OP3

Generated by Doxygen 1.14.0

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

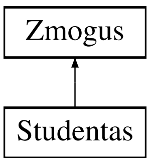
Chapter 4

Class Documentation

4.1 Studentas Class Reference

```
#include <Studentas.h>
```

Inheritance diagram for Studentas:



Public Member Functions

- Studentas ()
- Studentas (const std::string &var, const std::string &pav, const Vector< int > &nd, int egz)
- ∼Studentas ()
- Studentas (const Studentas &other)
- Studentas (Studentas &&other) noexcept
- Studentas & operator= (const Studentas &other)
- Studentas & operator= (Studentas &&other) noexcept
- std::string vardas () const override
- std::string pavarde () const override
- Vector< int > namuDarbai () const
- int egzaminas () const
- void setEgzaminas (int egz)
- double galutinisVidurkis () const
- double galutinisMediana () const
- void skaiciuotiCache () const
- void info () const override

Public Member Functions inherited from Zmogus

- Zmogus ()
- Zmogus (const std::string &var, const std::string &pav)
- virtual ∼Zmogus ()=default

Friends

- std::istream & operator>> (std::istream &in, Studentas &s)
- std::ostream & operator<< (std::ostream &out, const Studentas &s)

Additional Inherited Members

Protected Attributes inherited from Zmogus

- std::string var_
- std::string pav_

4.1.1 Constructor & Destructor Documentation

4.1.1.1 Studentas() [1/4]

```
Studentas::Studentas ()
```

4.1.1.2 Studentas() [2/4]

```
Studentas::Studentas (
              const std::string & var,
              const std::string & pav,
              const Vector< int > & nd,
              int egz)
```

4.1.1.3 ∼Studentas()

```
Studentas::∼Studentas ()
```

4.1.1.4 Studentas() [3/4]

```
Studentas::Studentas (
              const Studentas & other)
```

4.1.1.5 Studentas() [4/4]

```
Studentas::Studentas (
              Studentas && other)   [noexcept]
```

4.1.2 Member Function Documentation

4.1.2.1 egzaminas()

```
int Studentas::egzaminas () const  [inline]
```

4.1.2.2 galutinisMediana()

```
double Studentas::galutinisMediana () const
```

4.1.2.3 galutinisVidurkis()

```
double Studentas::galutinisVidurkis () const
```

4.1.2.4 info()

```
void Studentas::info () const  [inline], [override], [virtual]
```

Implements Zmogus.

4.1.2.5 namuDarbai()

```
Vector< int > Studentas::namuDarbai () const  [inline]
```

4.1.2.6 operator=() [1/2]

```
Studentas & Studentas::operator= (
             const Studentas & other)
```

4.1.2.7 operator=() [2/2]

```
Studentas & Studentas::operator= (
             Studentas && other)  [noexcept]
```

4.1.2.8 pavarde()

```
std::string Studentas::pavarde () const  [inline], [override], [virtual]
```

Reimplemented from Zmogus.

4.1.2.9 setEgzaminas()

```
void Studentas::setEgzaminas (
             int egz)  [inline]
```

4.1.2.10 skaiciuotiCache()

```
void Studentas::skaiciuotiCache () const
```

4.1.2.11 vardas()

```
std::string Studentas::vardas () const  [inline], [override], [virtual]
```

Reimplemented from Zmogus.

4.1.3 Friends And Related Symbol Documentation

4.1.3.1 operator<<

```
std::ostream & operator<< (
             std::ostream & out,
             const Studentas & s)  [friend]
```

4.1.3.2 operator>>

```
std::istream & operator>> (
             std::istream & in,
             Studentas & s)  [friend]
```

The documentation for this class was generated from the following files:

- Studentas.h
- Studentas.cpp

4.2 Vector< T > Class Template Reference

```
#include <Vector.h>
```

Public Types

- using value_type = T
- using size_type = std::size_t
- using reference = T&
- using const_reference = const T&
- using pointer = T∗
- using const_pointer = const T∗
- using iterator = T∗
- using const_iterator = const T∗
- using reverse_iterator = std::reverse_iterator<iterator>
- using const_reverse_iterator = std::reverse_iterator<const_iterator>

Public Member Functions

- Vector ()
- Vector (size_type n, const T &val=T())
- Vector (std::initializer_list< T > il)
- Vector (const Vector &other)
- Vector (Vector &&other) noexcept
- Vector & operator= (const Vector &other)
- Vector & operator= (Vector &&other) noexcept
- ∼Vector ()
- reference operator[ ] (size_type i)
- const_reference operator[ ] (size_type i) const
- reference at (size_type i)
- const_reference at (size_type i) const
- reference front ()
- const_reference front () const
- reference back ()
- const_reference back () const
- pointer data () noexcept
- const_pointer data () const noexcept
- iterator begin () noexcept
- const_iterator begin () const noexcept
- iterator end () noexcept
- const_iterator end () const noexcept
- reverse_iterator rbegin () noexcept
- const_reverse_iterator rbegin () const noexcept
- reverse_iterator rend () noexcept
- const_reverse_iterator rend () const noexcept
- bool empty () const noexcept
- size_type size () const noexcept
- size_type capacity () const noexcept
- void reserve (size_type n)
- void shrink_to_fit ()
- void clear () noexcept
- void push_back (const T &val)
- void push_back (T &&val)
- void pop_back ()
- void resize (size_type n, const T &val=T())
- void swap (Vector &other) noexcept
- iterator insert (const_iterator pos, const T &value)
- iterator erase (const_iterator pos)
- void assign (size_type n, const T &value)
- void assign (std::initializer_list< T > il)
- template<typename InputIt>
  std::enable_if<!std::is_integral< InputIt >::value, void >::type assign (InputIt first, InputIt last)
- template<typename... Args>
  void emplace_back (Args &&... args)
- template<typename InputIt>
  iterator insert (const_iterator pos, InputIt first, InputIt last)
- iterator insert (const_iterator pos, std::initializer_list< T > ilist)
- iterator erase (const_iterator first, const_iterator last)
- template<typename... Args>
  iterator emplace (const_iterator pos, Args &&... args)
- std::allocator< T > get_allocator () const
- bool operator== (const Vector &other) const

- bool operator!= (const Vector &other) const
- bool operator< (const Vector &other) const
- bool operator> (const Vector &other) const
- bool operator<= (const Vector &other) const
- bool operator>= (const Vector &other) const

4.2.1 Member Typedef Documentation

4.2.1.1 const_iterator

```
template<typename T>
using Vector< T >::const_iterator = const T*
```

4.2.1.2 const_pointer

```
template<typename T>
using Vector< T >::const_pointer = const T*
```

4.2.1.3 const_reference

```
template<typename T>
using Vector< T >::const_reference = const T&
```

4.2.1.4 const_reverse_iterator

```
template<typename T>
using Vector< T >::const_reverse_iterator = std::reverse_iterator<const_iterator>
```

4.2.1.5 iterator

```
template<typename T>
using Vector< T >::iterator = T*
```

4.2.1.6 pointer

```
template<typename T>
using Vector< T >::pointer = T*
```

4.2.1.7 reference

```
template<typename T>
using Vector< T >::reference = T&
```

4.2.1.8 reverse_iterator

```
template<typename T>
using Vector< T >::reverse_iterator = std::reverse_iterator<iterator>
```

4.2.1.9 size_type

```
template<typename T>
using Vector< T >::size_type = std::size_t
```

4.2.1.10 value_type

```
template<typename T>
using Vector< T >::value_type = T
```

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Vector() [1/5]

```
template<typename T>
Vector< T >::Vector ()  [inline]
```

4.2.2.2 Vector() [2/5]

```
template<typename T>
Vector< T >::Vector (
             size_type n,
             const T & val = T())  [inline]
```

4.2.2.3 Vector() [3/5]

```
template<typename T>
Vector< T >::Vector (
             std::initializer_list< T > il)  [inline]
```

4.2.2.4 Vector() [4/5]

```
template<typename T>
Vector< T >::Vector (
             const Vector< T > & other)  [inline]
```

4.2.2.5 Vector() [5/5]

```
template<typename T>
Vector< T >::Vector (
             Vector< T > && other)  [inline], [noexcept]
```

4.2.2.6 ∼Vector()

```
template<typename T>
Vector< T >::∼Vector ()  [inline]
```

4.2.3 Member Function Documentation

4.2.3.1 assign() [1/3]

```
template<typename T>
template<typename InputIt>
std::enable_if<!std::is_integral< InputIt >::value, void >::type Vector< T >::assign (
             InputIt first,
             InputIt last)  [inline]
```

4.2.3.2 assign() [2/3]

```
template<typename T>
void Vector< T >::assign (
             size_type n,
             const T & value)  [inline]
```

4.2.3.3 assign() [3/3]

```
template<typename T>
void Vector< T >::assign (
             std::initializer_list< T > il)  [inline]
```

4.2.3.4 at() [1/2]

```
template<typename T>
reference Vector< T >::at (
             size_type i)  [inline]
```

4.2.3.5 at() [2/2]

```
template<typename T>
const_reference Vector< T >::at (
             size_type i) const  [inline]
```

4.2.3.6 back() [1/2]

```
template<typename T>
reference Vector< T >::back ()  [inline]
```

4.2.3.7 back() [2/2]

```
template<typename T>
const_reference Vector< T >::back () const  [inline]
```

4.2.3.8 begin() [1/2]

```
template<typename T>
const_iterator Vector< T >::begin () const  [inline], [noexcept]
```

4.2.3.9 begin() [2/2]

```
template<typename T>
iterator Vector< T >::begin ()  [inline], [noexcept]
```

4.2.3.10 capacity()

```
template<typename T>
size_type Vector< T >::capacity () const  [inline], [noexcept]
```

4.2.3.11 clear()

```
template<typename T>
void Vector< T >::clear ()  [inline], [noexcept]
```

4.2.3.12 data() [1/2]

```
template<typename T>
const_pointer Vector< T >::data () const  [inline], [noexcept]
```

4.2.3.13 data() [2/2]

```
template<typename T>
pointer Vector< T >::data ()  [inline], [noexcept]
```

4.2.3.14 emplace()

```
template<typename T>
template<typename... Args>
iterator Vector< T >::emplace (
             const_iterator pos,
             Args &&...  args)  [inline]
```

4.2.3.15 emplace_back()

```
template<typename T>
template<typename... Args>
void Vector< T >::emplace_back (
            Args &&... args) [inline]
```

4.2.3.16 empty()

```
template<typename T>
bool Vector< T >::empty () const [inline], [noexcept]
```

4.2.3.17 end() [1/2]

```
template<typename T>
const_iterator Vector< T >::end () const [inline], [noexcept]
```

4.2.3.18 end() [2/2]

```
template<typename T>
iterator Vector< T >::end () [inline], [noexcept]
```

4.2.3.19 erase() [1/2]

```
template<typename T>
iterator Vector< T >::erase (
            const_iterator first,
            const_iterator last) [inline]
```

4.2.3.20 erase() [2/2]

```
template<typename T>
iterator Vector< T >::erase (
            const_iterator pos) [inline]
```

4.2.3.21 front() [1/2]

```
template<typename T>
reference Vector< T >::front () [inline]
```

4.2.3.22 front() [2/2]

```
template<typename T>
const_reference Vector< T >::front () const [inline]
```

4.2.3.23 get_allocator()

```
template<typename T>
std::allocator< T > Vector< T >::get_allocator () const  [inline]
```

4.2.3.24 insert() [1/3]

```
template<typename T>
iterator Vector< T >::insert (
             const_iterator pos,
             const T & value)  [inline]
```

4.2.3.25 insert() [2/3]

```
template<typename T>
template<typename InputIt>
iterator Vector< T >::insert (
             const_iterator pos,
             InputIt first,
             InputIt last)  [inline]
```

4.2.3.26 insert() [3/3]

```
template<typename T>
iterator Vector< T >::insert (
             const_iterator pos,
             std::initializer_list< T > ilist)  [inline]
```

4.2.3.27 operator"!=()

```
template<typename T>
bool Vector< T >::operator!= (
             const Vector< T > & other) const  [inline]
```

4.2.3.28 operator<()

```
template<typename T>
bool Vector< T >::operator< (
             const Vector< T > & other) const  [inline]
```

4.2.3.29 operator<=()

```
template<typename T>
bool Vector< T >::operator<= (
             const Vector< T > & other) const  [inline]
```

4.2.3.30 operator=() [1/2]

```
template<typename T>
Vector & Vector< T >::operator= (
             const Vector< T > & other)  [inline]
```

4.2.3.31 operator=() [2/2]

```
template<typename T>
Vector & Vector< T >::operator= (
             Vector< T > && other)  [inline], [noexcept]
```

4.2.3.32 operator==()

```
template<typename T>
bool Vector< T >::operator== (
             const Vector< T > & other) const  [inline]
```

4.2.3.33 operator>()

```
template<typename T>
bool Vector< T >::operator> (
             const Vector< T > & other) const  [inline]
```

4.2.3.34 operator>=()

```
template<typename T>
bool Vector< T >::operator>= (
             const Vector< T > & other) const  [inline]
```

4.2.3.35 operator[]() [1/2]

```
template<typename T>
reference Vector< T >::operator[] (
             size_type i)  [inline]
```

4.2.3.36 operator[]() [2/2]

```
template<typename T>
const_reference Vector< T >::operator[] (
             size_type i) const  [inline]
```

4.2.3.37 pop_back()

```
template<typename T>
void Vector< T >::pop_back ()  [inline]
```

4.2.3.38 push_back() [1/2]

```
template<typename T>
void Vector< T >::push_back (
             const T & val)  [inline]
```

4.2.3.39 push_back() [2/2]

```
template<typename T>
void Vector< T >::push_back (
             T && val)  [inline]
```

4.2.3.40 rbegin() [1/2]

```
template<typename T>
const_reverse_iterator Vector< T >::rbegin () const  [inline], [noexcept]
```

4.2.3.41 rbegin() [2/2]

```
template<typename T>
reverse_iterator Vector< T >::rbegin ()  [inline], [noexcept]
```

4.2.3.42 rend() [1/2]

```
template<typename T>
const_reverse_iterator Vector< T >::rend () const  [inline], [noexcept]
```

4.2.3.43 rend() [2/2]

```
template<typename T>
reverse_iterator Vector< T >::rend ()  [inline], [noexcept]
```

4.2.3.44 reserve()

```
template<typename T>
void Vector< T >::reserve (
             size_type n)  [inline]
```

4.2.3.45 resize()

```
template<typename T>
void Vector< T >::resize (
             size_type n,
             const T & val = T())  [inline]
```

4.2.3.46 shrink_to_fit()

```
template<typename T>
void Vector< T >::shrink_to_fit ()  [inline]
```

4.2.3.47 size()

```
template<typename T>
size_type Vector< T >::size () const  [inline], [noexcept]
```

4.2.3.48 swap()

```
template<typename T>
void Vector< T >::swap (
             Vector< T > & other)  [inline], [noexcept]
```

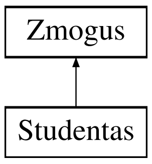
The documentation for this class was generated from the following file:

- Vector.h

4.3 Zmogus Class Reference

```
#include <Zmogus.h>
```

Inheritance diagram for Zmogus:



Public Member Functions

- Zmogus ()
- Zmogus (const std::string &var, const std::string &pav)
- virtual ∼Zmogus ()=default
- virtual std::string vardas () const
- virtual std::string pavarde () const
- virtual void info () const =0

Protected Attributes

- std::string var_
- std::string pav_

4.3.1 Constructor & Destructor Documentation

4.3.1.1 Zmogus() [1/2]

```
Zmogus::Zmogus ()  [inline]
```

4.3.1.2 Zmogus() [2/2]

```
Zmogus::Zmogus (
            const std::string & var,
            const std::string & pav)  [inline]
```

4.3.1.3 ∼Zmogus()

```
virtual Zmogus::∼Zmogus ()  [virtual], [default]
```

4.3.2 Member Function Documentation

4.3.2.1 info()

```
virtual void Zmogus::info () const  [pure virtual]
```

Implemented in Studentas.

4.3.2.2 pavarde()

```
virtual std::string Zmogus::pavarde () const  [inline], [virtual]
```

Reimplemented in Studentas.

4.3.2.3 vardas()

```
virtual std::string Zmogus::vardas () const  [inline], [virtual]
```

Reimplemented in Studentas.

4.3.3 Member Data Documentation

4.3.3.1 pav_

```
std::string Zmogus::pav_  [protected]
```

4.3.3.2 var_

```
std::string Zmogus::var_  [protected]
```

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

- Zmogus.h

Chapter 5

File Documentation

5.1 bench.cpp File Reference

```
#include <iostream>
#include <vector>
#include <chrono>
#include "Vector.h"
#include <iomanip>
```

Functions

- int main ()

5.1.1 Function Documentation

5.1.1.1 main()

```
int main ()
```

5.2 main.cpp File Reference

```
#include "Studentas.h"
#include "Mylib.h"
#include <iostream>
#include "Vector.h"
#include <chrono>
#include <ctime>
```

Functions

- void promptForSortingMethod ()
- int main ()

Variables

- char rikiavimas
- const int MAX_STUDENTU_SKAICIUS = 10000000

5.2.1 Function Documentation

5.2.1.1 main()

```
int main ()
```

5.2.1.2 promptForSortingMethod()

```
void promptForSortingMethod ()
```

5.2.2 Variable Documentation

5.2.2.1 MAX_STUDENTU_SKAICIUS

```
const int MAX_STUDENTU_SKAICIUS = 10000000
```

5.2.2.2 rikiavimas

```
char rikiavimas  [extern]
```

5.3 Mylib.h File Reference

```
#include <iostream>
#include <string>
#include <iomanip>
#include <algorithm>
#include "Vector.h"
#include <limits>
#include <cstdlib>
#include <ctime>
#include <fstream>
#include <sstream>
#include <chrono>
#include <stdexcept>
#include <list>
```

5.4 Mylib.h

Go to the documentation of this file.

```
#ifndef MYLIB_H
#define MYLIB_H

#include <iostream>
#include <string>
#include <iomanip>
#include <algorithm>
#include "Vector.h"
#include <limits>
#include <cstdlib>
#include <ctime>
#include <fstream>
#include <sstream>
#include <chrono>
#include <stdexcept>
#include <list>

using std::string;
using std::cout;
using std::cin;
using std::endl;
using std::fixed;
using std::setprecision;
using std::setw;
using std::left;
using std::vector;
using std::ifstream;
using std::ofstream;
using std::istringstream;
using std::chrono::high_resolution_clock;
using std::chrono::duration_cast;
using std::chrono::duration;
using std::chrono::seconds;
using std::to_string;
using std::exception;

#endif // MYLIB_H
```

5.5 Studentas.cpp File Reference

```
#include "Studentas.h"
#include "Mylib.h"
#include <fstream>
#include <iomanip>
#include <iostream>
#include "Vector.h"
#include <chrono>
#include <ctime>
#include <sstream>
#include <stdexcept>
#include <limits>
#include <algorithm>
#include <direct.h>
```

Functions

- std::istream & operator>> (std::istream &in, Studentas &s)
- std::ostream & operator<< (std::ostream &out, const Studentas &s)
- double Mediana (const Vector< int > &vec)
- void clearInput ()
- string generuotiVarda ()

- string generuotiPavarde ()
- bool skaitymas (Vector< Studentas > &studentai, const string &failoPav)
- void spausdinti (const Vector< Studentas > &studentai, ostream &out)
- void rikiuotiStudentus (Vector< Studentas > &studentai, char rikiavimas)
- void ivestiStudentus (Vector< Studentas > &studentai)
- void sortAndOutputStudents (Vector< Studentas > &studentai)
- void handleFileInput (Vector< Studentas > &studentai)
- void handleOutput (const Vector< Studentas > &studentai)
- void generateStudentFiles ()

Variables

- char rikiavimas

5.5.1 Function Documentation

5.5.1.1 clearInput()

```
void clearInput ()
```

5.5.1.2 generateStudentFiles()

```
void generateStudentFiles ()
```

5.5.1.3 generuotiPavarde()

```
string generuotiPavarde ()
```

5.5.1.4 generuotiVarda()

```
string generuotiVarda ()
```

5.5.1.5 handleFileInput()

```
void handleFileInput (
             Vector< Studentas > & studentai)
```

5.5.1.6 handleOutput()

```
void handleOutput (
             const Vector< Studentas > & studentai)
```

5.5.1.7 ivestiStudentus()

```
void ivestiStudentus (
             Vector< Studentas > & studentai)
```

5.5.1.8 Mediana()

```
double Mediana (
             const Vector< int > & vec)
```

5.5.1.9 operator<<()

```
std::ostream & operator<< (
             std::ostream & out,
             const Studentas & s)
```

5.5.1.10 operator>>()

```
std::istream & operator>> (
             std::istream & in,
             Studentas & s)
```

5.5.1.11 rikiuotiStudentus()

```
void rikiuotiStudentus (
             Vector< Studentas > & studentai,
             char rikiavimas)
```

5.5.1.12 skaitymas()

```
bool skaitymas (
             Vector< Studentas > & studentai,
             const string & failoPav)
```

5.5.1.13 sortAndOutputStudents()

```
void sortAndOutputStudents (
             Vector< Studentas > & studentai)
```

5.5.1.14 spausdinti()

```
void spausdinti (
             const Vector< Studentas > & studentai,
             ostream & out)
```

5.5.2 Variable Documentation

5.5.2.1 rikiavimas

```
char rikiavimas
```

5.6 Studentas.h File Reference

```
#include <iostream>
#include "Vector.h"
#include <string>
#include <algorithm>
#include <iomanip>
#include <fstream>
#include <sstream>
#include <stdexcept>
#include <chrono>
#include <limits>
#include "Zmogus.h"
```

Classes

- class Studentas

Functions

- double Mediana (const Vector< int > &vec)
- void clearInput ()
- std::string generuotiVarda ()
- std::string generuotiPavarde ()
- bool skaitymas (Vector< Studentas > &studentai, const std::string &failoPav)
- void spausdinti (const Vector< Studentas > &studentai, std::ostream &out)
- void rikiuotiStudentus (Vector< Studentas > &studentai, char rikiavimas)
- void ivestiStudentus (Vector< Studentas > &studentai)
- void sortAndOutputStudents (Vector< Studentas > &studentai)
- void handleFileInput (Vector< Studentas > &studentai)
- void handleOutput (const Vector< Studentas > &studentai)
- void generateStudentFiles ()

5.6.1 Function Documentation

5.6.1.1 clearInput()

```
void clearInput ()
```

5.6.1.2 generateStudentFiles()

```
void generateStudentFiles ()
```

5.6.1.3 generuotiPavarde()

```
std::string generuotiPavarde ()
```

5.6.1.4 generuotiVarda()

```
std::string generuotiVarda ()
```

5.6.1.5 handleFileInput()

```
void handleFileInput (
             Vector< Studentas > & studentai)
```

5.6.1.6 handleOutput()

```
void handleOutput (
             const Vector< Studentas > & studentai)
```

5.6.1.7 ivestiStudentus()

```
void ivestiStudentus (
             Vector< Studentas > & studentai)
```

5.6.1.8 Mediana()

```
double Mediana (
             const Vector< int > & vec)
```

5.6.1.9 rikiuotiStudentus()

```
void rikiuotiStudentus (
             Vector< Studentas > & studentai,
             char rikiavimas)
```

5.6.1.10 skaitymas()

```
bool skaitymas (
             Vector< Studentas > & studentai,
             const std::string & failoPav)
```

5.6.1.11 sortAndOutputStudents()

```
void sortAndOutputStudents (
            Vector< Studentas > & studentai)
```

5.6.1.12 spausdinti()

```
void spausdinti (
            const Vector< Studentas > & studentai,
            std::ostream & out)
```

5.7 Studentas.h

Go to the documentation of this file.

```
#ifndef STUDENTAS_H
#define STUDENTAS_H

#include <iostream>
#include "Vector.h"
#include <string>
#include <algorithm>
#include <iomanip>
#include <fstream>
#include <sstream>
#include <stdexcept>
#include <chrono>
#include <limits>
#include "Zmogus.h"

class Studentas : public Zmogus {
private:
    Vector<int> nd_;
    int egz_;
    mutable double cachedVidurkis = -1;
    mutable double cachedMediana = -1;

public:
    // Konstruktoriai ir destruktorius
    Studentas();
    Studentas(const std::string& var, const std::string& pav, const Vector<int>& nd, int egz);
    ~Studentas();

    // Rule of Five
    Studentas(const Studentas& other);
    Studentas(Studentas&& other) noexcept;
    Studentas& operator=(const Studentas& other);
    Studentas& operator=(Studentas&& other) noexcept;

    // Get'eriai (override)
    std::string vardas() const override { return var_; }
    std::string pavarde() const override { return pav_; }
    Vector<int> namuDarbai() const { return nd_; }
    int egzaminas() const { return egz_; }

    // Set'eriai
    void setEgzaminas(int egz) { egz_ = egz; }

    // Galutinio balo skaičiavimas
    double galutinisVidurkis() const;
    double galutinisMediana() const;
    void skaiciuotiCache() const;

    // Įvesties/išvesties operatoriai
    friend std::istream& operator»(std::istream& in, Studentas& s);
    friend std::ostream& operator«(std::ostream& out, const Studentas& s);


    void info() const override {
        std::cout « "Studentas: " « var_ « " " « pav_ « std::endl;
    }
};

// Pagalbinės funkcijos
```

```
double Mediana(const Vector<int>& vec);
void clearInput();
std::string generuotiVarda();
std::string generuotiPavarde();
bool skaitymas(Vector<Studentas>& studentai, const std::string& failoPav);
void spausdinti(const Vector<Studentas>& studentai, std::ostream& out);
void rikiuotiStudentus(Vector<Studentas>& studentai, char rikiavimas);
void ivestiStudentus(Vector<Studentas>& studentai);
void sortAndOutputStudents(Vector<Studentas>& studentai);
void handleFileInput(Vector<Studentas>& studentai);
void handleOutput(const Vector<Studentas>& studentai);
void generateStudentFiles();

#endif // STUDENTAS_H
```

5.8 testStudentas.cpp File Reference

```
#include "Vector.h"
#include "Studentas.h"
#include "Zmogus.h"
#include <iostream>
#include <sstream>
#include <cassert>
```

Functions

- void spausdintiStudenta (const Studentas &s, const string &prefix)
- int main ()

5.8.1 Function Documentation

5.8.1.1 main()

```
int main ()
```

5.8.1.2 spausdintiStudenta()

```
void spausdintiStudenta (
            const Studentas & s,
            const string & prefix)
```

5.9 Vector.h File Reference

```
#include <cstddef>
#include <initializer_list>
#include <stdexcept>
#include <algorithm>
#include <iterator>
#include <memory>
#include <utility>
#include <type_traits>
#include <vector>
```

Classes

- class Vector< T >

Functions

- template<typename T>
  void swap (Vector< T > &a, Vector< T > &b) noexcept

5.9.1 Function Documentation

5.9.1.1 swap()

```
template<typename T>
void swap (
            Vector< T > & a,
            Vector< T > & b)   [noexcept]
```

5.10 Vector.h

Go to the documentation of this file.

```
#ifndef VECTOR_H
#define VECTOR_H

#include <cstddef>
#include <initializer_list>
#include <stdexcept>
#include <algorithm>
#include <iterator>
#include <memory>
#include <utility>
#include <type_traits>
#include <vector> // tik palyginimui

template <typename T>
class Vector {
public:
    // Member types
    using value_type = T;
    using size_type = std::size_t;
    using reference = T&;
    using const_reference = const T&;
    using pointer = T*;
    using const_pointer = const T*;
    using iterator = T*;
    using const_iterator = const T*;
    using reverse_iterator = std::reverse_iterator<iterator>;
    using const_reverse_iterator = std::reverse_iterator<const_iterator>;

private:
    pointer data_;
    size_type sz_;
    size_type cap_;

    void reallocate(size_type new_cap) {
        pointer new_data = new value_type[new_cap];
        for (size_type i = 0; i < sz_; ++i)
            new_data[i] = std::move(data_[i]);
        delete[] data_;
        data_ = new_data;
        cap_ = new_cap;
    }

public:
    // Constructors
    Vector() : data_(nullptr), sz_(0), cap_(0) {}
    Vector(size_type n, const T& val = T()) : data_(new T[n]), sz_(n), cap_(n) {
        std::fill(data_, data_ + n, val);
    }
    Vector(std::initializer_list<T> il) : Vector(il.size()) {
        std::copy(il.begin(), il.end(), data_);
    }
    Vector(const Vector& other) : data_(new T[other.cap_]), sz_(other.sz_), cap_(other.cap_) {
        std::copy(other.data_, other.data_ + sz_, data_);
    }
    Vector(Vector&& other) noexcept : data_(other.data_), sz_(other.sz_), cap_(other.cap_) {
        other.data_ = nullptr; other.sz_ = 0; other.cap_ = 0;
    }
    Vector& operator=(const Vector& other) {
        if (this != &other) {
            delete[] data_;
            sz_ = other.sz_;
            cap_ = other.cap_;
            data_ = new T[cap_];
            std::copy(other.data_, other.data_ + sz_, data_);
        }
        return *this;
    }
    Vector& operator=(Vector&& other) noexcept {
        if (this != &other) {
            delete[] data_;
            data_ = other.data_;
            sz_ = other.sz_;
            cap_ = other.cap_;
            other.data_ = nullptr; other.sz_ = 0; other.cap_ = 0;
        }
        return *this;
    }
    ~Vector() { delete[] data_; }

    // Element access
    reference operator[](size_type i) { return data_[i]; }
    const_reference operator[](size_type i) const { return data_[i]; }
    reference at(size_type i) {
        if (i >= sz_) throw std::out_of_range("Vector::at");
        return data_[i];
    }
    const_reference at(size_type i) const {
        if (i >= sz_) throw std::out_of_range("Vector::at");
        return data_[i];
    }
    reference front() { return data_[0]; }
    const_reference front() const { return data_[0]; }
    reference back() { return data_[sz_ - 1]; }
    const_reference back() const { return data_[sz_ - 1]; }
    pointer data() noexcept { return data_; }
    const_pointer data() const noexcept { return data_; }

    // Iterators
    iterator begin() noexcept { return data_; }
    const_iterator begin() const noexcept { return data_; }
    iterator end() noexcept { return data_ + sz_; }
    const_iterator end() const noexcept { return data_ + sz_; }
    reverse_iterator rbegin() noexcept { return reverse_iterator(end()); }
    const_reverse_iterator rbegin() const noexcept { return const_reverse_iterator(end()); }
    reverse_iterator rend() noexcept { return reverse_iterator(begin()); }
    const_reverse_iterator rend() const noexcept { return const_reverse_iterator(begin()); }

    // Capacity
    bool empty() const noexcept { return sz_ == 0; }
    size_type size() const noexcept { return sz_; }
    size_type capacity() const noexcept { return cap_; }
    void reserve(size_type n) {
        if (n > cap_) reallocate(n);
    }
    void shrink_to_fit() {
        if (sz_ < cap_) reallocate(sz_);
    }

    // Modifiers
    void clear() noexcept { sz_ = 0; }
    void push_back(const T& val) {
        if (sz_ == cap_) reserve(cap_ == 0 ? 1 : cap_ * 2);
        data_[sz_++] = val;
    }
    void push_back(T&& val) {
        if (sz_ == cap_) reserve(cap_ == 0 ? 1 : cap_ * 2);
        data_[sz_++] = std::move(val);
    }
    void pop_back() {
        if (sz_ > 0) --sz_;
    }
    void resize(size_type n, const T& val = T()) {
        if (n > cap_) reserve(n);
        if (n > sz_) std::fill(data_ + sz_, data_ + n, val);
        sz_ = n;
    }
    void swap(Vector& other) noexcept {
        std::swap(data_, other.data_);
        std::swap(sz_, other.sz_);
        std::swap(cap_, other.cap_);
    }

    // Insert element at position
    iterator insert(const_iterator pos, const T& value) {
        size_type idx = pos - data_;
        if (sz_ == cap_) reserve(cap_ == 0 ? 1 : cap_ * 2);
        for (size_type i = sz_; i > idx; --i)
            data_[i] = std::move(data_[i - 1]);
        data_[idx] = value;
        ++sz_;
        return data_ + idx;
    }

    // Erase element at position
    iterator erase(const_iterator pos) {
        size_type idx = pos - data_;
        for (size_type i = idx; i + 1 < sz_; ++i)
            data_[i] = std::move(data_[i + 1]);
        --sz_;
        return data_ + idx;
    }

    // Assign n copies of value
    void assign(size_type n, const T& value) {
        if (n > cap_) reserve(n);
        std::fill(data_, data_ + n, value);
        sz_ = n;
    }

    // Assign from initializer_list
    void assign(std::initializer_list<T> il) {
        if (il.size() > cap_) reserve(il.size());
        std::copy(il.begin(), il.end(), data_);
        sz_ = il.size();
    }

    // Assign from iterator range
template <typename InputIt>
typename std::enable_if<!std::is_integral<InputIt>::value, void>::type
assign(InputIt first, InputIt last) {
    size_type n = std::distance(first, last);
    if (n > cap_) reserve(n);
    std::copy(first, last, data_);
    sz_ = n;
    }

    // Emplace back
    template <typename... Args>
    void emplace_back(Args&&... args) {
        if (sz_ == cap_) reserve(cap_ == 0 ? 1 : cap_ * 2);
        new (data_ + sz_) T(std::forward<Args>(args)...);
        ++sz_;
    }

    // Insert range [first, last) at position
    template <typename InputIt>
    iterator insert(const_iterator pos, InputIt first, InputIt last) {
        size_type idx = pos - data_;
        size_type count = std::distance(first, last);
        if (sz_ + count > cap_) reserve(std::max(cap_ * 2, sz_ + count));
        for (size_type i = sz_ + count; i-- > idx + count; )
            data_[i] = std::move(data_[i - count]);
        for (size_type i = 0; i < count; ++i)
            data_[idx + i] = *(first++);
        sz_ += count;
        return data_ + idx;
    }

    // Insert initializer_list at position
    iterator insert(const_iterator pos, std::initializer_list<T> ilist) {
        return insert(pos, ilist.begin(), ilist.end());
    }

    // Erase range [first, last)
    iterator erase(const_iterator first, const_iterator last) {
        size_type idx_first = first - data_;
        size_type idx_last = last - data_;
        size_type count = idx_last - idx_first;
        for (size_type i = idx_first; i + count < sz_; ++i)
```

```
                data_[i] = std::move(data_[i + count]);
            sz_ -= count;
            return data_ + idx_first;
        }

        // Emplace element at position
        template <typename... Args>
        iterator emplace(const_iterator pos, Args&&... args) {
            size_type idx = pos - data_;
            if (sz_ == cap_) reserve(cap_ == 0 ? 1 : cap_ * 2);
            for (size_type i = sz_; i > idx; --i)
                data_[i] = std::move(data_[i - 1]);
            new (data_ + idx) T(std::forward<Args>(args)...);
            ++sz_;
            return data_ + idx;
        }

        // get_allocator
        std::allocator<T> get_allocator() const { return std::allocator<T>(); }

        // Comparison operators
        bool operator==(const Vector& other) const {
            if (sz_ != other.sz_) return false;
            for (size_type i = 0; i < sz_; ++i)
                if (!(data_[i] == other.data_[i])) return false;
            return true;
        }
        bool operator!=(const Vector& other) const { return !(*this == other); }
        bool operator<(const Vector& other) const {
            return std::lexicographical_compare(begin(), end(), other.begin(), other.end());
        }
        bool operator>(const Vector& other) const { return other < *this; }
        bool operator<=(const Vector& other) const { return !(other < *this); }
        bool operator>=(const Vector& other) const { return !(*this < other); }
    };

    // Non-member swap
    template <typename T>
    void swap(Vector<T>& a, Vector<T>& b) noexcept {
        a.swap(b);
    }

    #endif // VECTOR_H
```

5.11 Zmogus.h File Reference

```
#include <string>
#include <iostream>
```

Classes

- class Zmogus

5.12 Zmogus.h

Go to the documentation of this file.

```
#ifndef ZMOGUS_H
#define ZMOGUS_H

#include <string>
#include <iostream>

class Zmogus {
protected:
    std::string var_;
    std::string pav_;
public:
    Zmogus() : var_(""), pav_("") {}
    Zmogus(const std::string& var, const std::string& pav) : var_(var), pav_(pav) {}
    virtual ~Zmogus() = default;

    virtual std::string vardas() const { return var_; }
    virtual std::string pavarde() const { return pav_; }

    // Grynai virtuali funkcija (abstrakti)
    virtual void info() const = 0;
};

#endif // ZMOGUS_H
```

Index