warsztat6

Generated by Doxygen 1.9.3

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Data Structure Documentation

3.1 Vector Struct Reference

Data Fields

- void ** **elements**
- size_t **size**
- size_t **capacity**

The documentation for this struct was generated from the following file:

- vector.h

Chapter 4

File Documentation

4.1 tests.c File Reference

Functions

- void traverseFunction (void ∗a)

  Pomocnicza funkcja przekazywana do vectorTraverse.
- void traverseFunction2 (void ∗a)

  Pomocnicza funkcja przekazywana do vectorTraverse.
- int main (void)

  Główna funkcja, w której odbywają się testy.

4.1.1 Function Documentation

4.1.1.1 main()

```
int main (
            void  )
```

Główna funkcja, w której odbywają się testy.

Returns

int

4.1.1.2 traverseFunction()

```
void traverseFunction (
            void * a )
```

Pomocnicza funkcja przekazywana do vectorTraverse.

Parameters

a	

4.1.1.3 traverseFunction2()

```
void traverseFunction2 (
            void * a )
```

Pomocnicza funkcja przekazywana do vectorTraverse.

Parameters

a	

4.2 vector.c File Reference

Functions

- void vectorInit (Vector ∗v)

  Funkcja służąca do inicjalizacji pustego wektora.
- void vectorPushBack (Vector ∗v, void ∗element)

  Funkcja wstawiająca nowy element na koniec wektora.
- void vectorPushFront (Vector ∗v, void ∗element)

  Funkcja wstawiająca nowy element na początek wektora.
- void vectorSet (Vector ∗v, void ∗newValue, size_t index)

  Funkcja służąca do zmieniania wybranego elementu wektora (identyfikowanego za pomocą indeksu).
- void ∗ vectorGet (Vector v, size_t index)

  Funkcja służąca do pobierania wybranego elementu wektora (za pomocą indeksu).
- int vectorEmpty (Vector v)

  Funkcja do sprawdzania, czy dany wektor jest pusty.
- size_t vectorSize (Vector v)

  Funkcja do pobierania rozmiaru wektora.
- size_t vectorCapacity (Vector v)

  Funkcja do pobierania pojemności wektora.
- void vectorTraverse (Vector v, void(∗function)(void ∗))

  Funkcja iterująca po każdym elemencie wektora, i wykonująca na nim funkcję function (wskaźnik do funkcji).
- void vectorReverse (Vector ∗v)

  Funkcja do odwracania kolejności elementów w wektorze (standardowy algorytm).
- Vector vectorClone (Vector ∗v)

  Funkcja służąca do klonowania wektora.
- void ∗∗ vectorToArray (Vector ∗v)

  Funkcja służąca do konwersji wektora na tablicę.
- void vectorPopFront (Vector ∗v)

  Funkcja usuwająca element wektora (z przodu).

- void vectorPopBack (Vector *v)

  Funkcja usuwająca element wektora od końca.
- void vectorClear (Vector *v)

  Funkcja służąca do czyszczenia wektora (usuwanie wszystkich elementów z wektora).
- void vectorDestroy (Vector *v)

  Funkcja do niszczenia wektora.

4.2.1 Function Documentation

4.2.1.1 vectorCapacity()

```
size_t vectorCapacity (
            Vector v )
```

Funkcja do pobierania pojemności wektora.

Parameters

v	

Returns

size_t

4.2.1.2 vectorClear()

```
void vectorClear (
            Vector * v )
```

Funkcja służąca do czyszczenia wektora (usuwanie wszystkich elementów z wektora).

Parameters

v	

4.2.1.3 vectorClone()

```
Vector vectorClone (
            Vector * v )
```

Funkcja służąca do klonowania wektora.

Parameters

v	

Returns

Vector

4.2.1.4 vectorDestroy()

```
void vectorDestroy (
            Vector * v )
```

Funkcja do niszczenia wektora.

Parameters

v	

4.2.1.5 vectorEmpty()

```
int vectorEmpty (
            Vector v )
```

Funkcja do sprawdzania, czy dany wektor jest pusty.

Parameters

v	

Returns

int

4.2.1.6 vectorGet()

```
void * vectorGet (
            Vector v,
            size_t index )
```

Funkcja służąca do pobierania wybranego elementu wektora (za pomocą indeksu).

Parameters

v	
index	

Returns

void∗

4.2.1.7 vectorInit()

```
void vectorInit (
            Vector * v )
```

Funkcja służąca do inicjalizacji pustego wektora.

Parameters

v	

4.2.1.8 vectorPopBack()

```
void vectorPopBack (
            Vector * v )
```

Funkcja usuwająca element wektora od końca.

Parameters

v	

4.2.1.9 vectorPopFront()

```
void vectorPopFront (
            Vector * v )
```

Funkcja usuwająca element wektora (z przodu).

Parameters

v	

4.2.1.10 vectorPushBack()

```
void vectorPushBack (
            Vector * v,
            void * element )
```

Funkcja wstawiająca nowy element na koniec wektora.

Parameters

v	
element	

4.2.1.11 vectorPushFront()

```
void vectorPushFront (
            Vector * v,
            void * element )
```

Funkcja wstawiająca nowy element na początek wektora.

Parameters

v	
element	

4.2.1.12 vectorReverse()

```
void vectorReverse (
            Vector * v )
```

Funkcja do odwracania kolejności elementów w wektorze (standardowy algorytm).

Parameters

v	

4.2.1.13 vectorSet()

```
void vectorSet (
```

```
            Vector * v,
            void * newValue,
            size_t index )
```

Funkcja służąca do zmieniania wybranego elementu wektora (identyfikowanego za pomocą indeksu).

Parameters

v	
newValue	
index	

4.2.1.14 vectorSize()

```
size_t vectorSize (
            Vector v )
```

Funkcja do pobierania rozmiaru wektora.

Parameters

v	

Returns

size_t

4.2.1.15 vectorToArray()

```
void ** vectorToArray (
            Vector * v )
```

Funkcja służąca do konwersji wektora na tablicę.

Parameters

v	

Returns

void∗∗

4.2.1.16 vectorTraverse()

```
void vectorTraverse (
            Vector v,
            void(*)(void *) function )
```

Funkcja iterująca po każdym elemencie wektora, i wykonująca na nim funkcję function (wskaźnik do funkcji).

Parameters

v	
function	

4.3 vector.h

```
1 #ifndef VECTOR_H
2 #define VECTOR_H
3
4 #include <stdio.h>
5
6 #define VECTOR_MAX_CAPACITY 2048*2048
7 #define VECTOR_INITIAL_CAPACITY 5
8
9 typedef struct Vector{
10
11     void** elements;
12     size_t size;
13     size_t capacity;
14
15 } Vector;
16
17
18 void vectorInit(Vector* v);
19 void vectorPushBack(Vector* v, void* element);
20 void vectorPushFront(Vector* v, void* element);
21 void vectorSet(Vector* v, void* newValue, size_t index);
22 void* vectorGet(Vector v, size_t index);
23 int vectorEmpty(Vector v);
24 size_t vectorSize(Vector v);
25 size_t vectorCapacity(Vector v);
26 void vectorTraverse(Vector v, void (*function)(void*));
27 void vectorReverse(Vector* v);
28 Vector vectorClone(Vector* v);
29 void** vectorToArray(Vector* v);
30 void vectorPopFront(Vector* v);
31 void vectorPopBack(Vector* v);
32 void vectorClear(Vector* v);
33 void vectorDestroy(Vector* v);
34
35 #endif
```

Index