MergeSort z użyciem Google Test

Wygenerowano za pomocą Doxygen 1.12.0

Rozdział 1

Indeks klas

1.1 Lista klas

Tutaj znajdują się klasy, struktury, unie i interfejsy wraz z ich krótkimi opisami:

Rozdział 2

Indeks plików

2.1 Lista plików

Tutaj znajduje się lista wszystkich udokumentowanych plików wraz z ich krótkimi opisami:

Rozdział 3

Dokumentacja klas

3.1 Dokumentacja klasy MergeSort

Klasa MergeSort.

`#include <MergeSort.h>`

Metody publiczne

- MergeSort ()
  Konstruktor domyślny klasy MergeSort.
- MergeSort (int input[ ], int size)
  Konstruktor używany do tworzenia obiektu tablicy o danym rozmiarze.
- ~MergeSort ()
  Domyślny destruktor.
- void printArray ()
  Funkcja printująca całą tablicę na konsolę.
- void setArray (int input[ ], int inputSize)
  Funkcja inicjalizująca tablicę.
- void sort ()
  Funkcja inicjująca proces sortowania.
- int * getArray ()
  Getter - Funkcja zwraca wskaźnik MergeSort::arr.

Metody prywatne

- void mergeSort (int *arr, int left, int right)
  Funkcja dzieląca tablice na mniejsze tablice.
- void merge (int *arr, int left, int mid, int right)
  Funkcja sortująca i scalająca tablice.

Atrybuty prywatne

- int * arr
- int size

3.1.1 Opis szczegółowy

Klasa MergeSort.

Klasa ta reprezentuje algorytm sortowania przez scalanie. Zawiera metody będące implementacją algorytmu sortującego.

3.1.2 Dokumentacja konstruktora i destruktora

3.1.2.1 MergeSort() [1/2]

```
MergeSort::MergeSort ()
```

Konstruktor domyślny klasy MergeSort.

Ustawia MergeSort::size na `0` oraz tworzy pustą tablicę

3.1.2.2 MergeSort() [2/2]

```
MergeSort::MergeSort (
            int input[],
            int size)
```

Konstruktor używany do tworzenia obiektu tablicy o danym rozmiarze.

Parametry

input[ ]	Tablica przekazana do konstruktora, która będzie posortowana.
size	wielkość przekazanej tablicy

Ustawia MergeSort::size na wartość przekazaną przez parametr `size` oraz alokuje pamięć dla tablicy MergeSort::arr o rozmiarze `size`

Inicjalizuje tablicę funkcją MergeSort::setArray()

Sortuje tablicę wywołując funkcję MergeSort::sort()

3.1.2.3 ~MergeSort()

```
MergeSort::~MergeSort ()
```

Domyślny destruktor.

Destruktor wywołuje `delete` na tablicy, zwalniając pamięć.

3.1.3 Dokumentacja funkcji składowych

3.1.3.1 getArray()

```
int * MergeSort::getArray ()
```

Getter - Funkcja zwraca wskaźnik MergeSort::arr.

Zwrócenie wskaźnika `*arr`

3.1.3.2 merge()

```
void MergeSort::merge (
            int * arr,
            int left,
            int mid,
            int right)  [private]
```

Funkcja sortująca i scalająca tablice.

Parametry

**arr*	tablica która będzie scalana
left	Lewy indeks tablicy
mid	Środkowy indeks tablicy
right	Prawy indeks tablicy

Wyznaczane są wielkości `n1` oraz `n2` dla nowych tablic `leftArr` oraz `rightArr` oraz alokacja pamięci.

Dane są kopiowane do nowych tablic `leftArr` oraz `rightArr`.

Następnie są tablice sortowane przez porównanie oraz scalane.

Jeżeli pozostały jakiekolwiek elementy w tablicach, są one także kopiowane.

Na samym końcu następuje zwolnienie pamięci dla dwóch tymczasowych tablic.

3.1.3.3 mergeSort()

```
void MergeSort::mergeSort (
            int * arr,
            int left,
            int right)  [private]
```

Funkcja dzieląca tablice na mniejsze tablice.

Parametry

**arr*	tablica która będzie dalej dzielona na mniejsze części
left	Lewy indeks tablicy
right	Prawy indeks tablicy

Wywoływana rekurencją. Wyznacza środek tablicy i przekazuje jako parametr granice do podziału na mniejsze tablice.

Gdy tablice będą wystarczająco małe, następuje wywołanie MergeSort::merge() aby posortować i scalić tablice.

3.1.3.4 printArray()

```
void MergeSort::printArray ()
```

Funkcja printująca całą tablicę na konsolę.

Printowanie na ekran poszczególnych elementów tablicy za pomocą pętli `for` oraz `std::cout`

3.1.3.5 setArray()

```
void MergeSort::setArray (
            int input[],
            int inputSize)
```

Funkcja inicjalizująca tablicę.

Parametry

input[]	tablica która będzie inicjalizowana.
inputSize	rozmiar tablicy do inicjalizacji.

Funkcja inicjalizuje tablicę, wypełniając ją indeksami pętli `for`

3.1.3.6 sort()

```
void MergeSort::sort ()
```

Funkcja inicjująca proces sortowania.

Inicjalizuje proces sortowania tablicy wywołując rekurencyjnie MergeSort::mergeSort.

Funkcja zakończy się gdy `size` będzie `<= 1`

3.1.4 Dokumentacja atrybutów składowych

3.1.4.1 arr

```
int* MergeSort::arr  [private]
```

Wskaźnik na dynamicznie alokowaną tablicę

3.1.4.2 size

```
int MergeSort::size  [private]
```

Przechowuje wielkość tablicy

Dokumentacja dla tej klasy została wygenerowana z plików:

- MergeSort/MergeSort.h
- MergeSort/MergeSort.cpp

Rozdział 4

Dokumentacja plików

4.1 Dokumentacja pliku GTest/test.cpp

```
#include "pch.h"
#include "../MergeSort/MergeSort.h"
#include "../MergeSort/MergeSort.cpp"
#include <algorithm>
#include <cstdlib>
```

Funkcje

- TEST (Sortowanie, TablicaNiezmieniona)

  Test który sprawdza, czy funkcja zachowuje tablicę niezmienioną, gdy ona jest już posortowana rosnąco.
- TEST (Sortowanie, TablicaNiezmienionaFail)

  Test który sprawdza, czy test poprawnie zrobi fail, gdy ręcznie ustawimy inną wartość.
- TEST (Sortowanie, SortowanieTablicyOdwrotnejKolejnosci)

  Test który sprawdza, czy funkcja poprawnie przesortuje tablicę podaną w odwrotnej kolejności.
- TEST (Sortowanie, SortowanieTablicyUjemnychWartosci)

  Test który sprawdza, czy funkcja poprawnie przesortuje tablicę ujemnych wartości.
- TEST (Sortowanie, SortowanieTablicyUjemnychIDodatnichWartosci)

  Test który sprawdza, czy funkcja poprawnie przesortuje tablicę ujemnych i dodatnich wartości.
- TEST (Sortowanie, SortowanieTablicyZDuplikatami)

  Test który sprawdza, czy funkcja poprawnie przesortuje tablicę dodatnich wartości z duplikatami.
- TEST (Sortowanie, SortowanieTablicyZUjemnymiDuplikatami)

  Test który sprawdza, czy funkcja poprawnie przesortuje tablicę ujemnych wartości z duplikatami.
- TEST (Sortowanie, SortowanieLosowejTablicyLiczb)

  Test który sprawdza, czy funkcja poprawnie przesortuje tablicę dodatnich losowych wartości.
- TEST (Inicjalizacja, NieRzucaWyjatkiem)

  Test który sprawdza, czy stworzenie pustej tablicy nie rzuci wyjątku.
- TEST (Sortowanie, SortowanieTablicyUjemnychDodatnichIDuplikatow)

  Test który sprawdza, czy funkcja poprawnie przesortuje tablicę wartości dodatnich, ujemnych z duplikatami.
- TEST (Sortowanie, SortowanieTablicyDwaElementyRosnaco)

  Test który sprawdza, czy funkcja poprawnie przesortuje tablicę dwuelementową rosnącą.
- TEST (Sortowanie, SortowanieDuzejTablicyPonad100Elementow)

  Test który sprawdza, czy funkcja poprawnie przesortuje dużą tablicę dodatnich losowych wartości.

- TEST (Sortowanie, SortowanieDuzejTablicyPonad100ElementowUjemnychDodatnichDuplikatow)

  Test który sprawdza, czy funkcja poprawnie przesortuje dużą tablicę losowych wartości dodatnich, ujemnych i duplikatów.
- TEST (Sortowanie, TablicaJednoelementowa)

  Test który sprawdza, czy funkcja poprawnie przesortuje tablicę jednoelementową.

4.1.1 Dokumentacja funkcji

4.1.1.1 TEST() [1/14]

```
TEST (
            Inicjalizacja ,
            NieRzucaWyjatkiem )
```

Test który sprawdza, czy stworzenie pustej tablicy nie rzuci wyjątku.

Tablica `input` zawiera już posortowane wartości. Są to dane wejściowe.

Zmienna `size` przechowuje wielkość tablicy.

Do sprawdzenia rzucenia wyjątku, użyto makra `ASSERT_NO_THROW`

Parametry

Sortowanie	
TablicaNiezmieniona	

4.1.1.2 TEST() [2/14]

```
TEST (
            Sortowanie ,
            SortowanieDuzejTablicyPonad100Elementow )
```

Test który sprawdza, czy funkcja poprawnie przesortuje dużą tablicę dodatnich losowych wartości.

Tablica `input` zawiera już posortowane wartości. Są to dane wejściowe.

Zmienna `size` przechowuje wielkość tablicy.

Aby wygenerować losowe wartości, użyto funkcji `rand()`

MergeSort tworzy obiekt z tablicą `input` oraz obiektem `size`.

Poprzez wywołanie MergeSort MergeSort::getArray(), wskaźnik do MergeSort::arr jest przekazany do zmiennej `tempArray`.

Wywołanie `std::sort` na tablicy, aby przesortować podaną tablicę z biblioteki `cstdlib`

Pętla `for` porównuje makrem `EXPECT_EQ` wartości tablicy `inputArray` z tablicą `tempArray`.

Parametry

Sortowanie	
TablicaNiezmieniona	

4.1.1.3 TEST() [3/14]

```
TEST (
            Sortowanie ,
            SortowanieDuzejTablicyPonad100ElementowUjemnychDodatnichDuplikatow )
```

Test który sprawdza, czy funkcja poprawnie przesortuje dużą tablicę losowych wartości dodatnich, ujemnych i duplikatów.

Tablica `input` zawiera już posortowane wartości. Są to dane wejściowe.

Zmienna `size` przechowuje wielkość tablicy.

Aby wygenerować losowe wartości, użyto funkcji `rand()`

MergeSort tworzy obiekt z tablicą `input` oraz obiektem `size`.

Poprzez wywołanie MergeSort MergeSort::getArray(), wskaźnik do MergeSort::arr jest przekazany do zmiennej `tempArray`.

Wywołanie `std::sort` na tablicy, aby przesortować podaną tablicę z biblioteki `cstdlib`

Pętla `for` porównuje makrem `EXPECT_EQ` wartości tablicy `inputArray` z tablicą `tempArray`.

Parametry

Sortowanie	
TablicaNiezmieniona	

4.1.1.4 TEST() [4/14]

```
TEST (
            Sortowanie ,
            SortowanieLosowejTablicyLiczb )
```

Test który sprawdza, czy funkcja poprawnie przesortuje tablicę dodatnich losowych wartości.

Tablica `input` zawiera już posortowane wartości. Są to dane wejściowe.

Zmienna `size` przechowuje wielkość tablicy.

Aby wygenerować losowe wartości, użyto funkcji `rand()`

MergeSort tworzy obiekt z tablicą `input` oraz obiektem `size`.

Poprzez wywołanie MergeSort MergeSort::getArray(), wskaźnik do MergeSort::arr jest przekazany do zmiennej `tempArray`.

Wywołanie `std::sort` na tablicy, aby przesortować podaną tablicę z biblioteki `cstdlib`

Pętla `for` porównuje makrem `EXPECT_EQ` wartości tablicy `inputArray` z tablicą `tempArray`.

Parametry

Sortowanie	
TablicaNiezmieniona	

4.1.1.5 TEST() [5/14]

```
TEST (
            Sortowanie ,
            SortowanieTablicyDwaElementyRosnaco )
```

Test który sprawdza, czy funkcja poprawnie przesortuje tablicę dwuelementową rosnącą.

Tablica `input` zawiera już posortowane wartości. Są to dane wejściowe.

Zmienna `size` przechowuje wielkość tablicy.

MergeSort tworzy obiekt z tablicą `input` oraz obiektem `size`.

Poprzez wywołanie MergeSort MergeSort::getArray(), wskaźnik do MergeSort::arr jest przekazany do zmiennej `tempArray`.

Wywołanie `std::sort` na tablicy, aby przesortować podaną tablicę z biblioteki `cstdlib`

Pętla `for` porównuje makrem `EXPECT_EQ` wartości tablicy `inputArray` z tablicą `tempArray`.

Parametry

Sortowanie	
TablicaNiezmieniona	

4.1.1.6 TEST() [6/14]

```
TEST (
            Sortowanie ,
            SortowanieTablicyOdwrotnejKolejnosci )
```

Test który sprawdza, czy funkcja poprawnie przesortuje tablicę podaną w odwrotnej kolejności.

Tablica `input` zawiera już posortowane wartości. Są to dane wejściowe.

Zmienna `size` przechowuje wielkość tablicy.

MergeSort tworzy obiekt z tablicą `input` oraz obiektem `size`.

Poprzez wywołanie MergeSort MergeSort::getArray(), wskaźnik do MergeSort::arr jest przekazany do zmiennej `tempArray`.

Wywołanie `std::sort` na tablicy, aby przesortować podaną tablicę z biblioteki `cstdlib`

Pętla `for` porównuje makrem `EXPECT_EQ` wartości tablicy `inputArray` z tablicą `tempArray`.

Parametry

Sortowanie	
TablicaNiezmieniona	

4.1.1.7 TEST() [7/14]

```
TEST (
            Sortowanie ,
            SortowanieTablicyUjemnychDodatnichIDuplikatow )
```

Test który sprawdza, czy funkcja poprawnie przesortuje tablicę wartości dodatnich, ujemnych z duplikatami.

Tablica `input` zawiera już posortowane wartości. Są to dane wejściowe.

Zmienna `size` przechowuje wielkość tablicy.

MergeSort tworzy obiekt z tablicą `input` oraz obiektem `size`.

Poprzez wywołanie MergeSort MergeSort::getArray(), wskaźnik do MergeSort::arr jest przekazany do zmiennej `tempArray`.

Wywołanie `std::sort` na tablicy, aby przesortować podaną tablicę z biblioteki `cstdlib`

Pętla `for` porównuje makrem `EXPECT_EQ` wartości tablicy `inputArray` z tablicą `tempArray`.

Parametry

Sortowanie	
TablicaNiezmieniona	

4.1.1.8 TEST() [8/14]

```
TEST (
            Sortowanie ,
            SortowanieTablicyUjemnychIDodatnichWartosci )
```

Test który sprawdza, czy funkcja poprawnie przesortuje tablicę ujemnych i dodatnich wartości.

Tablica `input` zawiera już posortowane wartości. Są to dane wejściowe.

Zmienna `size` przechowuje wielkość tablicy.

MergeSort tworzy obiekt z tablicą `input` oraz obiektem `size`.

Poprzez wywołanie MergeSort MergeSort::getArray(), wskaźnik do MergeSort::arr jest przekazany do zmiennej `tempArray`.

Wywołanie `std::sort` na tablicy, aby przesortować podaną tablicę z biblioteki `cstdlib`

Pętla `for` porównuje makrem `EXPECT_EQ` wartości tablicy `inputArray` z tablicą `tempArray`.

Parametry

Sortowanie	
TablicaNiezmieniona	

4.1.1.9 TEST() [9/14]

```
TEST (
            Sortowanie ,
            SortowanieTablicyUjemnychWartosci )
```

Test który sprawdza, czy funkcja poprawnie przesortuje tablicę ujemnych wartości.

Tablica `input` zawiera już posortowane wartości. Są to dane wejściowe.

Zmienna `size` przechowuje wielkość tablicy.

MergeSort tworzy obiekt z tablicą `input` oraz obiektem `size`.

Poprzez wywołanie MergeSort MergeSort::getArray(), wskaźnik do MergeSort::arr jest przekazany do zmiennej `tempArray`.

Wywołanie `std::sort` na tablicy, aby przesortować podaną tablicę z biblioteki `cstdlib`

Pętla `for` porównuje makrem `EXPECT_EQ` wartości tablicy `inputArray` z tablicą `tempArray`.

Parametry

Sortowanie	
TablicaNiezmieniona	

4.1.1.10 TEST() [10/14]

```
TEST (
            Sortowanie ,
            SortowanieTablicyZDuplikatami )
```

Test który sprawdza, czy funkcja poprawnie przesortuje tablicę dodatnich wartości z duplikatami.

Tablica `input` zawiera już posortowane wartości. Są to dane wejściowe.

Zmienna `size` przechowuje wielkość tablicy.

MergeSort tworzy obiekt z tablicą `input` oraz obiektem `size`.

Poprzez wywołanie MergeSort MergeSort::getArray(), wskaźnik do MergeSort::arr jest przekazany do zmiennej `tempArray`.

Wywołanie `std::sort` na tablicy, aby przesortować podaną tablicę z biblioteki `cstdlib`

Pętla `for` porównuje makrem `EXPECT_EQ` wartości tablicy `inputArray` z tablicą `tempArray`.

Parametry

Sortowanie	
TablicaNiezmieniona	

4.1.1.11 TEST() [11/14]

```
TEST (
            Sortowanie ,
            SortowanieTablicyZUjemnymiDuplikatami )
```

Test który sprawdza, czy funkcja poprawnie przesortuje tablicę ujemnych wartości z duplikatami.

Tablica `input` zawiera już posortowane wartości. Są to dane wejściowe.

Zmienna `size` przechowuje wielkość tablicy.

MergeSort tworzy obiekt z tablicą `input` oraz obiektem `size`.

Poprzez wywołanie MergeSort MergeSort::getArray(), wskaźnik do MergeSort::arr jest przekazany do zmiennej `tempArray`.

Wywołanie `std::sort` na tablicy, aby przesortować podaną tablicę z biblioteki `cstdlib`

Pętla `for` porównuje makrem `EXPECT_EQ` wartości tablicy `inputArray` z tablicą `tempArray`.

Parametry

Sortowanie	
TablicaNiezmieniona	

4.1.1.12 TEST() [12/14]

```
TEST (
            Sortowanie ,
            TablicaJednoelementowa )
```

Test który sprawdza, czy funkcja poprawnie przesortuje tablicę jednoelementową.

Tablica `input` zawiera już posortowane wartości. Są to dane wejściowe.

Zmienna `size` przechowuje wielkość tablicy.

MergeSort tworzy obiekt z tablicą `input` oraz obiektem `size`.

Poprzez wywołanie MergeSort MergeSort::getArray(), wskaźnik do MergeSort::arr jest przekazany do zmiennej `tempArray`.

Wywołanie `std::sort` na tablicy, aby przesortować podaną tablicę z biblioteki `cstdlib`

Pętla `for` porównuje makrem `EXPECT_EQ` wartości tablicy `inputArray` z tablicą `tempArray`.

Parametry

Sortowanie	
TablicaNiezmieniona	

4.1.1.13 TEST() [13/14]

```
TEST (
            Sortowanie ,
            TablicaNiezmieniona )
```

Test który sprawdza, czy funkcja zachowuje tablicę niezmienioną, gdy ona jest już posortowana rosnąco.

Tablica `input` zawiera już posortowane wartości. Są to dane wejściowe.

Zmienna `size` przechowuje wielkość tablicy.

MergeSort tworzy obiekt z tablicą `input` oraz obiektem `size`.

Poprzez wywołanie MergeSort MergeSort::getArray(), wskaźnik do MergeSort::arr jest przekazany do zmiennej `tempArray`.

Pętla `for` porównuje makrem `EXPECT_EQ` wartości tablicy `inputArray` z tablicą `tempArray`.

Parametry

Sortowanie	
TablicaNiezmieniona	

4.1.1.14 TEST() [14/14]

```
TEST (
            Sortowanie ,
            TablicaNiezmienionaFail )
```

Test który sprawdza, czy test poprawnie zrobi fail, gdy ręcznie ustawimy inną wartość.

Tablica `input` zawiera już posortowane wartości. Są to dane wejściowe.

Zmienna `size` przechowuje wielkość tablicy.

MergeSort tworzy obiekt z tablicą `input` oraz obiektem `size`.

Poprzez wywołanie MergeSort MergeSort::getArray(), wskaźnik do MergeSort::arr jest przekazany do zmiennej `tempArray`.

Ręcznia zmiana wartości indeksu 0: `tempArray[0] = 999;`

Pętla `for` porównuje makrem `EXPECT_EQ` wartości tablicy `inputArray` z tablicą `tempArray`.

Parametry

Sortowanie	
TablicaNiezmieniona	

4.2 Dokumentacja pliku MergeSort/MergeSort.cpp

```
#include <iostream>
#include "MergeSort.h"
```

4.3 MergeSort.cpp

Idź do dokumentacji tego pliku.

```
00001
00005 #include <iostream>
00006 #include "MergeSort.h"
00007
00011 MergeSort::MergeSort()
00012 {
00013     this->size = 0;
00014     this->arr = new int[this->size];
00015 }
00016
00025 MergeSort::MergeSort(int input[], int size)
00026 {
00027     this->size = size;
00028     this->arr = new int[size];
00029
00030     this->setArray(input, this->size);
00031
00032     this->sort();
00033 }
00034
00038 MergeSort::~MergeSort()
00039 {
00040     delete[] arr;
00041 }
00042
00046 void MergeSort::printArray()
00047 {
00048     for (int i = 0; i < this->size; i++) {
00049         std::cout << this->arr[i] << " ";
00050     }
00051     std::cout << std::endl;
00052 }
00053
00057 void MergeSort::setArray(int input[], int inputSize)
00058 {
00059     for (int i = 0; i < inputSize; i++) {
00060         this->arr[i] = input[i];
00061     }
00062 }
00063
00069 void MergeSort::sort()
00070 {
00071     if (size <= 1)
00072         return;
00073
00074     mergeSort(arr, 0, size - 1);
00075 }
00076
00080 int* MergeSort::getArray()
00081 {
00082     return this->arr;
00083 }
00084
00090 void MergeSort::mergeSort(int* arr, int left, int right)
00091 {
00092     if (left < right) {
```

```
00093
00094         int mid = left + (right - left) / 2;
00095
00096         mergeSort(arr, left, mid);
00097         mergeSort(arr, mid + 1, right);
00098
00099         merge(arr, left, mid, right);
00100     }
00101 }
00102
00114 void MergeSort::merge(int* arr, int left, int mid, int right)
00115 {
00116     int n1 = mid - left + 1;
00117     int n2 = right - mid;
00118
00119     int* leftArr = new int[n1];
00120     int* rightArr = new int[n2];
00121
00122
00123     for (int i = 0; i < n1; i++)
00124         leftArr[i] = arr[left + i];
00125
00126     for (int j = 0; j < n2; j++)
00127         rightArr[j] = arr[mid + 1 + j];
00128
00129
00130     int i = 0, j = 0, k = left;
00131     while (i < n1 && j < n2) {
00132         if (leftArr[i] <= rightArr[j]) {
00133             arr[k] = leftArr[i];
00134             i++;
00135         }
00136         else {
00137             arr[k] = rightArr[j];
00138             j++;
00139         }
00140         k++;
00141     }
00142
00143     while (i < n1) {
00144         arr[k] = leftArr[i];
00145         i++;
00146         k++;
00147     }
00148
00149     while (j < n2) {
00150         arr[k] = rightArr[j];
00151         j++;
00152         k++;
00153     }
00154
00155     delete[] leftArr;
00156     delete[] rightArr;
00157 }
00158
00159 int* MergeSort::getArray()
00160 {
00161     return this->arr;
00162 }
```

4.4 Dokumentacja pliku MergeSort/MergeSort.h

Komponenty

- class MergeSort

  Klasa MergeSort.

4.5 MergeSort.h

Idź do dokumentacji tego pliku.

```
00001
00005 #pragma once
00006
```

```
00014 class MergeSort
00015 {
00016 private:
00017     int* arr;
00019     int size;
00027     void mergeSort(int* arr, int left, int right);
00028
00036     void merge(int* arr, int left, int mid, int right);
00037
00038 public:
00039
00043     MergeSort();
00044
00050     MergeSort(int input[], int size);
00051
00055     ~MergeSort();
00056
00060     void printArray();
00061
00067     void setArray(int input[], int inputSize);
00068
00072     void sort();
00073
00077     int* getArray();
00078 };
00079
```

Skorowidz