# Tablice języka C

## Operatory C/C++

### Zestaw operatorów w Turbo C++ :

```
   []     ()     .      ->     ++     --
   &      *      +      -      ~      !
sizeof    /      %      <<     >>     <
   >      <=     >=     ==     !=     ^
   I      &&     ll     ?:     =      *=
   /=     %=     +=     —=     <<=    >>=
   &=     ^=     l=     ,      #      ##
```

### A następujące operatory są charakterystyczne dla C++:

```
::     .*     —>*
```

### Operatory arytmetyczne

| Nazwa operatora | Składnia |
|---|---|
| Dodawanie | a + b |
| Odejmowanie | a – b |
| Jednoargumentowy plus | +a |
| Jednoargumentowy minus | –a |
| Mnożenie | a * b |
| Dzielenie | a / b |
| Modulo (reszta z dzielenia) | a % b |
| Wzrost o 1 | ++a, a++ |
| Obniżenie o 1 | – –a, a– – |

### Operatory porównania

| Nazwa operatora | Składnia |
|---|---|
| Równe | a == b |
| Nie równe (różne) | a != b |
| Większe niż | a > b |
| Mniejsze niż | a < b |
| Większe lub równe | a >= b |
| Mniejsze lub równe | a <= b |

### Operatory logiczne

| Nazwa operatora | Składnia |
|---|---|
| Logiczna negacja (NOT) | !a |
| Logiczne AND | a && b |

| | |
|---|---|
| Logiczne OR | a \|\| b |

## Operatory na bitach

| Nazwa operatora | Składnia |
|---|---|
| Negacja bitowa (NOT) | ~a |
| Koniunkcja bitowa (AND) | a & b |
| Alternatywa bitowa (OR) | a \| b |
| Alternatywa rozłączna (XOR) | a ^ b |
| Przesunięcie bitów w lewo | a << b |
| Przesunięcie bitów w prawo | a >> b |

## Operatory przypisania

| Nazwa operatora | Składnia | Znaczenie |
|---|---|---|
| Podstawienie/przypisanie | a = b | a = b |
| Dodawanie | a += b | a = a + b |
| Odejmowanie | a –= b | a = a – b |
| Mnożenie | a *= b | a = a * b |
| Dzielenie | a /= b | a = a / b |
| Modulo (reszta z dzielenia) | a %= b | a = a % b |
| Bitowy AND | a &= b | a = a & b |
| Bitowy OR | a \|= b | a = a \| b |
| Bitowy XOR | a ^= b | a = a ^ b |
| Przesunięcie bitów w lewo | a <<= b | a = a << b |
| Przesunięcie bitów w prawo | a >>= b | a = a >> b |

## Operator warunkowy

| Operator | Przykład |
|---|---|
| rezultat = wyrażenie ? wartość_1 : wartość_2 | max = (a>b) ? a : b ; |

## Operatory podelementów (‘members’) i wskaźników

| Nazwa operatora | Składnia |
|---|---|
| Indeks | a[b] |
| Przekierowanie | *a |
| Adres elementu | &a |
| Dereferencja w strukturze (member b obiektu a, wskazany przez a) | a–>b |
| Referencja w strukturze (member b obiektu a) | a.b |
| Member wybrany przez ‘wskaźnik do membra’ b obiektu wskazany przez a | a–>*b |
| Member obiektu a wybrany przez ‘wskaźnik do membra’ b | a.*b |

## Inne operatory

| Nazwa operatora | Składnia |
|---|---|
| Wywołanie funkcji | a(a1, a2) |
| Przecinek | a, b |
| Instrukcja warunkowa | a ? b : c |
| *Scope resolution* | a::b |

| | |
|---|---|
| Literał zdefiniowany przez użytkownika | ”a” |
| Rozmiar (*sizeoff*) | sizeof (a)<br>sizeof(*type*) |
| Rozmiar parametru | sizeof…(Arg) |
| *Alignof* | alignof(*type*) |
| Typ identyfikacyjny | typeid(a)<br>typeid(*type*) |
| Wymiana typu (rzut) | (*type*)a |
| Wymiana typu | type(a) |
| Wymiana *static_cast* | static_cast<type>(a) |
| Wymiana *dynamic_cast* | dynamic_cast<type>(a) |
| Wymiana *constant_cast* | const_cast<type>(a) |
| Wymiana *reinterpret_cast* | reinerpret_cast<type>(a) |
| Alokowania pamięci | new type |
| Alokowania pamięci dla tablicy | new type[n] |
| Uwalnianie pamięci | delete a |
| Uwalnianie pamięci tablicy | delete [] a |
| *Exception check* | noexcept (a) |

# Kolejność wykonywania operatorów w C

| Operator | Symbole operatorów | Działanie w stronę |
|---|---|---|
| dołączania | `() [] -> :: .` | od lewej do prawej |
| jednoargumentowy | `! ~ + - ++ -- & * (typecast) sizeof new delete` | **od prawej do lewej** |
| dostępu | `.* ->*` | od lewej do prawej |
| mnożeniowy | `* / %` | od lewej do prawej |
| dodawania | `+ -` | od lewej do prawej |
| przesunięcia | `<< >>` | od lewej do prawej |
| relacyjny | `< <= > >=` | od lewej do prawej |
| równości | `== !=` | od lewej do prawej |
| AND bitowy | `&` | od lewej do prawej |
| XOR bitowy | `^` | od lewej do prawej |
| OR bitowy | `\|` | od lewej do prawej |
| AND logiczny | `&&` | od lewej do prawej |
| OR logiczny | `\|\|` | od lewej do prawej |
| warunkowy | `?:` (wyrażenie warunkowe) | **od prawej do lewej** |
| przypisania | `= *= /= %= += –= &= ^= \|= <<= >>=` | **od prawej do lewej** |
| przecinek | `,` | od lewej do prawej |

## Wyjaśnienie funkcji operatorów

Niektóre operatory mogą być w więcej niż jednej grupie

### Operatory poprzedzające obiekt i następujące po obiekcie

**[ ]** Operator indeksu łańcucha znakowego

**( )** Operator wywoływania funkcji

**.** (kropka) Operator dostępu do składników struktur/union

**->** Operator wskaźnika dostępu do składników struktur/union

**++** Operator poprzedzający obiekt lub po nim następujący, np. **++i**, **i++**

**– –** Operator poprzedzający obiekt lub po nim następujący, np. **– – i**, **i – –**

## Operatory jednoargumentowe

**&** Operator adresu

***** Operator pośredniczący, definiujący pointer

**+** Jednoargumentowe dodawanie

**–** Jednoargumentowe odejmowanie)

**~** Odwracanie bitów - 0 na 1 i 1 na 0 czyli w efekcie: dopełnienie do 1

**!** Logiczna negacja

**Increment and decrement operators**

**++** Operator poprzedzający obiekt lub po nim następujący. Podniesienie wartości zmiennej o 1 przed lub po jakimikolwiek innymi operacjami aktualnej instrukcji.

**– –** Operator poprzedzający obiekt lub po nim następujący. Obniżenie wartości zmiennej o 1 przed lub po jakimikolwiek innymi operacjami aktualnej instrukcji.

## Operator sizeof

**sizeof(*obiekt*)** Operator rozmiaru obiektu. Daje rozmiar obiektu w bajtach.

## Operatory dostępu dla obiektów, które należą do obiektów nadrzędnych

**Występuje tu wzajemna zależność dwóch obiektów. Np. dostanie się do struktury a stamtąd do zmiennej tej struktury. Obiekty po lewej stronie operatora zawierają w sobie obiekty po prawej stronie operatora.**

**Myśl tak: Miasto.Ulica.Nr_budynku.Nr_mieszkania**

**albo: Miasto->Ulica->Nr_budynku->Nr_mieszkania**

**.** (kropka) Dostęp bezpośredni czyli po nazwach, bez jawnego powołania się na numer pierwszej komórki obiektu (zmiennej)

**->** Dostęp po numerze komórki pamięci, pointerze, który ten numer trzyma)

## Operatory dla elementów klasy

**::** Odwołanie się do określonego zakresu dostępu, np. `std::string`

**.*** Dostęp poprzez pointer do elementu klasy

**->*** Dostęp poprzez pointer do pointera elementu klasy

**:** Inicjalizator klasy, konstruktora, itd. zgodnie z regułami 'inicjalizacji'

## Operatory mnożeniowe

* Multiply (Mnożenie)

/ Divide (Dzielenie)

% Remainder (Reszta z dzielenia)

## Operatory dodawania

There are two additive operators: **+** and **—**.

**+** Dodawanie

**-** Odejmowanie

## Operatory przesunięcia bitów

**<<** Przesunięcie bitów w lewo; pojawiające się wolne komórki po prawej stronie przyjmują bit 0

**>>** Przesunięcie bitów w prawo; wylatujące bity po prawej stronie są tracone

## Operatory testu porównania

- **<** 'Wartość lewej strony' jest mniejsza niż 'wartość prawej strony'
- **>** 'Wartość lewej strony' jest większa niż 'wartość prawej strony'
- **<=** 'Wartość lewej strony' jest mniejsza lub równa 'wartości prawej strony'
- **>=** 'Wartość lewej strony' jest większa lub równa 'wartości prawej strony'

## Operatory testu równości

- **==** 'Wartość lewej strony' jest równa 'wartości prawej strony'
- **!=** 'Wartość lewej strony' jest różna od 'wartości prawej strony'

## Logiczne operatory bitowe

- **&** Bitowe AND/i
- **^** Bitowe wykluczające się OR/lub: 1 lub 1 daje 0
- **|** Bitowe niewykluczające się OR/lub: 1 lub 1 daje 1

### Działania operatorów bitowych

| Wartość bitu w E1 | Wartość bitu w E2 | E1 & E2 | E1 ^ E2 | E1 \| E2 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |

```
#include<stdio.h>
int main()
{   printf("1 & 1    daje %d \n", (1 & 1) );
    printf("1 | 1    daje %d \n", (1 | 1) );
    printf("1 ^ 1    daje %d \n", (1 ^ 1) );
    printf("255 << 2 daje %d \n", (255 << 2) );
    printf("255 >> 2 daje %d \n", (255 >> 2) );
    printf("~255     daje %ld \n", ~255 );
    return 0;
}
```

Wynik działania programu:

```
1 & 1    daje 1
1 | 1    daje 1
1 ^ 1    daje 0
255 << 2 daje 1020
255 >> 2 daje 63
~255     daje -256

--------------------------------
Process exited after 12.73 seconds with return value 0
Press any key to continue . . .
```

Zwróć uwagę, że operator `&` daje `1` tylko wtedy, gdy odpowiadające im bity mają wartość `1`. To sprawia, że jest to przydatne do "maskowania" wybranych bitów wartości za pomocą wzorca bitowego, który ma zera w pozycjach, które chcesz wyłączyć.

```
  1011100   <-- bajt, z którego niektóre bity chcemy wyłączyć.
& 1101010   <-- wzorzec; zależy nam na wyłączeniu bitów pierwszego, trzeciego
---------       i piątego licząc od prawej strony. O inne bity nie dbamy.
  1001000
```

Operator `|` włącza bit, jeśli jedna lub obie wartości mają `1` w tej pozycji. Jeśli chcesz to zagwarantować — włącz określony bit we wzorcu.

```
  1011100   <-- bajt, w którym niektóre bity chcemy włączyć.
| 1001010   <-- wzorzec; zależy nam na włączeniu bitów drugiego, czwartego
---------       i siódmego licząc od prawej strony. O inne bity nie dbamy.
  1011110
```

Czwarta instrukcja dwukrotnie przesuwa w lewo wartość $255_{(10)}$ ($00000000\ 11111111_{(2)}$ staje się $00000011\ 11111100_{(2)}$). Ponieważ wartość każdego miejsca w liczbie binarnej jest dwa razy większa od miejsca po prawej stronie, jest to równoważne pomnożeniu `255 * 2 * 2` czyli $1020_{(10)}$.

W następnej instrukcji $255_{(10)}$ jest dwukrotnie przesunięte w prawo, więc $00000000\ 11111111_{(2)}$ staje się $00000000\ 00111111_{(2)}$ czyli $63_{(10)}$.

Na koniec ostatnia instrukcja zamienia `00000000 11111111` na `11111111 00000000`. Jest to odpowiednik $65\,535_{(10)}$ `-` $255_{(10)}$ czyli $65\,280_{(10)}$ (`11111111 11111111 - 00000000 11111111 = 11111111 00000000`) ale tylko wtedy, gdy deklarujemy ją jako *`unsigned int`*.

Nie ma znaku 'minus' przy liczbie w komórkach pamięci. Liczbę ujemną system rozpoznaje patrząc na wartość najwyższego bitu (tego po lewej stronie) - jeżeli jest nim `0` to liczba jest dodatnia, jeżeli `1` to ujemna. Bierze się to z metody zwanej "two's complements", co można przetłumaczyć jako "dopełnienie do jedynki". Co to więc za liczba ujemna? Powstaje ona z liczby dodatniej po inwersji bitów i dodanie jednego bitu. W przeciwną stronę będzie to:

```
11111110 11111111       <-- musi być to ...
-               1       <-- ... skoro odjęcie jednego bitu ...
-----------------
11111111 00000000       <-- ... ma nam dać właśnie to.
```

Inwersja bitów liczby $11111110\ 11111111_{(2)}$ da nam: `00000001 00000000` a to jest `256`. Tak więc $11111111\ 00000000_{(2)}$ to $-256_{(10)}$.

W taki sam sposób możesz zamienić dowolną liczbę systemu **dziesiętnego**, np. `-427` na jej reprezentację dodatnią:

```
       427  <-- liczba, której ujemnej reprezentacji szukamy
...9999572  <-- wynik dopełnienia cyfr do 9
+         1  <-- dodajemy 1
----------
...9999573  <-- jest to 'dodatnia reprezentacja liczby ujemnej' (-427).
```

Dodanie tej liczby do np. `500` ma nam dać `73`, tak jak `500 + (-427) = 73`

```
  ...0000500      <-- 500
+ ...9999573      <-- to jest 'dodatnia reprezentacja liczby ujemnej' (-427)
------------
  ...0000073      <-- 73 jak było do przewidzenia.
```

### Operatory logiczne

**&&** Logiczne AND/i

**||** Logiczne OR/lub

### Operator warunkowy ? :

**a ? x : y** Jeżeli wyrażenie 'a' jest prawdą, to rób co każe wyrażenie 'x'; w przeciwnym wypadku rób co każe wyrażenie 'y'

### Operatory przypisania

**=** Przypisanie wartości

***=** Przypisanie wartości po mnożeniu

**/=** Przypisanie wartości po dzieleniu

**%=** Przypisanie wartości reszty po dzieleniu

**+=** Przypisanie wartości po dodaniu

**-=** Przypisanie wartości po odjęciu

#### Bitowe operatory przypisania

**<<=** Przypisanie komórkom wartości bitów po ich przesunięciu w lewo)

**>>=** Przypisanie komórkom wartości bitów po ich przesunięciu w prawo)

**&=** Przypisanie komórkom wartości kolejnych bitów jako wyniku operacji AND na bicie tym i takiej samej pozycji bitu innego bajtu)

**^=** Przypisanie komórkom wartości kolejnych bitów jako wyniku operacji XOR na bicie tym i takiej samej pozycji bitu innego bajtu)

**|=** Przypisanie komórkom wartości kolejnych bitów jako wyniku operacji OR na bicie tym i takiej samej pozycji bitu innego bajtu)

### Przecinek jako operator

**,** Wyznacza kolejność operacji, np. `a, b, c;` od lewej strony do prawej

## Operatory dostępu do wnętrza obiektu

| Operator | Syntaks |
|---|---|
| indeks/subscript | a[b] |
| Pośredniego dostępu (poprzez *pointer*) | *a |
| adres obiektu | &a |
| bezpośredniego dostępu do elementu 'b' obiektu 'a' | a.b |
| pośredniego dostępu do elementu 'b' obiektu 'a' (poprzez *pointer*) | a->b |
| dostępu do *pointer*a elementu 'b' obiektu 'a' | a.*b |
| dostępu do *pointer*a elementu 'b' obiektu 'a' poprzez *pointer* | a->*b |

## Typ danych, ich rozmiar i zakres w Turbo C++

| Typ | Rozmiar (bity) | Zakres | Przykłady zastosowania |
|---|---|---|---|

| | | | |
|---|---|---|---|
| unsigned char | 8 | 0 ÷ 255 | Małe liczby i pełny zakres znaków |
| char | 8 | —128 ÷ 127 | Bardzo małe liczby i znaki ASCII |
| enum | 16 | —32,768 to 32,767 | Uporządkowany zbiór wartości |
| unsigned int | 16 | 0 ÷ 65.535 | Większe liczby pętli |
| short int | 16 | —32.768 ÷ 32.767 | Liczniki, małe liczby, kontrola pętli |
| int | 16 | —32.768 ÷ 32.767 | Liczniki, małe liczby, kontrola pętli |
| unsigned long | 32 | 0 ÷ 4.294.967.295 | Odległości astronomiczne |
| long | 32 | —2.147.483.648 ÷ 2.147.483.647 | Duże liczby, populacja |
| float | 32 | $3{,}4 \times 10^{-38} \div 3{,}4 \times 10^{38}$ | Notacja naukowa (precyzja 7-cyfrowa) |
| double | 64 | $1{,}7 \times 10^{-308} \div 1{,}7 \times 10^{308}$ | Notacja naukowa (precyzja 15-cyfrowa) |
| long double | 80 | $3{,}4 \times 10^{-4932} \div 1{,}1 \times 10^{4932}$ | Finanse (precyzja 19-cyfrowa) |
| near *pointer* | 16 | Nie dotyczy | Operowanie adresami pamięci |
| far *pointer* | 32 | Nie dotyczy | Operowanie adresami poza aktualny segment |

## Znaki specjalne (escape characters)

| Postać | Wartość | Znak | Co się wykonuje |
|---|---|---|---|
| \a | 0x07 | BEL | Słyszalny sygnał |
| \b | 0x08 | BS | Cofnięcie kursora ze zmazaniem znaku |
| \f | 0x0C | FF | Przejście do następnej strony |
| \n | 0x0A | LF | Przeskok kursora na początek następnej linii |
| \r | 0x0D | CR | Powrót kursora na początek tej linii w której aktualnie jest |
| \t | 0x09 | HT | Tabulacja pozioma |
| \v | 0x0B | VT | Tabulacja pionowa |
| \\ | 0x5c | \ | Pokazuje ukośnik odwrotny |
| \' | 0x27 | ' | Pokazuje pojedynczy cudzysłów (apostrof) |
| \" | 0x22 | " | Pokazuje cudzysłów |
| \? | 0x3F | ? | Pokazuje pytajnik |
| \OOO | | dowolny Łańcuch znaków składający się z maksymalnie trzech cyfr systemu ósemkowego | |
| \xHHH | | dowolny Łańcuch znaków cyfr szesnastkowych, a ÷ f małymi literami | |
| \XHHH | | dowolny Łańcuch znaków cyfr szesnastkowych, A ÷ F wielkimi literami | |

## Wszystkie słowa kluczowe Turbo C++

| | | | |
|---|---|---|---|
| **_asm** | **_ds** | **int** | **_seg** |
| **asm** | **else** | **_interrupt** | **short** |
| **auto** | **enum** | **interrupt** | **signed** |
| **break** | **_es** | **_loadds** | **sizeof** |
| **case** | **extern** | **long** | **_ss** |
| **_cdecl** | **_far** | **_near** | **static** |
| **cdecl** | **far** | **near** | **struct** |
| **char** | **_fastcall** | **new** | **switch** |
| **class** | **float** | **operator** | **template** |
| **const** | **for** | **_pascal** | **this** |

| | | | |
|---|---|---|---|
| **continue** | **friend** | **pascal** | **typedef** |
| **_cs** | **goto** | **private** | **union** |
| **default** | **_huge** | **protected** | **unsigned** |
| **delete** | **huge** | **public** | **virtual** |
| **do** | **if** | **register** | **void** |
| **doubie** | **inline** | **return** | **volatile** |
| | | **_saveregs** | **while** |

### Rozszerzenie Turbo C++ dla C

| | | | |
|---|---|---|---|
| **_edecl** | **_es** | **interrupt** | **pascal** |
| **cdecl** | **_far** | **_loadds** | **_saveregs** |
| **_cs** | **far** | **_hear** | **_seg** |
| **_ds** | **_fastcall** | **near** | **_ss** |
| | **huge** | **_pascal** | |

### Słowa kluczowe charakterystyczne dla C++

| | |
|---|---|
| **asm** | **operator** |
| **class** | **private** |
| **delete** | **protected** |
| **friend** | **public** |
| **inline** | **template** |
| **new** | **this** |
| | **virtual** |

## Typy podstawowe

**long**, **signed** i **unsigned** są modyfikatorami, które można zastosować do typów całkowitych**.**
Specyfikacja podstawowych typów jest tworzona z następujących słów kluczowych:

| | | |
|---|---|---|
| **char** | **int** | **signed** |
| **double** | **long** | **unsigned** |
| **float** | **short** | |

## Typy całkowite

| | |
|---|---|
| **char, signed char** | Synonimy jeżeli domyślny **char** jest ustawiony na **signed** |
| **unsigned char** | |
| **char, unsigned char** | Synonimy jeżeli domyślny **char** jest ustawiony na **unsigned** |
| **signed char** | |
| **int, signed int** | |
| **unsigned, unsigned int** | |
| **short, short int, signed short int** | |

**unsigned short, unsigned short int**
**long, long int, signed long int**
**unsigned long, unsigned long int**

# Lista specyfikatorów formatu w C

czyli: Jaki chcemy przedstawić określoną wartość na wydruku danych (głównie w funkcji *printf*).

Poniższa tabela zawiera najczęściej używane specyfikatory formatu w języku C

| Specyfikator formatu | Opis |
|---|---|
| %c | Pojedynczy znak |
| %d | Liczba całkowita ze znakiem |
| %f | Liczba zmiennoprzecinkowa w zapisie dziesiętnym |
| %e lub %E | Liczba w zapisie wykładniczym liczby zmiennoprzecinkowej |
| %g lub %G | Liczba zmiennoprzecinkowa (%g lub %f) którykolwiek zapis jest krótszy |
| %i | Liczba całkowita bez znaku |
| %ld lub %li | Długa liczba całkowita |
| %lf | Długa liczba zmiennoprzecinkowa; liczba podwójnej precyzji (*double*) |
| %lu | Liczba całkowita bez znaku lub długa (*long*) liczba całkowita |
| %lli lub %lld | Bardzo długa (*long long*) liczba całkowita |
| %llu | Bardzo długa (*long long*) liczba całkowita bez znaku |
| %o | Liczba w zapisie ósemkowym bez znaku |
| %p | Wskaźnik |
| %s | Łańcuch znakowy (*string*) |
| %u | Liczba całkowita bez znaku |
| %x lub %X | Liczba w zapisie szesnastkowym bez znaku |
| %n | Nic nie drukuje |
| %% | Drukuje znak % |

l (mała litera L) - prefiks używany z %d, %u, %x, %o aby wskazać, że jest to długa (*long*) liczba całkowita.

# Tryb otwierania pliku dla fopen()

| tryb | opis działania trybu |
|---|---|
| r | Otwiera istniejący plik tekstowy dla jego odczytu. |
| w | Otwiera plik tekstowy do wpisywania a istniejący tekst ginie. Jeżeli plik nie istnieje, nowy plik jest tworzony. |
| a | Otwiera plik tekstowy do dopisywania a więc stary tekst nie ginie. Jeżeli plik nie istnieje, nowy plik jest tworzony. |
| r+ | Otwiera plik tekstowy zarówno do odczytu jak i wpisywania. Nie niszczy starego tekstu. |
| w+ | Otwiera plik tekstowy zarówno do odczytu jak i wpisywania. Niszczy stary tekst. Jeżeli plik nie istnieje, nowy plik jest tworzony. |
| a+ | Otwiera plik tekstowy zarówno do odczytu jak i wpisywania. Dopisuje do istniejącego tekstu a więc on nie ginie. Jeżeli plik nie istnieje, nowy plik jest tworzony. |

Jeżeli używamy pliku binarnego to do symbolu trybu trzeba dopisać **b**

"rb", "wb", "ab", "rb+", "r+b", "wb+", "w+b", "ab+", "a+b"

# Lista błędów i ich numery

Poniższa tabela zawiera listę numerów błędów (*errno*) i odpowiadających numerowi opisów błędu.

errno = 0  oznacza brak błędu.

Numery błędów jak i ich opisy mogą się różnić w zależności od źródła języka "C".

| Numer | Nazwa błędu | Opis błędu jaki może się pojawić zarówno w `perror` jak i w `strerror( errno )` | Opis błędu po polsku |
|---|---|---|---|
| 0 | EZERO | **No error** | Brak błędu |
| 1 | EPERM | **Operation not permitted** | Niedozwolona operacja |
| 2 | ENOENT | **No such file or directory** | Nie ma takiego pliku ani katalogu |
| 3 | ESRCH | **No such process** | Nie ma takiego procesu |
| 4 | EINTR | **Interrupted system call** | Przerwane wywołanie systemowe |
| 5 | EIO | **Input/output error** | Błąd wejścia/wyjścia |
| 6 | ENXIO | **No such device or address** | Nie ma takiego urządzenia ani adresu |
| 7 | E2BIG | **Argument list too long** | Lista argumentów za długa |
| 8 | ENOEXEC | **Exec format error** | Błędny format pliku wykonywalnego |
| 9 | EBADF | **Bad file descriptor** | Błędne oznaczenie pliku |
| 10 | ECHILD | **No child processes** | Brak procesów podrzędnych |
| 11 | EAGAIN | **Resource temporarily unavailable** | Zasoby chwilowo niedostępne |
| 12 | ENOMEM | **Cannot allocate memory** | Za mało miejsca w pamięci operacyjnej |
| 13 | EACCES | **Permission denied** | Brak dostępu |
| 14 | EFAULT | **Bad address** | Błędny adres |
| 16 | EBUSY | **Device or resource busy** | Urządzenie lub zasoby zajęte |
| 17 | EEXIST | **File exists** | Plik już istnieje |
| 18 | EXDEV | **Invalid cross-device link** | Nieprawidłowy link między urządzeniami |
| 19 | ENODEV | **No such device** | Nie ma takiego urządzenia |
| 20 | ENOTDIR | **Not a directory** | To nie katalog |
| 21 | EISDIR | **Is a directory** | To jest katalog |
| 22 | EINVAL | **Invalid argument** | Błędny argument |
| 23 | ENFILE | **Too many files open in system** | Zbyt wiele plików jest otwartych w systemie |
| 24 | EMFILE | **Too many open files** | Za dużo otwartych plików w procesie |
| 25 | ENOTTY | **Inappropriate ioctl for device** | Niewłaściwa operacja kontrolna wejścia/wyjścia |
| 27 | EFBIG | **File too large** | Plik jest za duży |
| 28 | ENOSPC | **No space left on device** | Brak miejsca na urządzeniu |
| 29 | ESPIPE | **Invalid seek** | Nieprawidłowe wyszukiwanie |
| 30 | EROFS | **Read-only file system** | System plików wyłącznie do odczytu |
| 31 | EMLINK | **Too many links** | Za dużo linków |
| 32 | EPIPE | **Broken pipe** | Przerwany potok |
| 33 | EDOM | **Numerical argument out of domain** | Argument funkcji matematycznej nie należy do dziedziny funkcji |
| 34 | ERANGE | **Numerical result out of range** | Wynik jest zbyt duży. |
| 36 | EDEADLK | **Resource deadlock avoided** | Wystąpło zakleszczenie (deadlock) zasobów |
| 38 | ENAMETOOLONG | **File name too long** | Zbyt długa nazwa pliku |
| 39 | ENOLCK | **No locks available** | Brak dostępnych blokad |
| 40 | ENOSYS | **Function not implemented** | Funkcja nie jest zaimplementowana |
| 41 | ENOTEMPTY | **Directory not empty** | Katalog nie jest pusty |
| 42 | EILSEQ | **Invalid or incomplete multibyte or wide character** | Niedozwolona sekwencja bajtów |
| 80 | STRUNCATE | **A string copy or concatenation resulted in a truncated string** | Kopiowanie lub łączenie łańcuchów znakowych spowodowało obcięcie łańcucha znakowego |
| 100 | EADDRINUSE | **Address already in use** | Adres jest już w użyciu |
| 101 | EADDRNOTAVAIL | **Cannot assign requested address** | Adres jest  niedostępny |
| 102 | EAFNOSUPPORT | **Address family not supported by protocol** | Rodzina adresów nie jest obsługiwana |
| 103 | EALREADY | **Operation already in progress** | Operacja jest już wykonywana |
| 105 | ECANCELED | **Operation canceled** | Operacja jest anulowana |
| 106 | ECONNABORTED | **Software caused connection abort** | Połączenie zostało przerwane |
| 107 | ECONNREFUSED | **Connection refused** | Połączenie zostało odrzucone |
| 108 | ECONNRESET | **Connection reset by peer** | Połączenie zerwane przez inną osobę |
| 109 | EDESTADDRREQ | **Destination address required** | Wymagany jest adres docelowy |
| 110 | EHOSTUNREACH | **No route to host** | Nie ma drogi do serwera |
| 112 | EINPROGRESS | **Operation now in progress** | Operacja jest teraz właśnie wykonywana |
| 113 | EISCONN | **Transport endpoint is already connected** | Gniazdo już jest połączone |
| 114 | ELOOP | **Too many levels of symbolic links** | Zbyt wiele poziomów symbolicznych linków |
| 115 | EMSGSIZE | **Message too long** | Komunikat jest za długi |
| 116 | ENETDOWN | **Network is down** | Sieć nie działa (jest wyłączona) |

| | | | |
|---|---|---|---|
| 117 | ENETRESET | **Network dropped connection on reset** | Resetowanie sieci |
| 118 | ENETUNREACH | **Network is unreachable** | Sieć jest niedostępna |
| 119 | ENOBUFS | **No buffer space available** | Brak miejsca w buforze |
| 123 | ENOPROTOOPT | **Protocol not available** | Protokół jest niedostępny |
| 126 | ENOTCONN | **Transport endpoint is not connected** | Gniazdo nie jest podłączone |
| 128 | ENOTSOCK | **Socket operation on non-socket** | Operacja gniazda na nie-gnieździe |
| 129 | ENOTSUP | **Operation not supported** | Operacja nie jest obsługiwana |
| 130 | EOPNOTSUPP | **Operation not supported** | Operacja nie jest obsługiwana |
| 132 | EOVERFLOW | **Value too large for defined data type** | Wartość zbyt duża dla zdefiniowanego typu danych |
| 133 | EOWNERDEAD | **Owner died** | Właściciel nie żyje |
| 134 | EPROTO | **Protocol error** | Błąd protokołu |
| 135 | EPROTONOSUPPORT | **Protocol not supported** | Protokół nie jest obsługiwany |
| 136 | EPROTOTYPE | **Protocol wrong type for socket** | Nieprawidłowy typ protokołu. |
| 138 | ETIMEDOUT | **Connection timed out** | Przekroczony czas oczekiwania na połączenie |
| 140 | EWOULDBLOCK | **Resource temporarily unavailable** | Operacja zostanie zablokowana |