf.s, 438, 441
- getc, 306
- getchar, 307, 332
- getenv, 341, 356
- getenv.s, 342, 356
- getpayload, 261
- getpayloadd128, 261
- getpayloadd32, 261
- getpayloadd64, 261
- getpayloadf, 261
- getpayloadl, 261
- gets, 307
- gets.s, 307, 334
- getwc, 436, 443
- getwchar, 436, 443
- gmtime, 415, 424
- gmtime.r, 415, 424
- gmtime.s, 416, 424
- hypot, 247, 255, 391
- hypotd128, 247, 255
- hypotd32, 247, 255
- hypotd64, 247, 255
- hypotf, 247, 255, 391
- hypotl, 247, 255, 391
- ignore_handler.s, 342, 367
- ilogb, 248, 253, 391
- ilogbd128, 248, 253
- ilogbd32, 248, 253
- ilogbd64, 248, 253
- ilogbf, 248, 253, 391
- ilogbl, 248, 253, 391
- imaxabs, 230, 231
- imaxdiv, 230, 231
- isalnum, 205, 206
- isalpha, 205, 206
- isblank, 205, 206
- iscntrl, 205, 206
- isdigit, 205, 206
- isgraph, 205, 206
- islower, 205, 206
- isprint, 205, 206
- ispunct, 205, 206
- isspace, 205, 206
- isupper, 205, 206
- iswalnum, 456, 457
- iswalpha, 456, 457
- iswblank, 456, 457
- iswcntrl, 456, 457
- iswctype, 456, 458
- iswdigit, 456, 457
- iswgraph, 456, 457
- iswlower, 456, 457
- iswprint, 456, 457
- iswpunct, 456, 457
- iswspace, 456, 457
- iswupper, 456, 457
- iswxdigit, 456, 457
- isxdigit, 205, 206
- labs, 340, 348
- ldexp, 248, 253, 391
- ldexpd128, 248, 253
- ldexpd32, 248, 253
- ldexpd64, 248, 253
- ldexpf, 248, 253, 391
- ldexpl, 248, 253, 391
- ldiv, 340, 348
- lgamma, 248, 255, 391
- lgammad128, 248, 255
- lgammad32, 248, 255
- lgammad64, 248, 255
- lgammaf, 248, 255, 391
- lgammal, 248, 255, 391
- llabs, 340, 348
- lldiv, 340, 348
- llint, 248, 256
- llintd128, 248, 256
- llintd32, 248, 256
- llintd64, 248, 256
- llintf, 248, 256

- llintl, 248, 256
- llogb, 248, 253, 391
- llogbd128, 248, 253
- llogbd32, 248, 253
- llogbd64, 248, 253
- llogbf, 248, 253
- llogbl, 248, 253
- llquantexpd128, 249, 260, 393
- llquantexpd32, 249, 260, 393
- llquantexpd64, 249, 260, 393
- llrint, 391
- llrintf, 391
- llrintl, 391
- llround, 248, 256, 392
- llroundd128, 248, 256
- llroundd32, 248, 256
- llroundd64, 248, 256
- llroundf, 248, 256, 392
- llroundl, 248, 256, 392
- localeconv, 238, 241
- localtime, 415, 424
- localtime_r, 415, 424
- localtime_s, 416, 424
- log, 248, 253, 389
- log10, 248, 254, 392
- log10d128, 248, 254
- log10d32, 248, 254
- log10d64, 248, 254
- log10f, 248, 254, 392
- log10l, 248, 254, 392
- log10p1, 248, 254, 392
- log10p1d128, 248, 254
- log10p1d32, 248, 254
- log10p1d64, 248, 254
- log10p1f, 248, 254, 392
- log10p1l, 248, 254, 392
- log1p, 248, 253, 392
- log1pd128, 248, 253
- log1pd32, 248, 253
- log1pd64, 248, 253
- log1pf, 248, 253, 392
- log1pl, 248, 253, 392
- log2, 248, 254, 392
- log2d128, 248, 254
- log2d32, 248, 254
- log2d64, 248, 254
- log2f, 248, 254, 392
- log2l, 248, 254, 392
- log2p1, 248, 254, 392
- log2p1d128, 248, 254
- log2p1d32, 248, 254
- log2p1d64, 248, 254
- log2p1f, 248, 254, 392
- log2p1l, 248, 254, 392
- logb, 248, 253, 392
- logbd128, 248, 253
- logbd32, 248, 253
- logbd64, 248, 253
- logbf, 248, 253, 392
- logbl, 248, 253, 392
- logd128, 248, 253
- logd32, 248, 253
- logd64, 248, 253
- logf, 248, 253, 389
- logl, 248, 253, 389
- logp1, 248, 253, 392
- logp1d128, 248, 253
- logp1d32, 248, 253
- logp1d64, 248, 253
- logp1f, 248, 253, 392
- logp1l, 248, 253, 392
- longjmp, 263, 264, 355
- lrint, 248, 256, 392
- lrintd128, 248, 256
- lrintd32, 248, 256
- lrintd64, 248, 256
- lrintf, 248, 256, 392
- lrintl, 248, 256, 392
- lround, 248, 256, 392
- lroundd128, 248, 256
- lroundd32, 248, 256
- lroundd64, 248, 256
- lroundf, 248, 256, 392
- lroundl, 248, 256, 392
- main, 161
- main
 - argc, 161
 - argv, 161
- malloc, 341, 350
- mblen, 341, 361
- mbrlen, 437, 450
- mbrtoc16, 430, 433
- mbrtoc32, 430, 433
- mbrtoc8, 430, 433
- mbrtowc, 437, 451
- mbsinit, 437, 449
- mbsrtowcs, 437, 453
- mbsrtowcs_s, 438, 453
- mbstowcs, 341, 363
- mbstowcs_s, 342, 363
- mbtowc, 341, 362
- memalign, 341, 352
- memccpy, 370, 386
- memchr, 370, 384
- memcmp, 370, 385
- memcpy, 370, 385
- memcpy_s, 371, 385
- memmove, 370, 386
- memmove_s, 371, 386
- memset, 370, 386
- memset_s, 371, 386
- mktime, 415, 424
- modf, 248, 254
- modfd128, 248, 254
- modfd32, 248, 254
- modfd64, 248, 254
- modff, 248, 254
- modfl, 248, 254
- mtx_destroy, 395, 403
- mtx_init, 395, 403
- mtx_lock, 395, 403
- mtx_timedlock, 395, 404
- mtx_trylock, 395, 404
- mtx_unlock, 395, 404
- nan, 248, 257
- nand128, 248, 257
- nand32, 248, 257
- nand64, 248, 257
- nanf, 248, 257
- nanl, 248, 257
- nearbyint, 248, 255, 392
- nearbyintd128, 248, 255
- nearbyintd32, 248, 255
- nearbyintd64, 248, 255
- nearbyintf, 248, 255, 392
- nearbyintl, 248, 255, 392
- nextafter, 248, 257, 392
- nextafterd128, 248, 257
- nextafterd32, 248, 257
- nextafterd64, 248, 257
- nextafterf, 248, 257, 392
- nextafterl, 248, 257, 392
- nextdown, 248, 257, 392
- nextdownd128, 248, 257
- nextdownd32, 248, 257
- nextdownd64, 248, 257
- nextdownf, 248, 257, 392
- nextdownl, 248, 257, 392
- nexttoward, 248, 257, 392
- nexttowardd128, 248, 257
- nexttowardd32, 248, 257
- nexttowardd64, 248, 257
- nexttowardf, 248, 257, 392
- nexttowardl, 248, 257, 392
- nextup, 248, 257, 392
- nextupd128, 248, 257
- nextupd32, 248, 257
- nextupd64, 248, 257
- nextupf, 248, 257, 392
- nextupl, 248, 257, 392
- perror, 306, 311

- pow, 247, 255, 389
- powd128, 247, 255
- powd32, 247, 255
- powd64, 247, 255
- powf, 247, 255, 389
- powl, 247, 255, 389
- pown, 247, 255, 392
- pownd128, 247, 255
- pownd32, 247, 255
- pownd64, 247, 255
- pownf, 247, 255, 392
- pownl, 247, 255, 392
- powr, 247, 255, 392
- powrd128, 247, 255
- powrd32, 247, 255
- powrd64, 247, 255
- powrf, 247, 255, 392
- powrl, 247, 255, 392
- printf, 306, 323
- printf_s, 307, 323
- putc, 307
- putchar, 307, 333
- puts, 307, 334
- putwc, 436, 443
- putwchar, 436, 443
- qsort, 341, 358
- qsort_s, 342, 358
- quantized128, 249, 260, 393
- quantized32, 249, 260, 393
- quantized64, 249, 260, 393
- quantumd128, 249, 260, 393
- quantumd32, 249, 260, 393
- quantumd64, 249, 260, 393
- quick_exit, 341, 355
- raise, 266, 267
- rand, 340, 347
- realloc, 341, 350, 367
- remainder, 248, 256, 392
- remainderd128, 248, 256
- remainderd32, 248, 256
- remainderd64, 248, 256
- remainderf, 248, 256, 392
- remainderl, 248, 256, 392
- remove, 306, 311
- remquo, 248, 256, 392
- remquod128, 248, 256
- remquod32, 248, 256
- remquod64, 248, 256
- remquof, 248, 256, 392
- remquol, 248, 256, 392
- rename, 306, 311
- rewind, 307, 338
- rint, 248, 255, 392
- rintd128, 248, 255
- rintd32, 248, 255
- rintd64, 248, 255
- rintf, 248, 255, 392
- rintl, 248, 255, 392
- rootn, 247, 255, 392
- rootnd128, 247, 255
- rootnd32, 247, 255
- rootnd64, 247, 255
- rootnf, 247, 255, 392
- rootnl, 247, 255, 392
- round, 248, 256, 392
- roundd128, 248, 256
- roundd32, 248, 256
- roundd64, 248, 256
- roundeven, 248, 256, 392
- roundevend128, 248, 256
- roundevend32, 248, 256
- roundevend64, 248, 256
- roundevenf, 248, 256, 392
- roundevenl, 248, 256, 392
- roundf, 248, 256, 392
- roundl, 248, 256, 392
- rsqrt, 247, 255, 392
- rsqrtid128, 247, 255
- rsqrtid32, 247, 255
- rsqrtid64, 247, 255
- rsqrtf, 247, 255, 392
- rsqrtl, 247, 255, 392
- samequantumd128, 249, 260, 393
- samequantumd32, 249, 260, 393
- samequantumd64, 249, 260, 393
- scalbln, 248, 254, 392
- scalblnd128, 248, 254
- scalblnd32, 248, 254
- scalblnd64, 248, 254
- scalblnf, 248, 254, 392
- scalblnl, 248, 254, 392
- scalbn, 248, 254, 392
- scalbnd128, 248, 254
- scalbnd32, 248, 254
- scalbnd64, 248, 254
- scalbnf, 248, 254, 392
- scalbnl, 248, 254, 392
- scanf, 306, 330
- scanf_s, 307, 330
- set.constraint_handler_s, 342, 365
- setbuf, 306, 318
- setjmp, 263
- setlocale, 238, 239
- setpayload, 261
- setpayloadadd128, 261
- setpayloadadd32, 261
- setpayloadadd64, 261
- setpayloadf, 261
- setpayloadl, 261
- setpayloadsig, 261
- setpayloadsigd128, 261
- setpayloadsigd32, 261
- setpayloadsigd64, 261
- setpayloadsigf, 261
- setpayloadsigl, 261
- setvbuf, 306, 319
- signal, 266
- sin, 247, 252, 389
- sind128, 247, 252
- sind32, 247, 252
- sind64, 247, 252
- sinf, 247, 252, 389
- singnal, 268
- sinh, 247, 252, 389
- sinhd128, 247, 252
- sinhd32, 247, 252
- sinhd64, 247, 252
- sinhf, 247, 252, 389
- sinhl, 247, 252, 389
- sinl, 247, 252, 389
- sinpi, 247, 252, 392
- sinpid128, 247, 252
- sinpid32, 247, 252
- sinpid64, 247, 252
- sinpif, 247, 252, 392
- sinpil, 247, 252, 392
- snprintf, 306, 324
- snprintf_s, 307, 324
- snwprintf_s, 438, 439
- sprintf, 306, 325
- sprintf_s, 307, 325
- sqrt, 247, 255, 389
- sqrtd128, 247, 255
- sqrtd32, 247, 255
- sqrtd64, 247, 255
- sqrtf, 247, 255, 389
- sqrtl, 247, 255, 389
- srand, 340, 347
- sscanf, 306, 330
- sscanf_s, 307, 330
- stdc.bit.ceiluc, 290
- stdc.bit.ceilui, 290
- stdc.bit.ceilul, 290
- stdc.bit.ceilull, 290
- stdc.bit.ceiulus, 290
- stdc.bit.flooruc, 290
- stdc.bit.floorui, 290
- stdc.bit.floorul, 290
- stdc.bit.floorull, 290
- stdc.bit.floorus, 290
- stdc.bit.widthuc, 290
- stdc.bit.widthui, 290
- stdc.bit.widthul, 290
- stdc.bit.widthull, 290
- stdc.bit.widthus, 290
- stdc.count.onesuc, 289
- stdc.count.onesui, 289
- stdc.count.onesul, 289
- stdc.count.onesull, 289
- stdc.count.onesus, 289
- stdc.count.zerosuc, 289
- stdc.count.zerosui, 289
- stdc.count.zerosul, 289
- stdc.count.zerosull, 289

- stdc_count_zerosus, 289
- stdc_first_leading_oneuc, 289
- stdc_first_leading_oneui, 289
- stdc_first_leading_oneul, 289
- stdc_first_leading_oneull, 289
- stdc_first_leading_oneus, 289
- stdc_first_leading_zerouc, 288
- stdc_first_leading_zeroui, 288
- stdc_first_leading_zeroul, 288
- stdc_first_leading_zeroull, 288
- stdc_first_leading_zerous, 288
- stdc_first_trailing_oneuc, 289
- stdc_first_trailing_oneui, 289
- stdc_first_trailing_oneul, 289
- stdc_first_trailing_oneull, 289
- stdc_first_trailing_oneus, 289
- stdc_first_trailing_zerouc, 289
- stdc_first_trailing_zeroui, 289
- stdc_first_trailing_zeroul, 289
- stdc_first_trailing_zeroull, 289
- stdc_first_trailing_zerous, 289
- stdc_has_single_bituc, 289
- stdc_has_single_bitui, 289
- stdc_has_single_bitul, 289
- stdc_has_single_bitull, 289
- stdc_has_single_bitus, 289
- stdc_leading_onesuc, 288
- stdc_leading_onesui, 288
- stdc_leading_onesul, 288
- stdc_leading_onesull, 288
- stdc_leading_onesus, 288
- stdc_leading_zerosuc, 288
- stdc_leading_zerosui, 288
- stdc_leading_zerosul, 288
- stdc_leading_zerosull, 288
- stdc_leading_zerosus, 288
- stdc_trailing_onesuc, 288
- stdc_trailing_onesui, 288
- stdc_trailing_onesul, 288
- stdc_trailing_onesull, 288
- stdc_trailing_onesus, 288
- stdc_trailing_zerosuc, 288
- stdc_trailing_zerosui, 288
- stdc_trailing_zerosul, 288
- stdc_trailing_zerosull, 288
- stdc_trailing_zerosus, 288
- strcat, 370, 374
- strcat_s, 371, 374
- strchr, 370, 379
- strcmp, 370, 376
- strcoll, 370, 377
- strcpy, 370, 371
- strcpy_s, 371
- strncpy, 370, 379
- strdup, 370, 373
- strerror, 370, 382
- strerror_s, 371, 382
- strerrorlen_s, 371, 383
- strfromd, 340, 344
- strfromd128, 340, 346
- strfromd32, 340, 346
- strfromd64, 340, 346
- strfromf, 340, 344
- strfroml, 340, 344
- strftime, 415, 425
- strlen, 370, 384
- strlen_s, 384
- strncat, 370, 376
- strncat_s, 371, 376
- strncmp, 370, 377
- strncpy, 370, 372
- strncpy_s, 371, 372
- strndup, 370, 374
- strnlen_s, 371
- strpbrk, 370, 380
- strrchr, 370, 380
- strspn, 370, 380
- strstr, 370, 380
- strtod, 340, 344
- strtod128, 340, 347
- strtod32, 340, 347
- strtod64, 340, 347
- strtof, 340, 344
- strtoimax, 230, 232
- strtok, 370, 381
- strtok_s, 371, 381
- strtol, 340, 345
- strtol_d, 340, 344
- strtol_l, 340, 345
- strtoul, 340, 345
- strtoull, 340, 345
- strtoumax, 230, 232
- strxfrm, 370, 378
- swprintf, 436, 439
- swprintf_s, 438, 439
- swscanf, 436, 442
- swscanf_s, 438, 442
- system, 341, 357
- tan, 247, 252, 389
- tand128, 247, 252
- tand32, 247, 252
- tand64, 247, 252
- tanf, 247, 252, 389
- tanh, 247, 252, 389
- tanhd128, 247, 252
- tanhd32, 247, 252
- tanhd64, 247, 252
- tanhf, 247, 252, 389
- tanh_l, 247, 252, 389
- tanl, 247, 252, 389
- tanpi, 247, 252, 392
- tanpid128, 247, 252
- tanpid32, 247, 252
- tanpid64, 247, 252
- tanpif, 247, 252, 392
- tanpil, 247, 252, 392
- tgamma, 248, 255, 392
- tgammad128, 248, 255
- tgammad32, 248, 255
- tgammad64, 248, 255
- tgammalf, 248, 255, 392
- tgammal, 248, 255, 392
- thrd_create, 395, 396
- thrd_current, 395, 398
- thrd_detach, 395, 397, 398
- thrd_equal, 395, 398
- thrd_exit, 395, 400
- thrd_join, 395, 397, 400
- thrd_sleep, 395, 400
- thrd_yield, 395, 401
- time, 414, 420
- timegm, 415, 424
- timespec_get, 414, 419, 421
- timespec_getres, 414, 421
- tmpfile, 306, 312, 355
- tmpfile_s, 307, 312
- tmpnam, 306, 313
- tmpnam_s, 307, 313
- tolower, 205, 206
- totalorder, 261
- totalorderd128, 261
- totalorderd32, 261
- totalorderd64, 261
- totalorderf, 261
- totalorderl, 261
- totalordermag, 261
- totalordermagd128, 261
- totalordermagd32, 261
- totalordermagd64, 261
- totalordermagf, 261
- totalordermagl, 261
- toupper, 205, 206
- towctrans, 456, 459
- tolower, 456, 459
- towupper, 456, 459
- trunc, 248, 256, 392
- truncd128, 248, 256
- truncd32, 248, 256
- truncd64, 248, 256
- truncf, 248, 256, 392
- trunc_l, 248, 256, 392
- tss_create, 396, 410
- tss_delete, 396, 410
- tss_get, 396, 411
- tss_set, 396, 411
- ufromfp, 248, 256, 392
- ufromfpd128, 248, 256
- ufromfpd32, 248, 256
- ufromfpd64, 248, 256
- ufromfpf, 248, 256, 392

ufromfpl, 248, 256, 392
 ufromfpx, 248, 256, 392
 ufromfpxd128, 248, 256
 ufromfpxd32, 248, 256
 ufromfpxd64, 248, 256
 ufromfpxf, 248, 256, 392
 ufromfpxl, 248, 256, 392
 ungetc, 307, 334, 338
 ungetwc, 436, 444
 vfprintf, 306, 326
 vfprintf_s, 307, 326
 vfscanf, 331
 vfscanf_s, 307, 331
 vfwprintf, 436, 440
 vfwprintf_s, 438, 440
 vfwscanf, 436, 442
 vfwscanf_s, 438, 442
 vprintf, 306, 327
 vprintf_s, 307, 327
 vscanf, 332
 vscanf_s, 307, 332
 vsnprintf, 306, 327
 vsnprintf_s, 307, 327
 vsnwprintf_s, 438, 440
 vsprintf, 306, 327
 vsprintf_s, 307, 327
 vsscanf, 332
 vsscanf_s, 307, 332
 vswprintf, 436, 440
 vswprintf_s, 438, 440
 vswscanf, 436, 442
 vswscanf_s, 438, 442
 vwprintf, 436, 440
 vwprintf_s, 438, 440
 wscanf, 436, 442
 wscanf_s, 438, 442
 wctomb, 437, 452
 wctomb_s, 438, 452
 wcsat, 436, 446
 wscat_s, 438, 446
 wscchr, 437, 446
 wscmp, 436, 446
 wscoll, 436, 446
 wscpy, 436, 445
 wscpy_s, 438, 445
 wscspn, 437, 446
 wcsftime, 437, 447
 wcslen, 437, 447
 wcslen_s, 447
 wcsncat, 436, 446
 wcsncat_s, 438, 446
 wcsncmp, 436, 446
 wcsncpy, 436, 445
 wcsncpy_s, 438, 445
 wcsnlen_s, 438
 wcsrchr, 437, 447
 wcsrchr, 437, 446
 wcsrtombs, 437, 453

wcsrtombs_s, 438, 453
 wcssp, 437, 446
 wcsstr, 437, 447
 wcstod, 436, 444
 wcstod128, 436, 445
 wcstod32, 436, 445
 wcstod64, 436, 445
 wcstof, 436, 444
 wcstoimax, 230, 232
 wcstok, 437, 447
 wcstok_s, 438, 447
 wcstol, 436, 445
 wcstold, 436, 444
 wcstoll, 436, 445
 wcstombs, 341, 364
 wcstombs_s, 342, 364
 wcstoul, 436, 445
 wcstoull, 436, 445
 wcstoumax, 230, 232
 wcsxfrm, 436, 446
 wctob, 437, 448
 wctomb, 341, 362
 wctomb_s, 342, 362
 wctrans, 456, 460
 wctype, 456, 459
 wmemchr, 437, 448
 wmemcmp, 437, 448
 wmemcpy, 437, 448
 wmemcpy_s, 438, 448
 wmemmove, 437, 448
 wmemmove_s, 438, 448
 wmemset, 437, 448
 wprintf, 436, 439
 wprintf_s, 438, 439
 wscanf, 436, 442
 wscanf_s, 438, 442

Funkcje

Definicja, 164
 Deklaracja, 43, 163
 Prototyp, 43, 163

Identyfikator, 41–44

func, 44
 kategoria identyfikatorów
 etykieta, 42
 identyfikator zwykły, 42
 składowa, 42
 znacznik, 42
 zakres identyfikatorów
 blok, 43
 funkcja, 43
 plik, 43
 prototyp funkcji, 43
 zewnętrzny, 41
 łączenie identyfikatorów
 brak, 44
 wewnętrzne, 44
 zewnętrzne, 44

Inicjalizacja, 47
 jawna, 47
 niejawna, 47

Instrukcja

Blok, 152
 break, słowo kluczowe,
 42, 159
 case, słowo kluczowe,
 42, 155
 continue,
 słowo kluczowe,
 42, 159
 default, słowo kluczowe,
 42, 155
 do, słowo kluczowe, 42,
 157
 else, słowo kluczowe,
 42, 154
 for, słowo kluczowe,
 42, 157
 goto, słowo kluczowe,
 42, 158
 if, słowo kluczowe, 42,
 154
 Prosta, 152
 return, słowo kluczowe,
 42, 159, 354
 switch, słowo kluczowe,
 42, 155
 while, słowo kluczowe,
 42, 156
 Złożona, 152

Komentarz, 40

Kwalifikator typu

_Atomic, słowo kluczowe,
 42, 99, 104
 const, słowo kluczowe,
 42, 99, 100
 restrict,
 słowo kluczowe,
 42, 99, 102
 volatile,
 słowo kluczowe,
 42, 99, 102

L-wartość, 131

Makro

_Complex_I, 196, 198
 _IOFBF, 305
 _IOLBF, 305
 _IONBF, 305
 _Imaginary_I, 196, 199
 _Noreturn, 368
 _PRINTF_NAN_LEN_MAX, 305
 DATE, 184
 FILE, 184

- __LINE__, 184
- __STDC.ANALYZABLE__, 185
- __STDC.ENDIAN_BIG__, 286, 287
- __STDC.ENDIAN_LITTLE__, 286, 287
- __STDC.ENDIAN_NATIVE__, 286, 287
- __STDC.HOSTED__, 184
- __STDC.IEC.559.COMPLEX__, 185, 197
- __STDC.IEC.559__, 185, 261
- __STDC.IEC.60559.BFP__, 185, 192, 244, 261
- __STDC.IEC.60559.COMPLEX__, 185, 197
- __STDC.IEC.60559.DFP__, 185, 192, 244, 261, 346
- __STDC.IEC.60559.TYPES__, 185
- __STDC.ISO.10646__, 185
- __STDC.LIB_EXT1__, 185, 191, 303, 307, 341, 365, 371, 416, 437
- __STDC.MB.MIGHT.NEQ.WC__, 185
- __STDC.NO.ATOMICS__, 185, 191, 275
- __STDC.NO.COMPLEX__, 61, 186, 191, 196, 197, 388
- __STDC.NO.THREADS__, 186, 191, 394
- __STDC.NO.VLA__, 186
- __STDC.UTF.16__, 184, 430
- __STDC.UTF.32__, 184, 430
- __STDC.VERSION.FENV.H__, 211
- __STDC.VERSION.MATH.H__, 245
- __STDC.VERSION.STDINT.H__, 302
- __STDC.VERSION.STDLIB.H__, 339
- __STDC.VERSION.TGMATH.H__, 388
- __STDC.VERSION.TIME.H__, 414
- __STDC.VERSION__, 184
- __STDC.WANT.IEC.60559.EXT__, 192, 244, 261
- __STDC.WANT.LIB_EXT1__, 303, 307, 341, 365, 371, 416, 437
- __STDC__, 184
- __TIME__, 184
- __bool.true.false.are.defined__, 291
- __has.c.attribute__, 112
- __has.embed__, 174
- __has.include__, 170
- acos, 389
- acosh, 389
- acospi, 390
- and, 234
- and.eq, 234
- asin, 389
- asinh, 389
- asinp, 390
- assert, 193
- atan, 389
- atan2, 390
- atan2pi, 391
- atanh, 389
- atanpi, 391
- ATOMIC.BOOL.LOCK.FREE, 275, 277
- ATOMIC.CHAR16.T.LOCK.FREE, 275, 277
- ATOMIC.CHAR32.T.LOCK.FREE, 275, 277
- ATOMIC.CHAR8.T.LOCK.FREE, 275, 277
- ATOMIC.CHAR.LOCK.FREE, 275, 277
- ATOMIC.FLAG.INIT, 277, 284
- ATOMIC.INT.LOCK.FREE, 275, 277
- ATOMIC.LLONG.LOCK.FREE, 275, 277
- ATOMIC.LONG.LOCK.FREE, 275, 277
- ATOMIC.POINTER.LOCK.FREE, 275, 277
- ATOMIC.SHORT.LOCK.FREE, 275, 277
- ATOMIC.WCHAR.T.LOCK.FREE, 275, 277
- bitand, 234
- BITINT.MAXWIDTH, 56, 235
- bitor, 234
- BOOL.MAX, 235
- BOOL.WIDTH, 235
- BUFSIZ, 305
- carg, 393
- cbrt, 391
- ceil, 391
- CHAR.BIT, 235
- CHAR.MAX, 235
- CHAR.MIN, 235
- CHAR.WIDTH, 235
- cimag, 393
- ckd.add, 292
- ckd.mul, 292
- ckd.sub, 292
- CLOCKS.PER.SEC, 414, 418
- CMPLX, 196, 199
- CMPLXF, 196, 199
- CMPLXL, 196, 199
- compl, 234
- compoundn, 391
- conj, 393
- copysign, 391
- cos, 389
- cosh, 389
- cospi, 391
- cproj, 393
- CR.DECIMAL.DIG, 226
- creal, 393
- d32add, 393
- d32div, 393
- d32fma, 393
- d32mul, 393
- d32sqrt, 393
- d32sub, 393
- d64add, 393
- d64div, 393
- d64fma, 393
- d64mul, 393
- d64sqrt, 393
- d64sub, 393
- dadd, 393
- DBL.DECIMAL.DIG, 227
- DBL.DIG, 227
- DBL.EPSILON, 227
- DBL.HAS.SUBNORM, 227, 229
- DBL.IS.IEC.60559, 227
- DBL.MANT.DIG, 227
- DBL.MAX, 227
- DBL.MAX.10.EXP, 227
- DBL.MAX.EXP, 227
- DBL.MIN, 227
- DBL.MIN.10.EXP, 227
- DBL.MIN.EXP, 227
- DBL.NORM.MAX, 227
- DBL.SNAN, 227
- DBL.TRUE.MIN, 227
- ddiv, 393
- DEC128.EPSILON, 228
- DEC128.MANT.DIG, 228
- DEC128.MAX, 228
- DEC128.MAX.EXP, 228
- DEC128.MIN, 228
- DEC128.MIN.EXP, 228
- DEC128.SNAN, 228
- DEC128.TRUE.MIN, 228
- DEC32.EPSILON, 228
- DEC32.MANT.DIG, 228
- DEC32.MAX, 228
- DEC32.MAX.EXP, 228
- DEC32.MIN, 228
- DEC32.MIN.EXP, 228
- DEC32.SNAN, 228
- DEC32.TRUE.MIN, 228
- DEC64.EPSILON, 228
- DEC64.MANT.DIG, 228
- DEC64.MAX, 228
- DEC64.MAX.EXP, 228
- DEC64.MIN, 228
- DEC64.MIN.EXP, 228
- DEC64.SNAN, 228
- DEC64.TRUE.MIN, 228
- DEC.EVAL.METHOD, 228, 245
- DEC.INFINITY, 228, 245
- DEC.NAN, 228, 245
- DECIMAL.DIG, 226, 229
- dfma, 393
- dmul, 393
- dsqrt, 393
- dsub, 393
- EDOM, 208
- EILSEQ, 208
- EOF, 205, 305
- ERANGE, 208, 345, 346
- erf, 391
- erfc, 391
- errno, 208, 209, 345, 346
- EXIT.FAILURE, 339, 355
- EXIT.SUCCESS, 339, 355
- exp, 389

- exp10, 391
- exp10m1, 391
- exp2, 391
- exp2m1, 391
- expm1, 391
- fabs, 389
- fadd, 393
- fdiv, 391
- fdiv, 393
- FE_ALL_EXCEPT, 212
- FE_DEC_DOWNWARD, 212
- FE_DEC_TONEAREST, 212
- FE_DEC_TONEARESTFROMZERO, 212
- FE_DEC_TOWARDZERO, 212
- FE_DEC_UPWARD, 212
- FE_DFL_ENV, 212
- FE_DFL_MODE, 212
- FE_DIVBYZERO, 212
- FE_DOWNWARD, 212
- FE_INEXACT, 212, 217
- FE_INVALID, 212
- FE_OVERFLOW, 212, 217
- FE_TONEAREST, 212
- FE_TONEARESTFROMZERO, 212
- FE_TOWARDZERO, 212
- FE_UNDERFLOW, 212, 217
- FE_UPWARD, 212
- ffma, 393
- FILENAME_MAX, 306
- floor, 391
- FLT_DECIMAL_DIG, 227
- FLT_DIG, 227
- FLT_EPSILON, 227
- FLT_EVAL_METHOD, 60, 226, 244
- FLT_HAS_SUBNORM, 227, 229
- FLT_IS_IEC_60559, 227
- FLT_MANT_DIG, 227
- FLT_MAX, 227
- FLT_MAX_10_EXP, 227
- FLT_MAX_EXP, 227
- FLT_MIN, 227
- FLT_MIN_10_EXP, 227
- FLT_MIN_EXP, 227
- FLT_NORM_MAX, 227
- FLT_RADIX, 226
- FLT_ROUNDS, 227
- FLT_SNAN, 227
- FLT_TRUE_MIN, 227
- fma, 391
- fmax, 391
- fmaximum, 391
- fmaximum_mag, 391
- fmaximum_mag_num, 391
- fmaximum_num, 391
- fmin, 391
- fminimum, 391
- fminimum_mag, 391
- fminimum_mag_num, 391
- fminimum_num, 391
- fmod, 391
- fmul, 393
- FOPEN_MAX, 306
- FP_FAST_D32ADDD128, 246
- FP_FAST_D32ADDD64, 246
- FP_FAST_D32DIVD128, 246
- FP_FAST_D32DIVD64, 246
- FP_FAST_D32FMAD128, 246
- FP_FAST_D32FMAD64, 246
- FP_FAST_D32MULD128, 246
- FP_FAST_D32MULD64, 246
- FP_FAST_D32SQRTD128, 246
- FP_FAST_D32SQRTD64, 246
- FP_FAST_D32SUBD128, 246
- FP_FAST_D32SUBD64, 246
- FP_FAST_D64ADDD128, 246
- FP_FAST_D64DIVD128, 246
- FP_FAST_D64FMAD128, 246
- FP_FAST_D64MULD128, 246
- FP_FAST_D64SQRTD128, 246
- FP_FAST_D64SUBD128, 246
- FP_FAST_DADDL, 246
- FP_FAST_DDIVL, 246
- FP_FAST_DFMAL, 246
- FP_FAST_DMULL, 246
- FP_FAST_DSQRTL, 246
- FP_FAST_DSUBL, 246
- FP_FAST_FADD, 246
- FP_FAST_FADDL, 246
- FP_FAST_FDIV, 246
- FP_FAST_FDIVL, 246
- FP_FAST_FFMA, 246
- FP_FAST_FFMAL, 246
- FP_FAST_FMA, 246
- FP_FAST_FMA128, 246
- FP_FAST_FMA32, 246
- FP_FAST_FMA64, 246
- FP_FAST_FMAF, 246
- FP_FAST_FMAL, 246
- FP_FAST_FMUL, 246
- FP_FAST_FMULL, 246
- FP_FAST_FSQRT, 246
- FP_FAST_FSQRTL, 246
- FP_FAST_FSUB, 246
- FP_FAST_FSUBL, 246
- FP_ILOGBO, 246
- FP_ILOGBNAN, 246
- FP_INFINITE, 246
- FP_INT_DOWNWARD, 246
- FP_INT_TONEAREST, 246
- FP_INT_TONEARESTFROMZERO, 246
- FP_INT_TOWARDZERO, 246
- FP_INT_UPWARD, 246
- FP_NAN, 246
- FP_NORMAL, 246
- FP_SUBNORMAL, 246
- FP_ZERO, 246
- fpclassify, 247, 249
- frexp, 391
- fromfp, 391
- fromfp_x, 391
- fsqrt, 393
- fsub, 393
- getc, 332
- HUGE_VAL, 245, 345
- HUGE_VAL_D128, 245
- HUGE_VAL_D32, 245
- HUGE_VAL_D64, 245
- HUGE_VALF, 245, 345
- HUGE_VALL, 245, 345
- hypot, 391
- I, 196, 199
- ilogb, 391
- INFINITY, 59, 226, 245
- INT16_C, 123, 303
- INT16_MAX, 302
- INT16_MIN, 302
- INT16_WIDTH, 302
- INT32_C, 123, 303
- INT32_MAX, 302
- INT32_MIN, 302
- INT32_WIDTH, 302
- INT64_C, 123, 303
- INT64_MAX, 302
- INT64_MIN, 302
- INT64_WIDTH, 302
- INT8_C, 123, 303
- INT8_MAX, 302
- INT8_MIN, 302
- INT8_WIDTH, 302
- INT_FAST16_MAX, 302
- INT_FAST16_MIN, 302
- INT_FAST16_WIDTH, 302
- INT_FAST32_MAX, 302
- INT_FAST32_MIN, 302
- INT_FAST32_WIDTH, 302
- INT_FAST64_MAX, 302
- INT_FAST64_MIN, 302
- INT_FAST64_WIDTH, 302
- INT_FAST8_MAX, 302
- INT_FAST8_MIN, 302
- INT_FAST8_WIDTH, 302
- INT_LEAST16_MAX, 302
- INT_LEAST16_MIN, 302
- INT_LEAST16_WIDTH, 302
- INT_LEAST32_MAX, 302
- INT_LEAST32_MIN, 302
- INT_LEAST32_WIDTH, 302
- INT_LEAST64_MAX, 302
- INT_LEAST64_MIN, 302
- INT_LEAST64_WIDTH, 302
- INT_LEAST8_MAX, 302
- INT_LEAST8_MIN, 302
- INT_LEAST8_WIDTH, 302
- INT_MAX, 235

INT.MIN, 235	log10, 392	PRIoMAX, 230
INT.WIDTH, 235	log10p1, 392	PRIoPTR, 230
INTMAX.C, 123, 303	log1p, 392	PRIo32, 230
INTMAX.MAX, 302	log2, 392	PRIo64, 230
INTMAX.MIN, 302	log2p1, 392	PRIoFAST32, 230
INTMAX.WIDTH, 302	logb, 392	PRIoFAST64, 230
INTPTR.MAX, 302	logp1, 392	PRIoLEAST32, 230
INTPTR.MIN, 302	LONG.MAX, 235, 346	PRIoLEAST64, 230
INTPTR.WIDTH, 302	LONG.MIN, 235, 346	PRIoMAX, 230
iscanonical, 247, 249	LONG.WIDTH, 235	PRIoPTR, 230
iseqsig, 247, 251	lrint, 392	PRIX32, 230
isfinite, 247, 249	lround, 392	PRIX32, 230
isgreater, 247, 251	MATH_ERREXCEPT, 246	PRIX64, 230
isgreaterequal, 247, 251	math_errhandling, 246	PRIX64, 230
isinf, 247, 249	MATH_ERRNO, 246	PRIXFAST32, 230
isless, 247, 251	MB_CUR_MAX, 339, 361	PRIXFAST32, 230
islessequal, 247, 251	MB_LEN_MAX, 235	PRIXFAST64, 230
islessgreater, 247, 251	NAN, 60, 226, 245	PRIXFAST64, 230
isnan, 247, 249	NDEBUG, 193	PRIXLEAST32, 230
isnormal, 247, 249	nearbyint, 392	PRIXLEAST32, 230
issignaling, 247, 249	nextafter, 392	PRIXLEAST64, 230
issubnormal, 247, 249	nextdown, 392	PRIXLEAST64, 230
isunordered, 247, 251	nexttoward, 392	PRIXMAX, 230
iszero, 247, 249	nextup, 392	PRIXMAX, 230
kill_dependency, 276, 283	noreturn, 368	PRIXPTR, 230
L_tmpnam, 306	not, 234	PRIXPTR, 230
L_tmpnam.s, 307	not_eq, 234	PTRDIFF.MAX, 303
LC.ALL, 238	NULL, 238, 294, 298, 306, 339, 369,	PTRDIFF.MIN, 303
LC.COLLATE, 238	414, 435	PTRDIFF.WIDTH, 302
LC.CTYPE, 238, 361, 456	offsetof, 294, 298	putc, 333
LC.MONETARY, 238	ONCE_FLAG_INIT, 394, 401	quantize, 393
LC.NUMERIC, 238	or, 234	quantum, 393
LC.TIME, 238	or_eq, 234	RAND.MAX, 339, 347
LDBL.DECIMAL_DIG, 227	pow, 389	remainder, 392
LDBL.DIG, 227	pown, 392	remquo, 392
LDBL.EPSILON, 227	powr, 392	rint, 392
LDBL.HAS_SUBNORM, 227, 229	PRId32, 230	rootn, 392
LDBL.IS_IEC.60559, 227	PRId64, 230	round, 392
LDBL.MANT_DIG, 227	PRIdFAST32, 230	roundeven, 392
LDBL.MAX, 227	PRIdFAST64, 230	RSIZE.MAX, 303
LDBL.MAX.10_EXP, 227	PRIdLEAST32, 230	rsqrt, 392
LDBL.MAX_EXP, 227	PRIdLEAST64, 230	samequantum, 393
LDBL.MIN, 227	PRIdMAX, 230	scalbn, 392
LDBL.MIN.10_EXP, 227	PRIdPTR, 230	scalbn, 392
LDBL.MIN_EXP, 227	PRIi32, 230	SCHAR.MAX, 235
LDBL.NORM.MAX, 227	PRIi64, 230	SCHAR.MIN, 235
LDBL.SNAN, 227	PRIiFAST32, 230	SCHAR.WIDTH, 235
LDBL.TRUE.MIN, 227	PRIiFAST64, 230	SCNd32, 230
ldexp, 391	PRIiLEAST32, 230	SCNd64, 230
lgamma, 391	PRIiLEAST64, 230	SCNdFAST32, 230
llogb, 391	PRIiMAX, 230	SCNdFAST64, 230
LLONG.MAX, 235, 346	PRIIPTR, 230	SCNdLEAST32, 230
LLONG.MIN, 235, 346	PRIo32, 230	SCNdLEAST64, 230
LLONG.WIDTH, 235	PRIo64, 230	SCNdMAX, 230
llquantexp, 393	PRIoFAST32, 230	SCNdPTR, 230
llrint, 391	PRIoFAST64, 230	SCNi32, 230
llround, 392	PRIoLEAST32, 230	SCNi64, 230
log, 389	PRIoLEAST64, 230	SCNiFAST32, 230

SCNiFAST64, 230
 SCNiLEAST32, 230
 SCNiLEAST64, 230
 SCNiMAX, 230
 SCNiPTR, 230
 SCNo32, 230
 SCNo64, 230
 SCNoFAST32, 230
 SCNoFAST64, 230
 SCNoLEAST32, 230
 SCNoLEAST64, 230
 SCNoMAX, 230
 SCNoPTR, 230
 SCNu32, 230
 SCNu64, 230
 SCNuFAST32, 230
 SCNuFAST64, 230
 SCNuLEAST32, 230
 SCNuLEAST64, 230
 SCNuMAX, 230
 SCNuPTR, 230
 SCNx32, 230
 SCNx64, 230
 SCNxFAST32, 230
 SCNxFAST64, 230
 SCNxLEAST32, 230
 SCNxLEAST64, 230
 SCNxMAX, 230
 SCNxPTR, 230
 SEEK_CUR, 306
 SEEK_END, 306
 SEEK_SET, 306
 SHRT_MAX, 235
 SHRT_MIN, 235
 SHRT_WIDTH, 235
 SIG_ATOMIC_MAX, 303
 SIG_ATOMIC_MIN, 303
 SIG_ATOMIC_WIDTH, 302
 SIG_DFL, 266
 SIG_ERR, 266
 SIG_IGN, 266
 SIGABRT, 266, 267
 SIGFPE, 266, 267
 SIGILL, 266, 267
 SIGINT, 266, 267
 signbit, 247, 249
 SIGSEGV, 266, 267
 SIGTERM, 266, 267
 sin, 389
 sinh, 389
 sinpi, 392
 SIZE_MAX, 303
 SIZE_WIDTH, 302
 sqrt, 389
 stdc.bit.ceil, 286, 290
 stdc.bit.floor, 286, 290
 stdc.bit.width, 286, 290
 stdc.count.ones, 286, 289

stdc.count.zeros, 286, 289
 stdc.first.leading.one, 286, 289
 stdc.first.leading.zero, 286, 288
 stdc.first.trailing.one, 286, 289
 stdc.first.trailing.zero, 286, 289
 stdc.has_single.bit, 286, 289
 stdc.leading.ones, 286, 288
 stdc.leading.zeros, 286, 288
 stdc.trailing.ones, 286, 288
 stdc.trailing.zeros, 286, 288
 tan, 389
 tanh, 389
 tanpi, 392
 tgamma, 392
 TIME_ACTIVE, 414, 419
 TIME_MONOTONIC, 414, 419
 TIME_THREAD_ACTIVE, 414, 419
 TIME_UTC, 414, 419
 TMP_MAX, 306
 TMP_MAX_S, 307
 trunc, 392
 TSS_DTOR_ITERATIONS, 396
 UCHAR_MAX, 235
 UCHAR_WIDTH, 235
 ufromfp, 392
 ufromfpx, 392
 UINT16_C, 123, 303
 UINT16_MAX, 302
 UINT16_WIDTH, 302
 UINT32_C, 123, 303
 UINT32_MAX, 302
 UINT32_WIDTH, 302
 UINT64_C, 123, 303
 UINT64_MAX, 302
 UINT64_WIDTH, 302
 UINT8_C, 123, 303
 UINT8_MAX, 302
 UINT8_WIDTH, 302
 UINT_FAST16_MAX, 302
 UINT_FAST16_WIDTH, 302
 UINT_FAST32_MAX, 302
 UINT_FAST32_WIDTH, 302
 UINT_FAST64_MAX, 302
 UINT_FAST64_WIDTH, 302
 UINT_FAST8_MAX, 302
 UINT_FAST8_WIDTH, 302
 UINT_LEAST16_MAX, 302
 UINT_LEAST16_WIDTH, 302
 UINT_LEAST32_MAX, 302
 UINT_LEAST32_WIDTH, 302
 UINT_LEAST64_MAX, 302
 UINT_LEAST64_WIDTH, 302
 UINT_LEAST8_MAX, 302
 UINT_LEAST8_WIDTH, 302
 UINT_MAX, 235
 UINT_WIDTH, 235
 UINTMAX_C, 123, 303
 UINTMAX_MAX, 302

UINTMAX_WIDTH, 302
 UINTPTR_MAX, 302
 UINTPTR_WIDTH, 302
 ULLONG_MAX, 235
 ULLONG_WIDTH, 235
 ULONG_MAX, 235, 346
 ULONG_MIN, 346
 ULONG_WIDTH, 235
 unreachable, 294, 299
 USHRT_MAX, 235
 USHRT_WIDTH, 235
 va_arg, 272, 273
 va_copy, 272, 274
 va_end, 272, 273
 va_start, 272, 273
 WCHAR_MAX, 303, 435
 WCHAR_MIN, 303, 435
 WCHAR_WIDTH, 302, 435
 WEOF, 435, 438, 456
 WINT_MAX, 303
 WINT_MIN, 303
 WINT_WIDTH, 302
 xor, 234
 xor_eq, 234

Makrodefinicja, 175

Argument
 __VA_ARGS__, 178
 __VA_OPT__, 178
 Operator
 ##, 177
 #, 177
 _Pragma, 187
 defined, 180

Obiekt, 44

Operator, 134
 !, 139, 234
 !=, 234
 ~, 140
 (*typ*), 146
 (*typ*){}, 147
 (), 146
 *, 136, 144
 *, 137
 +, 136
 ++, 136
 +=", 137
 ., 144
 -, 136
 --, 136
 -=, 137
 ->, 145
 ., 145
 /, 136
 /=", 137
 <, 138
 <<, 140
 <<=, 137

- <=, 138
 - =, 137
 - ==, 138
 - >, 138
 - >=, 138
 - >>, 140
 - >>=, 137
 - ?:, 142
 - [], 145
 - %, 136
 - %=, 137
 - &, 140, 144, 234
 - &=, 137, 234
 - &&, 139, 234
 - ^, 140, 234
 - ~=, 137, 234
 - _Alignof,
 - słowo kluczowe, 42
 - ~, 234
 - |, 234
 - |=, 234
 - ||, 139, 234
 - alignof, słowo kluczowe,
 - 42, 148
 - sizeof, słowo kluczowe,
 - 42, 148
 - typeof, słowo kluczowe,
 - 42, 150
 - typeof.unqual, słowo
 - kluczowe, 42, 150
- Plik nagłówkowy
- <assert.h>, 193–195
 - <complex.h>, 191, 196–204
 - <ctype.h>, 205–207
 - <errno.h>, 208–210
 - <env.h>, 191, 211–225
 - <float.h>, 226–229
 - <inttypes.h>, 230–233
 - <iso646.h>, 234
 - <limits.h>, 235–237
 - <locale.h>, 238–243
 - <math.h>, 191, 244–262
 - <setjmp.h>, 263–265
 - <signal.h>, 266–270
 - <stdalign.h>, 191, 271
 - <stdarg.h>, 272–274
 - <stdatomic.h>, 191, 275–285
 - <stdbit.h>, 191, 286–290
 - <stdbool.h>, 191, 291
 - <stdckdint.h>, 191, 292–293
 - <stddef.h>, 294–300
 - <stdint.h>, 191, 301–304
 - <stdio.h>, 191, 305–338
 - <stdlib.h>, 191, 339–367
 - <stdnoreturn.h>, 191, 368
 - <string.h>, 191, 369–387
 - <tgmath.h>, 191, 388–393
 - <threads.h>, 191, 394–413
 - <time.h>, 191, 414–429
 - <uchar.h>, 430–434
 - <wchar.h>, 191, 435–455
 - <wctype.h>, 456–460
- Priorytety Operatorów, 134
- Sekwencja
- dwuznakowa, 40
 - trójznakowa, 39
- Specyfikator funkcji
- _Noreturn, słowo kluczowe,
 - 42, 167
 - inline, słowo kluczowe,
 - 42, 166
- Specyfikator klasy pamięci
- _Thread_local, słowo
 - kluczowe, 42
 - auto, słowo kluczowe,
 - 42, 93
 - constexpr, słowo kluczowe,
 - 42, 93, 98
 - extern, słowo kluczowe,
 - 42, 93, 94
 - register, słowo kluczowe,
 - 42, 93, 95
 - static, słowo kluczowe,
 - 42, 93, 95
 - thread_local, słowo
 - kluczowe, 42, 93, 97,
 - 410
- Specyfikator typu
- signed, słowo kluczowe,
 - 42, 53
 - unsigned, słowo kluczowe,
 - 42, 53
- Specyfikator wyrównania
- _Alignas, słowo kluczowe,
 - 42
 - alignas, słowo kluczowe,
 - 42, 106
- Stała
- false, słowo kluczowe,
 - 42, 52, 130
 - memory_order_acq_rel,
 - 275, 282
 - memory_order_acquire,
 - 275, 282
 - memory_order_consume,
 - 275, 282
 - memory_order_relaxed,
 - 275, 282
 - memory_order_release,
 - 275, 282
 - memory_order_seq_cst,
 - 275, 282
 - mtx_plain, 395, 403
 - mtx_recursive, 395, 403
 - mtx_timed, 395, 403
 - nullptr, słowo kluczowe,
 - 42, 130
 - thrd_busy, 394
 - thrd_error, 394
 - thrd_nomem, 394
 - thrd_success, 394
 - thrd_timeout, 394
 - true, słowo kluczowe,
 - 42, 52, 130
- Strumień
- stderr, 305, 309
 - stdin, 305, 309
 - stdout, 305, 309
- Słowo kluczowe
- _Alignas, specyfikator
 - wyrównania, 42
 - _Alignof, operator, 42
 - _Atomic, kwalifikator
 - typu, 42, 99, 104
 - _BitInt, typ, 42, 56
 - _Bool, typ, 42, 52, 54
 - _Complex, typ, 42
 - _Decimal128, typ, 42, 60
 - _Decimal32, typ, 42, 60
 - _Decimal64, typ, 42, 60
 - _Generic, wybór ogólny,
 - 42, 132
 - _Imaginary, typ, 42
 - _Noreturn, specyfikator
 - funkcji, 42, 167
 - _Static_assert,
 - deklaracja, 42
 - _Thread_local,
 - specyfikator klasy
 - pamięci, 42
 - alignas, specyfikator
 - wyrównania, 42,
 - 106
 - alignof, operator, 42,
 - 148
 - auto, specyfikator klasy
 - pamięci, 42, 93
 - bool, typ, 42, 52, 54,
 - 235
 - break, instrukcja, 42,
 - 159
 - case, instrukcja, 42,
 - 155
 - char, typ, 42, 53, 235
 - const, kwalifikator
 - typu, 42, 99, 100
 - constexpr, specyfikator
 - klasy pamięci, 42,
 - 93, 98
 - continue, instrukcja,
 - 42, 159

- default, instrukcja, 42, 155
- do, instrukcja, 42, 157
- double, typ, 42, 58, 388
- else, instrukcja, 42, 154
- enum, typ, 42, 64
- extern, specyfikator klasy
 - pamięci, 42, 93, 94
- false, stała, 42, 52, 130
- float, typ, 42, 58, 388
- for, instrukcja, 42, 157
- goto, instrukcja, 42, 158
- if, instrukcja, 42, 154
- inline, specyfikator funkcji, 42, 166
- int, typ, 42, 53, 54, 205, 235
- long, typ, 42, 54, 235
- nullptr, stała, 42, 130
- register, specyfikator klasy
 - pamięci, 42, 93, 95
- restrict, kwalifikator typu, 42, 99, 102
- return, instrukcja, 42, 159, 354
- short, typ, 42, 54, 235
- signed, specyfikator typu, 42, 53
- sizeof, operator, 42, 148
- static, specyfikator klasy
 - pamięci, 42, 93, 95
- static_assert, 194
- static_assert, deklaracja, 42
- struct, typ, 42, 81
- switch, instrukcja, 42, 155
- thread_local, specyfikator
 - klasy pamięci, 42, 93, 97, 410
- true, stała, 42, 52, 130
- typedef, deklaracja, 42, 91
- typeof, operator, 42, 150
- typeof_unqual, operator, 42, 150
- union, typ, 42, 87
- unsigned, specyfikator typu, 42, 53
- void, typ, 42, 62
- volatile, kwalifikator typu, 42, 99, 102
- while, instrukcja, 42, 156
- _Complex, słowo kluczowe, 42
- _Decimal128, słowo kluczowe, 42, 60
- _Decimal32, słowo kluczowe, 42, 60
- _Decimal32_t, 244
- _Decimal64, słowo kluczowe, 42, 60
- _Decimal64_t, 244
- _Imaginary, słowo kluczowe, 42
- atomic_bool, 276
- atomic_char, 276
- atomic_char16_t, 276
- atomic_char32_t, 276
- atomic_char8_t, 276
- atomic_flag, 275, 284
- atomic_int, 276
- atomic_int_fast16_t, 276
- atomic_int_fast32_t, 276
- atomic_int_fast64_t, 276
- atomic_int_fast8_t, 276
- atomic_int_least16_t, 276
- atomic_int_least32_t, 276
- atomic_int_least64_t, 276
- atomic_int_least8_t, 276
- atomic_intmax_t, 276
- atomic_intptr_t, 276
- atomic_llong, 276
- atomic_long, 276
- atomic_ptrdiff_t, 276
- atomic_schar, 276
- atomic_short, 276
- atomic_size_t, 276
- atomic_uchar, 276
- atomic_uint, 276
- atomic_uint_fast16_t, 276
- atomic_uint_fast32_t, 276
- atomic_uint_fast64_t, 276
- atomic_uint_fast8_t, 276
- atomic_uint_least16_t, 276
- atomic_uint_least32_t, 276
- atomic_uint_least64_t, 276
- atomic_uint_least8_t, 276
- atomic_uintmax_t, 276
- atomic_uintptr_t, 276
- atomic_ullong, 276
- atomic_ulong, 276
- atomic_ushort, 276
- atomic_wchar_t, 276
- bool, słowo kluczowe, 42, 52, 54, 235
- char, słowo kluczowe, 42, 53, 235
- char16_t, 430
- char32_t, 430
- char8_t, 430
- clock_t, 414, 416
- cnd_t, 395
- complex, 196, 388
- constraint_handler_t, 342, 365
- div_t, 339
- double, słowo kluczowe, 42, 58, 388
- double_Complex, 61, 197
- double_Imaginary, 62, 198
- double_complex, 61, 198
- double_imaginary, 62, 198
- double_t, 244
- enum, słowo kluczowe, 42, 64
- errno_t, 208, 307, 342, 371, 416, 438
- femode_t, 211
- fenv_t, 211
- fexcept_t, 211
- FILE, 305
- float, słowo kluczowe, 42, 58, 388
- float_Complex, 61, 197
- float_Imaginary, 62, 198
- float_complex, 61, 198
- float_imaginary, 62, 198
- float_t, 244
- fpos_t, 305
- imaginary, 196
- imaxdiv_t, 230, 232
- int, słowo kluczowe, 42, 53, 54, 205, 235
- int16_t, 57, 301
- int32_t, 57, 301
- int64_t, 57, 301
- int8_t, 57, 301
- int_fast16_t, 57, 301
- int_fast32_t, 57, 301
- int_fast64_t, 57, 301
- int_fast8_t, 57, 301
- int_least16_t, 57, 123, 301
- int_least32_t, 57, 123, 301
- int_least64_t, 57, 123, 301
- int_least8_t, 57, 123, 301
- intmax_t, 57, 123, 301
- intptr_t, 57, 301
- jmp_buf, 263, 265
- ldiv_t, 339
- lldiv_t, 339
- long, słowo kluczowe, 42, 54, 235

Trygraf, sekwencja trójznakowa, 39

Typ

- _BitInt, słowo kluczowe, 42, 56
- _Bool, słowo kluczowe, 42, 52, 54

- long double, 58, 388
 - long double _Complex, 61, 197
 - long double _Imaginary, 62, 198
 - long double complex, 61, 198
 - long double imaginary, 62, 198
 - long int, 53, 54
 - long long, 54, 235
 - long long int, 53, 54
 - max_align_t, 294
 - mbstate_t, 430, 435
 - memory_order, 275, 282
 - mtx_t, 395
 - niekompletny, 113
 - nullptr_t, 294, 295
 - once_flag, 394
 - pole bitowe, 88
 - ptrdiff_t, 294, 295
 - QWchar_t, 435
 - rsize_t, 294, 296, 307, 342, 371, 416, 438
 - short, słowo kluczowe, 42, 54, 235
 - short int, 53, 54
 - sig_atomic_t, 266, 270, 302
 - signed, 54
 - signed _BitInt, 56
 - signed char, 53, 235
 - signed int, 54
 - signed long, 54
 - signed long int, 54
 - signed long long, 54
 - signed long long int, 54
 - signed short, 54
 - signed short int, 54
 - size_t, 294, 296, 302, 305, 339, 369, 414, 430, 435
 - struct, słowo kluczowe, 42, 81
 - struct lconv, 238
 - struct timespec, 414, 417
 - struct tm, 414, 417, 435
 - struktura, 81
 - tablica, 67
 - thr_start_t, 394, 396
 - thr_t, 394
 - thread_local, 396
 - time_t, 414, 418
 - tss_dtor_t, 396
 - tss_t, 396
 - uint16_t, 301
 - uint32_t, 301
 - uint64_t, 301
 - uint8_t, 301
 - uint_fast16_t, 301
 - uint_fast32_t, 301
 - uint_fast64_t, 301
 - uint_fast8_t, 301
 - uint_least16_t, 123, 301
 - uint_least32_t, 123, 301
 - uint_least64_t, 123, 301
 - uint_least8_t, 123, 301
 - uintmax_t, 58, 123, 301
 - uintptr_t, 301
 - unia, 87
 - union, słowo kluczowe, 42, 87
 - unsigend long, 54
 - unsigned, 54
 - unsigned _BitInt, 56
 - unsigned char, 205, 235
 - unsigned int, 54, 235
 - unsigned long, 235
 - unsigned long int, 54
 - unsigned long long, 54, 235
 - unsigned long long int, 54
 - unsigned short, 54, 235
 - unsigned short int, 54
 - va_list, 272
 - void, słowo kluczowe, 42, 62
 - wchar_t, 294, 296, 302, 339, 435
 - wctrans_t, 456
 - wctype_t, 456
 - wint_t, 302, 435, 456
 - wskaźnik, 75
 - zgodny, 114
 - łańcuch znaków, 73
- Wybór ogólny, 132
- _Generic, słowo kluczowe, 42, 132
- Wyrażenie, 131
- Zestaw znaków
- dodatkowy, 37
 - podstawowy, 37
 - wykonania, 37
 - źródłowy, 37
- Znak
- jednobajtowy, 38
 - rozszerzony, 38
 - wielobajtowy, 38

PROGRAM PARTNERSKI

— GRUPY HELION —



1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>

GRUPA
Helion 

ELEMENTARZ JĘZYKA C

Mimo że C — stworzony przez pracownika Laboratoriów Bella Dennisa Ritchiego — liczy sobie pół wieku, wciąż pozostaje niezwykle popularnym proceduralnym językiem ogólnego zastosowania. To dzięki niemu powstają elementy systemów operacyjnych i programów użytkowych. Tymczasem za sprawą Międzynarodowej Organizacji Normalizacyjnej (ISO) światło dzienne ujrzy najnowsza wersja standardu języka, która ukaże się pod koniec 2023 roku.

Właśnie tej iteracji poświęcony jest podręcznik Jarosława Stańczyka *Nowoczesny C. Przegląd C23 z przykładami*. Treść została podzielona na trzy części. W pierwszej autor opisuje budowę programu i kod źródłowy języka C, a także przedstawia narzędzia do kompilacji i uruchamiania przykładów dołączonych do książki. Jej trzon stanowią pozostałe dwie części, które zawierają między innymi omówienie języka C i standardowej biblioteki. Prezentowanym zagadnieniom towarzyszą liczne przykłady — przy każdym podano nazwę pliku źródłowego. Wszystkie kody źródłowe można samodzielnie pobrać z serwisu GitHub.

W książce między innymi:

- nowości i zmiany wprowadzone w C23
- elementy leksykalne
- podstawowe typy danych
- stałe, wyrażenia i operatory
- instrukcje, funkcje
- dyrektywy preprocesora
- biblioteka standardowa

Z językiem C za pan brat!

Jarosław Stańczyk — absolwent Politechniki Wrocławskiej, doktor nauk technicznych w dziedzinie informatyki. Pracował na Uniwersytecie w Magdeburgu jako stypendysta Niemieckiej Fundacji ds. Badań (Deutsche Forschungsgemeinschaft). Zajmuje się tworzeniem oprogramowania systemów wbudowanych. Pasjonat informatyki i robotyki, szczególnie zainteresowany emocjami maszyn i relacją człowiek – maszyna. Entuzjasta wolnego oprogramowania.

Helion



helion.pl



HELION SA
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

KOD KORZYŚCI
Sięgnij po więcej! ▶



ISBN 978-83-283-9995-2



Cena: 119,00 zł