My Project

Generated by Doxygen 1.13.2

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:

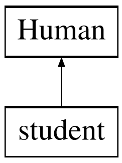
Chapter 4

Class Documentation

4.1 Human Class Reference

```
#include <student.h>
```

Inheritance diagram for Human:



Public Member Functions

- Human ()
- Human (const std::string &v, const std::string &p)
- const std::string & getVardas () const
- const std::string & getPavarde () const
- void setVardas (const std::string &v)
- void setPavarde (const std::string &p)
- virtual ~Human ()=default
- virtual void skaiciuotiGalutini (char galutinioBudas)=0

Protected Attributes

- std::string vardas
- std::string pavarde

4.1.1 Constructor & Destructor Documentation

4.1.1.1 Human() [1/2]

```
Human::Human ()  [inline]
```

4.1.1.2 Human() [2/2]

```
Human::Human (
            const std::string & v,
            const std::string & p)  [inline]
```

4.1.1.3 ∼Human()

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

4.1.2 Member Function Documentation

4.1.2.1 getPavarde()

```
const std::string & Human::getPavarde () const  [inline]
```

4.1.2.2 getVardas()

```
const std::string & Human::getVardas () const  [inline]
```

4.1.2.3 setPavarde()

```
void Human::setPavarde (
            const std::string & p)  [inline]
```

4.1.2.4 setVardas()

```
void Human::setVardas (
            const std::string & v)  [inline]
```

4.1.2.5 skaiciuotiGalutini()

```
virtual void Human::skaiciuotiGalutini (
            char galutinioBudas)  [pure virtual]
```

Implemented in student.

4.1.3 Member Data Documentation

4.1.3.1 pavarde

```
std::string Human::pavarde  [protected]
```

4.1.3.2 vardas

```
std::string Human::vardas  [protected]
```

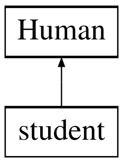
The documentation for this class was generated from the following file:

- vector/include/student.h

4.2 student Class Reference

```
#include <student.h>
```

Inheritance diagram for student:



Public Member Functions

- student ()=default
- student (const student &other) noexcept
- student & operator= (const student &other) noexcept
- student (student &&other) noexcept
- student & operator= (student &&other) noexcept
- student (std::string v, std::string p, Vector< float > pazymiai, float egz) noexcept
- void setPazymiai (Vector< float > paz) noexcept
- void setEgzaminoRezultatas (float egz) noexcept
- void setGalutinisV (float V) noexcept
- void setGalutinisM (float M) noexcept
- const Vector< float > & getPazymiai () const
- float getEgzaminoRezultatas () const
- float getGalutinisV () const
- float getGalutinisM () const
- float skaiciuotiVid () const
- float skaiciuotiMed () const
- void skaiciuotiGalutini (char galutinioBudas) override
- void addPazymys (float pazymys)

Public Member Functions inherited from Human

- Human ()
- Human (const std::string &v, const std::string &p)
- const std::string & getVardas () const
- const std::string & getPavarde () const
- void setVardas (const std::string &v)
- void setPavarde (const std::string &p)
- virtual ~Human ()=default

Private Attributes

- Vector< float > pazymiai {}
- float egzaminoRezultatas = 0.0f
- float galutinisM = 0.0f
- float galutinisV = 0.0f

Friends

- std::ostream & operator<< (std::ostream &os, const student &studentas)
- std::istream & operator>> (std::istream &in, student &studentas)

Additional Inherited Members

Protected Attributes inherited from Human

- std::string vardas
- std::string pavarde

4.2.1 Constructor & Destructor Documentation

4.2.1.1 student() [1/4]

```
student::student ()  [default]
```

4.2.1.2 student() [2/4]

```
student::student (
            const student & other)  [inline], [noexcept]
```

4.2.1.3 student() [3/4]

```
student::student (
            student && other)  [inline], [noexcept]
```

4.2.1.4 student() [4/4]

```
student::student (
            std::string v,
            std::string p,
            Vector< float > pazymiai,
            float egz)  [inline], [noexcept]
```

4.2.2 Member Function Documentation

4.2.2.1 addPazymys()

```
void student::addPazymys (
             float pazymys)  [inline]
```

4.2.2.2 getEgzaminoRezultatas()

```
float student::getEgzaminoRezultatas () const  [inline]
```

4.2.2.3 getGalutinisM()

```
float student::getGalutinisM () const  [inline]
```

4.2.2.4 getGalutinisV()

```
float student::getGalutinisV () const  [inline]
```

4.2.2.5 getPazymiai()

```
const Vector< float > & student::getPazymiai () const  [inline]
```

4.2.2.6 operator=() [1/2]

```
student & student::operator= (
             const student & other)  [inline], [noexcept]
```

4.2.2.7 operator=() [2/2]

```
student & student::operator= (
             student && other)  [inline], [noexcept]
```

4.2.2.8 setEgzaminoRezultatas()

```
void student::setEgzaminoRezultatas (
             float egz)  [inline], [noexcept]
```

4.2.2.9 setGalutinisM()

```
void student::setGalutinisM (
             float M)  [inline], [noexcept]
```

4.2.2.10 setGalutinisV()

```
void student::setGalutinisV (
            float V)  [inline], [noexcept]
```

4.2.2.11 setPazymiai()

```
void student::setPazymiai (
            Vector< float > paz)  [inline], [noexcept]
```

4.2.2.12 skaiciuotiGalutini()

```
void student::skaiciuotiGalutini (
            char galutinioBudas)  [inline], [override], [virtual]
```

Implements Human.

4.2.2.13 skaiciuotiMed()

```
float student::skaiciuotiMed () const
```

4.2.2.14 skaiciuotiVid()

```
float student::skaiciuotiVid () const
```

4.2.3 Friends And Related Symbol Documentation

4.2.3.1 operator<<

```
std::ostream & operator<< (
            std::ostream & os,
            const student & studentas)  [friend]
```

4.2.3.2 operator>>

```
std::istream & operator>> (
            std::istream & in,
            student & studentas)  [friend]
```

4.2.4 Member Data Documentation

4.2.4.1 egzaminoRezultatas

```
float student::egzaminoRezultatas = 0.0f  [private]
```

4.2.4.2 galutinisM

```
float student::galutinisM = 0.0f  [private]
```

4.2.4.3 galutinisV

```
float student::galutinisV = 0.0f  [private]
```

4.2.4.4 pazymiai

```
Vector<float> student::pazymiai {}  [private]
```

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

- vector/include/student.h
- vector/source/student.cpp

4.3 Vector< T > Class Template Reference

```
#include <vector.h>
```

Public Types

- using iterator = T∗
- using const_iterator = const T∗

Public Member Functions

- Vector ()
- Vector (std::initializer_list< T > init)
- ∼Vector ()
- Vector (const Vector &other)
- Vector & operator= (const Vector &other)
- Vector (Vector &&other) noexcept
- Vector & operator= (Vector &&other) noexcept
- bool operator== (const Vector &o) const
- bool operator!= (const Vector &o) const
- size_t size () const
- size_t capacity () const
- bool empty () const
- void reserve (size_t newCap)
- void shrink_to_fit ()
- void clear () noexcept
- void resize (size_t count, const T &value=T())
- void push_back (const T &value)
- void push_back (T &&value)
- void pop_back ()

- void swap (Vector &other) noexcept
- void assign (size_t n, const T &value)
- template<typename InputIt>
  void assign (InputIt first, InputIt last)
- iterator begin ()
- const_iterator begin () const
- iterator end ()
- const_iterator end () const
- iterator insert (const_iterator pos, const T &value)
- iterator erase (const_iterator pos)
- template<typename... Args>
  void emplace_back (Args &&... args)
- T & operator[ ] (size_t i)
- const T & operator[ ] (size_t i) const
- T & at (size_t i)
- const T & at (size_t i) const
- T & front ()
- const T & front () const
- T & back ()
- const T & back () const
- Vector & operator= (std::initializer_list< T > init)

Private Member Functions

- void reallocate (size_t newCap)

Private Attributes

- T * data
- size_t size_
- size_t capacity_

Friends

- template<typename U>
  bool operator< (const Vector< U > &a, const Vector< U > &b)

4.3.1 Member Typedef Documentation

4.3.1.1 const_iterator

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

4.3.1.2 iterator

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

4.3.2 Constructor & Destructor Documentation

4.3.2.1 Vector() [1/4]

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

4.3.2.2 Vector() [2/4]

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

4.3.2.3 ∼Vector()

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

4.3.2.4 Vector() [3/4]

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

4.3.2.5 Vector() [4/4]

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

4.3.3 Member Function Documentation

4.3.3.1 assign() [1/2]

```
template<typename T>
template<typename InputIt>
void Vector< T >::assign (
            InputIt first,
            InputIt last)  [inline]
```

4.3.3.2 assign() [2/2]

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

4.3.3.3 at() [1/2]

```
template<typename T>
T & Vector< T >::at (
             size_t i)  [inline]
```

4.3.3.4 at() [2/2]

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

4.3.3.5 back() [1/2]

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

4.3.3.6 back() [2/2]

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

4.3.3.7 begin() [1/2]

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

4.3.3.8 begin() [2/2]

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

4.3.3.9 capacity()

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

4.3.3.10 clear()

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

4.3.3.11 emplace_back()

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

4.3.3.12 empty()

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

4.3.3.13 end() [1/2]

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

4.3.3.14 end() [2/2]

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

4.3.3.15 erase()

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

4.3.3.16 front() [1/2]

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

4.3.3.17 front() [2/2]

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

4.3.3.18 insert()

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

4.3.3.19 operator"!=()

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

4.3.3.20 operator=() [1/3]

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

4.3.3.21 operator=() [2/3]

```
template<typename T>
Vector & Vector< T >::operator= (
             std::initializer_list< T > init)  [inline]
```

4.3.3.22 operator=() [3/3]

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

4.3.3.23 operator==()

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

4.3.3.24 operator[]() [1/2]

```
template<typename T>
T & Vector< T >::operator[] (
             size_t i)  [inline]
```

4.3.3.25 operator[]() [2/2]

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

4.3.3.26 pop_back()

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

4.3.3.27 push_back() [1/2]

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

4.3.3.28 push_back() [2/2]

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

4.3.3.29 reallocate()

```
template<typename T>
void Vector< T >::reallocate (
             size_t newCap)  [inline], [private]
```

4.3.3.30 reserve()

```
template<typename T>
void Vector< T >::reserve (
             size_t newCap)  [inline]
```

4.3.3.31 resize()

```
template<typename T>
void Vector< T >::resize (
             size_t count,
             const T & value = T())  [inline]
```

4.3.3.32 shrink_to_fit()

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

4.3.3.33 size()

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

4.3.3.34 swap()

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

4.3.4 Friends And Related Symbol Documentation

4.3.4.1 operator<

```
template<typename T>
template<typename U>
bool operator< (
            const Vector< U > & a,
            const Vector< U > & b)  [friend]
```

4.3.5 Member Data Documentation

4.3.5.1 capacity_

```
template<typename T>
size_t Vector< T >::capacity_  [private]
```

4.3.5.2 data

```
template<typename T>
T* Vector< T >::data  [private]
```

4.3.5.3 size_

```
template<typename T>
size_t Vector< T >::size_  [private]
```

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

- vector/include/vector.h

Chapter 5

File Documentation

5.1 vector/include/functions.h File Reference

this file contains functions declarations

```
#include <vector>
#include "../include/student.h"
#include "../include/vector.h"
```

Functions

- void rusiuotiOutput (Vector< student > &grupe, char rusiavimoBudas, char galutinioBudas)
- void spausdinimasTerminale (const Vector< student > &grupe, char galutinioBudas)
- void spausdinimasFaile (const Vector< student > &grupe, char galutinioBudas)
- void generuotiFaila (int pKiekis, int studentuKiekis, const std::string &failoPavadinimas)
- void spausdinimas (char spausBudas, char rusiavimoBudas, char galutinioBudas, Vector< student > &grupe)
- void sortedStudentSpausdinimas (std::string lowGradeFailas, std::string highGradeFailas, Vector< student > &nepazangus, Vector< student > &normalus, char galutinioBudas)
- void nuskaitytiGeneruotusFailus (const std::string &failoPavadinimas, Vector< student > &grupe, int pKiekis, char galutinioBudas)
- void skirstytiStudentus (Vector< student > &grupe, Vector< student > &nepazangus, Vector< student > &normalus, char galutinioBudas)
- void pirmas (Vector< student > &grupe, char spausBudas, char rusiavimoBudas, char galutinioBudas, int pKiekis)
- void antras (Vector< student > &grupe, char spausBudas, char rusiavimoBudas, char galutinioBudas, int pKiekis)
- void trecias (Vector< student > &grupe, const Vector< std::string > &vardai, const Vector< std::string > &pavardes, char spausBudas, char rusiavimoBudas, char galutinioBudas, int pKiekis)
- void ketvirtas (Vector< student > &grupe, int pKiekis, char galutinioBudas)
- void penktas (int pKiekis)
- void sestas (Vector< student > &grupe, Vector< student > &testGrupe, Vector< student > &nepazangus, Vector< student > &normalus, char galutinioBudas, char rusiavimoBudas, int pKiekis)
- void septintas (char galutinioBudas)
- void astuntas ()

5.1.1 Detailed Description

this file contains functions declarations

Author

Narbas

Version

v2.0

Date

2025-05-07

Copyright

Copyright (c) 2025

5.1.2 Function Documentation

5.1.2.1 antras()

```
void antras (
            Vector< student > & grupe,
            char spausBudas,
            char rusiavimoBudas,
            char galutinioBudas,
            int pKiekis)
```

5.1.2.2 astuntas()

```
void astuntas ()
```

5.1.2.3 generuotiFaila()

```
void generuotiFaila (
            int pKiekis,
            int studentuKiekis,
            const std::string & failoPavadinimas)
```

5.1.2.4 ketvirtas()

```
void ketvirtas (
            Vector< student > & grupe,
            int pKiekis,
            char galutinioBudas)
```

5.1.2.5 nuskaitytiGeneruotusFailus()

```
void nuskaitytiGeneruotusFailus (
             const std::string & failoPavadinimas,
             Vector< student > & grupe,
             int pKiekis,
             char galutinioBudas)
```

5.1.2.6 penktas()

```
void penktas (
             int pKiekis)
```

5.1.2.7 pirmas()

```
void pirmas (
             Vector< student > & grupe,
             char spausBudas,
             char rusiavimoBudas,
             char galutinioBudas,
             int pKiekis)
```

5.1.2.8 rusiuotiOutput()

```
void rusiuotiOutput (
             Vector< student > & grupe,
             char rusiavimoBudas,
             char galutinioBudas)
```

5.1.2.9 septintas()

```
void septintas (
             char galutinioBudas)
```

5.1.2.10 sestas()

```
void sestas (
             Vector< student > & grupe,
             Vector< student > & testGrupe,
             Vector< student > & nepazangus,
             Vector< student > & normalus,
             char galutinioBudas,
             char rusiavimoBudas,
             int pKiekis)
```

5.1.2.11 skirstytiStudentus()

```
void skirstytiStudentus (
            Vector< student > & grupe,
            Vector< student > & nepazangus,
            Vector< student > & normalus,
            char galutinioBudas)
```

5.1.2.12 sortedStudentSpausdinimas()

```
void sortedStudentSpausdinimas (
            std::string lowGradeFailas,
            std::string highGradeFailas,
            Vector< student > & nepazangus,
            Vector< student > & normalus,
            char galutinioBudas)
```

5.1.2.13 spausdinimas()

```
void spausdinimas (
            char spausBudas,
            char rusiavimoBudas,
            char galutinioBudas,
            Vector< student > & grupe)
```

5.1.2.14 spausdinimasFaile()

```
void spausdinimasFaile (
            const Vector< student > & grupe,
            char galutinioBudas)
```

5.1.2.15 spausdinimasTerminale()

```
void spausdinimasTerminale (
            const Vector< student > & grupe,
            char galutinioBudas)
```

5.1.2.16 trecias()

```
void trecias (
            Vector< student > & grupe,
            const Vector< std::string > & vardai,
            const Vector< std::string > & pavardes,
            char spausBudas,
            char rusiavimoBudas,
            char galutinioBudas,
            int pKiekis)
```

5.2 functions.h

Go to the documentation of this file.

```
#ifndef FUNCTIONS_H
#define FUNCTIONS_H

#include <vector>
#include "../include/student.h"
#include "../include/vector.h"

void rusiuotiOutput(Vector<student>& grupe, char rusiavimoBudas, char galutinioBudas);
void spausdinimasTerminale(const Vector<student>& grupe, char galutinioBudas);
void spausdinimasFaile(const Vector<student>& grupe, char galutinioBudas);
void generuotiFaila(int pKiekis, int studentuKiekis, const std::string& failoPavadinimas);
void spausdinimas(char spausBudas, char rusiavimoBudas, char galutinioBudas, Vector<student>&grupe);
void sortedStudentSpausdinimas(std::string lowGradeFailas, std::string highGradeFailas,
    Vector<student>&nepazangus, Vector<student>&normalus, char galutinioBudas);
void nuskaitytiGeneruotusFailus(const std::string& failoPavadinimas, Vector<student> & grupe, int
    pKiekis, char galutinioBudas);
void skirstytiStudentus(Vector<student> & grupe, Vector<student>& nepazangus, Vector<student>&
    normalus, char galutinioBudas);

void pirmas(Vector<student>& grupe, char spausBudas, char rusiavimoBudas, char galutinioBudas, int
    pKiekis);
void antras(Vector<student>& grupe, char spausBudas, char rusiavimoBudas, char galutinioBudas, int
    pKiekis);
void trecias(Vector<student>& grupe, const Vector<std::string>& vardai, const Vector<std::string>&
    pavardes, char spausBudas, char rusiavimoBudas, char galutinioBudas, int pKiekis);
void ketvirtas(Vector<student>& grupe, int pKiekis, char galutinioBudas);
void penktas(int pKiekis);
void sestas(Vector<student>& grupe, Vector<student>& testGrupe, Vector<student>& nepazangus,
    Vector<student>& normalus, char galutinioBudas, char rusiavimoBudas, int pKiekis);
void septintas(char galutinioBudas);
void astuntas();
#endif
```

5.3 vector/include/student.h File Reference

this file contains Human and Student classes

```
#include <vector>
#include <string>
#include <iostream>
#include "../include/vector.h"
```

Classes

- class Human
- class student

Functions

- std::ostream & operator<< (std::ostream &os, const student &studentas)
- std::istream & operator>> (std::istream &in, student &studentas)

5.3.1 Detailed Description

this file contains Human and Student classes

Author

Narbas

Version

v2.0

Date

2025-05-07

Copyright

Copyright (c) 2025

5.3.2 Function Documentation

5.3.2.1 operator<<()

```
std::ostream & operator<< (
             std::ostream & os,
             const student & studentas)
```

5.3.2.2 operator>>()

```
std::istream & operator>> (
             std::istream & in,
             student & studentas)
```

5.4 student.h

Go to the documentation of this file.

```
#ifndef STUDENT_H
#define STUDENT_H


#include <vector>
#include <string>
#include <iostream>
#include "../include/vector.h"
class Human{
    protected:

        std::string vardas;
        std::string pavarde;

    public:

    Human() : vardas(""), pavarde("") {}
    Human(const std::string& v, const std::string& p) : vardas(v), pavarde(p) {}
```

```cpp

    //getters
    const std::string& getVardas() const { return vardas; }
    const std::string& getPavarde() const { return pavarde; }
    //setters
    void setVardas(const std::string& v)   { vardas = v; }
    void setPavarde(const std::string& p)  { pavarde = p; }

    virtual ~Human() = default;

    virtual void skaiciuotiGalutini(char galutinioBudas) = 0;

};

class student : public Human{

    private:

        Vector<float> pazymiai{};
        float egzaminoRezultatas = 0.0f;
        float galutinisM = 0.0f;
        float galutinisV = 0.0f;

    public:

    student() = default;

    //rule of 5-----------------------------------------------------
    //copy
    student(const student &other) noexcept : Human(other.getVardas(), other.getPavarde()), pazymiai(other.pazymiai),
                                    egzaminoRezultatas(other.egzaminoRezultatas), galutinisM(other.galutinisM), galutinisV(other.galutinisV){};
    //copy asg
    student& operator=(const student &other) noexcept {
        if(this != &other){
            setVardas(other.getVardas());
            setPavarde(other.getPavarde());
            pazymiai = other.pazymiai;
            egzaminoRezultatas = other.egzaminoRezultatas;
            galutinisM = other.galutinisM;
            galutinisV = other.galutinisV;
        }

        return *this;
    };
    //move
    student(student &&other) noexcept : Human(std::move(other.vardas), std::move(other.pavarde)),
      pazymiai(std::move(other.pazymiai)),
      egzaminoRezultatas(other.egzaminoRezultatas),
      galutinisM(other.galutinisM),
      galutinisV(other.galutinisV) {
            other.setEgzaminoRezultatas(0);
            other.setGalutinisM(0);
            other.setGalutinisV(0);
      }
    //move asg
    student& operator=(student &&other) noexcept {
        if(this != &other){
            vardas = std::move(other.vardas);
            pavarde = std::move(other.pavarde);
            pazymiai = std::move(other.pazymiai);
            egzaminoRezultatas = other.egzaminoRezultatas;
            galutinisM = other.galutinisM;
            galutinisV = other.galutinisV;

            other.setEgzaminoRezultatas(0);
            other.setGalutinisM(0);
            other.setGalutinisV(0);
        }
        return *this;
    };
    // -------------------------------------------------------------

    //isvestis, ivestis overloads
    friend std::ostream& operator « (std::ostream& os, const student& studentas);
    friend std::istream& operator » (std::istream& in, student& studentas);

    //parametrizuotas ctor
    student(std::string v, std::string p, Vector<float> pazymiai, float egz) noexcept : Human(v, p), pazymiai(std::move(pazymiai)), egzaminoRezultatas(egz) {}

    //setters
    void setPazymiai(Vector<float> paz) noexcept{ pazymiai = std::move(paz); }
    void setEgzaminoRezultatas(float egz) noexcept { egzaminoRezultatas = egz; }
    void setGalutinisV(float V) noexcept { galutinisV = V; }
    void setGalutinisM(float M) noexcept { galutinisM = M; }

    //getters
    const Vector<float>& getPazymiai() const { return pazymiai; }
    float getEgzaminoRezultatas() const { return egzaminoRezultatas; }
    float getGalutinisV() const { return galutinisV; }
    float getGalutinisM() const { return galutinisM; }

    //methods
    float skaiciuotiVid() const;
    float skaiciuotiMed() const;
    void skaiciuotiGalutini(char galutinioBudas) override;
    void addPazymys(float pazymys) { pazymiai.push_back(pazymys); }
};

std::ostream& operator«(std::ostream& os, const student& studentas);
std::istream& operator»(std::istream& in, student& studentas);

#endif
```

5.5 vector/include/utils.h File Reference

this file contains templates

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

Functions

- template<typename T>
  T tikrintiInput (const std::string &prompt, const std::string &klaida)

5.5.1 Detailed Description

this file contains templates

Author

Narbas

Version

v2.0

Date

2025-05-07

Copyright

Copyright (c) 2025

5.5.2 Function Documentation

5.5.2.1 tikrintiInput()

```
template<typename T>
T tikrintiInput (
            const std::string & prompt,
            const std::string & klaida)
```

5.6 utils.h

Go to the documentation of this file.

```
#ifndef UTILS_H
#define UTILS_H


#include <iostream>
#include <limits>
#include <string>

template<typename T>
T tikrintiInput(const std::string& prompt, const std::string& klaida) {
    T value;
    while (true) {
        std::cout « prompt;
        std::cin » value;
        if (!std::cin.fail()) {
            std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
            return value;
        }
        std::cerr « klaida « std::endl;
        std::cin.clear();
        std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
    }
}

#endif
```

5.7 vector/include/vector.h File Reference

```
#include <iostream>
#include <initializer_list>
#include <stdexcept>
#include <algorithm>
#include <iterator>
#include <utility>
#include <memory>
```

Classes

- class Vector< T >

Functions

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

5.7.1 Function Documentation

5.7.1.1 operator<()

```
template<typename T>
bool operator< (
             const Vector< T > & a,
             const Vector< T > & b)
```

5.7.1.2 operator<=()

```
template<typename T>
bool operator<= (
             const Vector< T > & a,
             const Vector< T > & b)
```

5.7.1.3 operator>()

```
template<typename T>
bool operator> (
             const Vector< T > & a,
             const Vector< T > & b)
```

5.7.1.4 operator>=()

```
template<typename T>
bool operator>= (
             const Vector< T > & a,
             const Vector< T > & b)
```

5.7.1.5 swap()

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

5.8 vector.h

Go to the documentation of this file.

```
#ifndef VECTOR_H
#define VECTOR_H

#include <iostream>
#include <initializer_list>
#include <stdexcept>
#include <algorithm>
#include <iterator>
#include <utility>
#include <memory>

template<typename T>
class Vector {
private:
    T* data;
    size_t size_;
    size_t capacity_;

    void reallocate(size_t newCap) {
        T* newData = static_cast<T*>(::operator new(newCap * sizeof(T)));

        for (size_t i = 0; i < size_; ++i) {
            try {
                new(newData + i) T(std::move(data[i]));
            } catch (...) {
                for (size_t j = 0; j < i; ++j) {
                    (newData + j)->~T();
                }
                ::operator delete(newData);
                throw;
            }
        }
        for (size_t i = 0; i < size_; ++i) {
            (data + i)->~T();
        }
        ::operator delete(data);
        data = newData;
        capacity_ = newCap;
    }

public:
    // default ctor
    Vector() : data(nullptr), size_(0), capacity_(0) {}

    // initializer-list ctor
    Vector(std::initializer_list<T> init)
      : data(nullptr), size_(0), capacity_(0)
    {
        reserve(init.size());
        for (const auto& x : init) {
            push_back(x);
        }
    }

    ~Vector() {
        clear();
        ::operator delete(data);
    }

    Vector(const Vector& other)
      : data(nullptr), size_(0), capacity_(0)
    {
        if (other.size_ > 0) {
            reserve(other.capacity_);
            for (size_t i = 0; i < other.size_; ++i) {
                try {
                    new(data + i) T(other.data[i]);
                    ++size_;
                } catch (...) {
                    for (size_t j = 0; j < i; ++j) {
                        (data + j)->~T();
                    }
                    ::operator delete(data);
                    data = nullptr;
                    size_ = 0;
                    capacity_ = 0;
                    throw;
                }
            }
        }
    }

    Vector& operator=(const Vector& other) {
        if (this != &other) {
            Vector temp(other);
            swap(temp);
        }
        return *this;
    }

    Vector(Vector&& other) noexcept
      : data(other.data), size_(other.size_), capacity_(other.capacity_)
    {
        other.data = nullptr;
        other.size_ = 0;
        other.capacity_ = 0;
    }

    Vector& operator=(Vector&& other) noexcept {
        if (this != &other) {
            clear();
            ::operator delete(data);

            data = other.data;
            size_ = other.size_;
            capacity_ = other.capacity_;

            other.data = nullptr;
            other.size_ = 0;
            other.capacity_ = 0;
        }
        return *this;
    }

    bool operator==(const Vector& o) const {
        if (size_ != o.size_) return false;
        for (size_t i = 0; i < size_; ++i)
            if (data[i] != o.data[i]) return false;
        return true;
    }

    bool operator!=(const Vector& o) const {
        return !(*this == o);
    }

    size_t size() const { return size_; }
    size_t capacity() const { return capacity_; }
    bool   empty() const { return size_ == 0; }

    void reserve(size_t newCap) {
        if (newCap > capacity_) {
            if (capacity_ == 0) {
                data = static_cast<T*>(::operator new(newCap * sizeof(T)));
                capacity_ = newCap;
            } else {
                reallocate(newCap);
            }
        }
    }

    void shrink_to_fit() {
        if (capacity_ > size_) {
            if (size_ == 0) {
                ::operator delete(data);
                data = nullptr;
                capacity_ = 0;
            } else {
                reallocate(size_);
            }
        }
    }

    void clear() noexcept {
        for (size_t i = 0; i < size_; ++i) {
            (data + i)->~T();
        }
        size_ = 0;
    }

    void resize(size_t count, const T& value = T()) {
        if (count > capacity_) {
            reserve(count);
        }

        if (count > size_) {
            // Construct new elements
            for (size_t i = size_; i < count; ++i) {
                new(data + i) T(value);
            }
        } else if (count < size_) {
            // Destroy excess elements
            for (size_t i = count; i < size_; ++i) {
                (data + i)->~T();
            }
        }
        size_ = count;
    }

    void push_back(const T& value) {
        if (size_ == capacity_) {
            reserve(capacity_ == 0 ? 1 : capacity_ * 2);
        }
        new(data + size_) T(value);
        ++size_;
    }

    void push_back(T&& value) {
        if (size_ == capacity_) {
            reserve(capacity_ == 0 ? 1 : capacity_ * 2);
        }
        new(data + size_) T(std::move(value));
        ++size_;
    }

    void pop_back() {
        if (size_ == 0) {
            throw std::out_of_range("pop_back() on empty Vector");
        }
        --size_;
        (data + size_)->~T();
    }

    void swap(Vector& other) noexcept {
        std::swap(data, other.data);
        std::swap(size_, other.size_);
        std::swap(capacity_, other.capacity_);
    }

    void assign(size_t n, const T& value) {
        clear();
        if (n > capacity_) {
            reserve(n);
        }
        for (size_t i = 0; i < n; ++i) {
            new(data + i) T(value);
        }
        size_ = n;
    }

    template<typename InputIt>
    void assign(InputIt first, InputIt last) {
        clear();
        for (; first != last; ++first) {
            push_back(*first);
        }
    }

    using iterator = T*;
    using const_iterator = const T*;

    iterator begin() { return data; }
    const_iterator begin() const { return data; }
    iterator end() { return data + size_; }
    const_iterator end() const { return data + size_; }

    iterator insert(const_iterator pos, const T& value) {
        size_t idx = pos - data;
        if (size_ == capacity_) {
            reserve(capacity_ == 0 ? 1 : capacity_ * 2);
        }

        for (size_t i = size_; i > idx; --i) {
            new(data + i) T(std::move(data[i-1]));
            (data + i - 1)->~T();
        }

        new(data + idx) T(value);
        ++size_;
        return data + idx;
    }

    iterator erase(const_iterator pos) {
        size_t idx = pos - data;
        if (idx >= size_) {
            return end();
        }

        (data + idx)->~T();

        for (size_t i = idx; i + 1 < size_; ++i) {
            new(data + i) T(std::move(data[i + 1]));
            (data + i + 1)->~T();
        }

        --size_;
        return data + idx;
    }

    template<typename... Args>
    void emplace_back(Args&&... args) {
        if (size_ == capacity_) {
            reserve(capacity_ == 0 ? 1 : capacity_ * 2);
        }
        new(data + size_) T(std::forward<Args>(args)...);
        ++size_;
    }

    T&       operator[](size_t i)       { return data[i]; }
    const T& operator[](size_t i) const { return data[i]; }

    T&       at(size_t i) {
        if (i >= size_) throw std::out_of_range("Vector::at");
        return data[i];
    }
    const T& at(size_t i) const {
        if (i >= size_) throw std::out_of_range("Vector::at");
        return data[i];
    }

    T& front() { return at(0); }
    const T& front() const { return at(0); }
    T& back() { return at(size_-1); }
    const T& back()  const { return at(size_-1); }

    // assign from initializer_list
    Vector& operator=(std::initializer_list<T> init) {
        assign(init.begin(), init.end());
        return *this;
    }

    template<typename U>
    friend bool operator< (const Vector<U>& a, const Vector<U>& b);
};

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

template<typename T>
bool operator< (const Vector<T>& a, const Vector<T>& b) {
    return std::lexicographical_compare(
        a.begin(), a.end(),
        b.begin(), b.end()
    );
}

template<typename T>
bool operator> (const Vector<T>& a, const Vector<T>& b) { return b < a; }

template<typename T>
bool operator<= (const Vector<T>& a, const Vector<T>& b) { return !(b < a); }

template<typename T>
bool operator>= (const Vector<T>& a, const Vector<T>& b) { return !(a < b); }

#endif // VECTOR_H
```

5.9 vector/source/functions.cpp File Reference

this file contains all functions

```
#include "functions.h"
#include <numeric>
#include <algorithm>
```

```
#include <iomanip>
#include <fstream>
#include <iostream>
#include <chrono>
#include <sstream>
#include <random>
#include "student.h"
#include "vector.h"
```

Functions

- void rusiuotiOutput (Vector< student > &grupe, char rusiavimoBudas, char galutinioBudas)
- void spausdinimasTerminale (const Vector< student > &grupe, char galutinioBudas)
- void spausdinimasFaile (const Vector< student > &grupe, char galutinioBudas)
- void generuotiFaila (int pKiekis, int studentuKiekis, const std::string &failoPavadinimas)
- void spausdinimas (char spausBudas, char rusiavimoBudas, char galutinioBudas, Vector< student > &grupe)
- void sortedStudentSpausdinimas (std::string lowGradeFailas, std::string highGradeFailas, Vector< student > &nepazangus, Vector< student > &normalus, char galutinioBudas)
- void nuskaitytiGeneruotusFailus (const std::string &failoPavadinimas, Vector< student > &grupe, int pKiekis, char galutinioBudas)
- void skirstytiStudentus (Vector< student > &grupe, Vector< student > &nepazangus, Vector< student > &normalus, char galutinioBudas)
- void pirmas (Vector< student > &grupe, char spausBudas, char rusiavimoBudas, char galutinioBudas, int pKiekis)
- void antras (Vector< student > &grupe, char spausBudas, char rusiavimoBudas, char galutinioBudas, int pKiekis)
- void trecias (Vector< student > &grupe, const Vector< std::string > &vardai, const Vector< std::string > &pavardes, char spausBudas, char rusiavimoBudas, char galutinioBudas, int pKiekis)
- void ketvirtas (Vector< student > &grupe, int pKiekis, char galutinioBudas)
- void penktas (int pKiekis)
- void sestas (Vector< student > &grupe, Vector< student > &testGrupe, Vector< student > &nepazangus, Vector< student > &normalus, char galutinioBudas, char rusiavimoBudas, int pKiekis)
- void septintas (char galutinioBudas)
- void astuntas ()

Variables

- double programosLaikas

5.9.1 Detailed Description

this file contains all functions

Author

Narbas

Version

v2.0

Date

2025-05-07

Copyright

Copyright (c) 2025

5.9.2 Function Documentation

5.9.2.1 antras()

```
void antras (
             Vector< student > & grupe,
             char spausBudas,
             char rusiavimoBudas,
             char galutinioBudas,
             int pKiekis)
```

5.9.2.2 astuntas()

```
void astuntas ()
```

5.9.2.3 generuotiFaila()

```
void generuotiFaila (
             int pKiekis,
             int studentuKiekis,
             const std::string & failoPavadinimas)
```

5.9.2.4 ketvirtas()

```
void ketvirtas (
             Vector< student > & grupe,
             int pKiekis,
             char galutinioBudas)
```

5.9.2.5 nuskaitytiGeneruotusFailus()

```
void nuskaitytiGeneruotusFailus (
             const std::string & failoPavadinimas,
             Vector< student > & grupe,
             int pKiekis,
             char galutinioBudas)
```

5.9.2.6 penktas()

```
void penktas (
             int pKiekis)
```

5.9.2.7 pirmas()

```
void pirmas (
             Vector< student > & grupe,
             char spausBudas,
             char rusiavimoBudas,
             char galutinioBudas,
             int pKiekis)
```

5.9.2.8 rusiuotiOutput()

```
void rusiuotiOutput (
            Vector< student > & grupe,
            char rusiavimoBudas,
            char galutinioBudas)
```

5.9.2.9 septintas()

```
void septintas (
            char galutinioBudas)
```

5.9.2.10 sestas()

```
void sestas (
            Vector< student > & grupe,
            Vector< student > & testGrupe,
            Vector< student > & nepazangus,
            Vector< student > & normalus,
            char galutinioBudas,
            char rusiavimoBudas,
            int pKiekis)
```

5.9.2.11 skirstytiStudentus()

```
void skirstytiStudentus (
            Vector< student > & grupe,
            Vector< student > & nepazangus,
            Vector< student > & normalus,
            char galutinioBudas)
```

5.9.2.12 sortedStudentSpausdinimas()

```
void sortedStudentSpausdinimas (
            std::string lowGradeFailas,
            std::string highGradeFailas,
            Vector< student > & nepazangus,
            Vector< student > & normalus,
            char galutinioBudas)
```

5.9.2.13 spausdinimas()

```
void spausdinimas (
            char spausBudas,
            char rusiavimoBudas,
            char galutinioBudas,
            Vector< student > & grupe)
```

5.9.2.14 spausdinimasFaile()

```
void spausdinimasFaile (
            const Vector< student > & grupe,
            char galutinioBudas)
```

5.9.2.15 spausdinimasTerminale()

```
void spausdinimasTerminale (
            const Vector< student > & grupe,
            char galutinioBudas)
```

5.9.2.16 trecias()

```
void trecias (
            Vector< student > & grupe,
            const Vector< std::string > & vardai,
            const Vector< std::string > & pavardes,
            char spausBudas,
            char rusiavimoBudas,
            char galutinioBudas,
            int pKiekis)
```

5.9.3 Variable Documentation

5.9.3.1 programosLaikas

```
double programosLaikas  [extern]
```

5.10 vector/source/main.cpp File Reference

This file handles program's flow.

```
#include <iostream>
#include <vector>
#include <ctime>
#include <limits>
#include <cstdlib>
#include <chrono>
#include <sstream>
#include <fstream>
#include <algorithm>
#include "../include/student.h"
#include "../include/functions.h"
#include <iomanip>
#include "../include/utils.h"
#include "../include/vector.h"
```

Functions

- int main ()

Variables

- double programosLaikas = 0.0

5.10.1 Detailed Description

This file handles program's flow.

Author

Narbas

Version

v2.0

Date

2025-05-07

Copyright

Copyright (c) 2025

5.10.2 Function Documentation

5.10.2.1 main()

```
int main ()
```

5.10.3 Variable Documentation

5.10.3.1 programosLaikas

```
double programosLaikas = 0.0
```

5.11 vector/source/student.cpp File Reference

this file contains class methods implementation

```
#include "student.h"
#include <algorithm>
#include <numeric>
#include <iomanip>
```

Functions

- std::ostream & operator<< (std::ostream &os, const student &studentas)
- std::istream & operator>> (std::istream &in, student &studentas)

5.11.1 Detailed Description

this file contains class methods implementation

Author

Narbas

Version

v2.0

Date

2025-05-07

Copyright

Copyright (c) 2025

5.11.2 Function Documentation

5.11.2.1 operator<<()

```
std::ostream & operator<< (
            std::ostream & os,
            const student & studentas)
```

5.11.2.2 operator>>()

```
std::istream & operator>> (
            std::istream & in,
            student & studentas)
```

Index