# Studentai ir jų pažymiai

Sugeneruota Doxygen 1.11.0

**1 VU_OP_3uzd-MS** 1
1.1 v.pradinė . . . 1
1.1.1 v.pradine.cpp . . . 1
1.2 v0.1 . . . 1
1.2.1 cmasyvas.cpp . . . 1
1.2.2 vector.cpp . . . 1
1.3 v0.2 . . . 1
1.4 v0.3 . . . 2
1.5 v0.4 . . . 2
1.6 v1.0 . . . 2
1.7 v1.1 . . . 2
1.8 v1.2 . . . 2
1.9 v1.5 . . . 3
1.10 v2.0 . . . 3
1.10.1 instrukcija kaip paleisti programą UNIX(MacOS, Linux,...) sistemoje . . . 3

**2 Hierarchijos Indeksas** 5
2.1 Klasių hierarchija . . . 5

**3 Klasės Indeksas** 7
3.1 Klasės . . . 7

**4 Failo Indeksas** 9
4.1 Failai . . . 9

**5 Klasės Dokumentacija** 11
5.1 Studentas Klasė . . . 11
5.1.1 Konstruktoriaus ir Destruktoriaus Dokumentacija . . . 12
5.1.1.1 Studentas() [1/5] . . . 12
5.1.1.2 Studentas() [2/5] . . . 12
5.1.1.3 Studentas() [3/5] . . . 12
5.1.1.4 Studentas() [4/5] . . . 13
5.1.1.5 Studentas() [5/5] . . . 13
5.1.1.6 ∼Studentas() . . . 13
5.1.2 Metodų Dokumentacija . . . 13
5.1.2.1 clearEverything() . . . 13
5.1.2.2 getEgz() . . . 13
5.1.2.3 getMediana() . . . 13
5.1.2.4 getNd() . . . 13
5.1.2.5 getVidurkis() . . . 14
5.1.2.6 lastNdtoEgz() . . . 14
5.1.2.7 ndAppend() . . . 14
5.1.2.8 ndResize() . . . 14
5.1.2.9 ndSk() . . . 14

5.1.2.10 operator"!=() . . . 14
5.1.2.11 operator=() [1/2] . . . 14
5.1.2.12 operator=() [2/2] . . . 15
5.1.2.13 operator==() . . . 15
5.1.2.14 setEgz() . . . 15
5.1.2.15 setNd() . . . 15
5.1.2.16 skaiciuotiMed() . . . 15
5.1.2.17 skaiciuotiVid() . . . 16
5.1.2.18 whoAmI() . . . 16
5.1.3 Draugiškų Ir Susijusių Funkcijų Dokumentacija . . . 16
5.1.3.1 operator<< [1/2] . . . 16
5.1.3.2 operator<< [2/2] . . . 16
5.1.3.3 operator>> [1/2] . . . 17
5.1.3.4 operator>> [2/2] . . . 18
5.2 Timer Klasė . . . 18
5.2.1 Konstruktoriaus ir Destruktoriaus Dokumentacija . . . 18
5.2.1.1 Timer() . . . 18
5.2.2 Metodų Dokumentacija . . . 19
5.2.2.1 elapsed() . . . 19
5.2.2.2 reset() . . . 19
5.3 Vector< T > Klasė Šablonas . . . 19
5.3.1 Konstruktoriaus ir Destruktoriaus Dokumentacija . . . 20
5.3.1.1 Vector() [1/5] . . . 20
5.3.1.2 Vector() [2/5] . . . 20
5.3.1.3 Vector() [3/5] . . . 20
5.3.1.4 Vector() [4/5] . . . 20
5.3.1.5 Vector() [5/5] . . . 20
5.3.1.6 ∼Vector() . . . 20
5.3.2 Metodų Dokumentacija . . . 21
5.3.2.1 assign() . . . 21
5.3.2.2 at() [1/2] . . . 21
5.3.2.3 at() [2/2] . . . 21
5.3.2.4 back() [1/2] . . . 21
5.3.2.5 back() [2/2] . . . 21
5.3.2.6 begin() [1/2] . . . 22
5.3.2.7 begin() [2/2] . . . 22
5.3.2.8 capacity() . . . 22
5.3.2.9 clear() . . . 22
5.3.2.10 emplace() . . . 22
5.3.2.11 emplace_back() . . . 22
5.3.2.12 empty() . . . 23
5.3.2.13 end() [1/2] . . . 23

5.3.2.14 end() [2/2] . . . 23
5.3.2.15 front() [1/2] . . . 23
5.3.2.16 front() [2/2] . . . 23
5.3.2.17 insert() . . . 23
5.3.2.18 insert_range() . . . 24
5.3.2.19 operator"!=() . . . 24
5.3.2.20 operator=() [1/2] . . . 24
5.3.2.21 operator=() [2/2] . . . 24
5.3.2.22 operator==() . . . 25
5.3.2.23 operator[]() [1/2] . . . 25
5.3.2.24 operator[]() [2/2] . . . 25
5.3.2.25 pop_back() . . . 25
5.3.2.26 push_back() [1/2] . . . 25
5.3.2.27 push_back() [2/2] . . . 25
5.3.2.28 reserve() . . . 26
5.3.2.29 resize() [1/2] . . . 26
5.3.2.30 resize() [2/2] . . . 26
5.3.2.31 shrink_to_fit() . . . 26
5.3.2.32 size() . . . 26
5.4 Zmogus Klasė . . . 27
5.4.1 Konstruktoriaus ir Destruktoriaus Dokumentacija . . . 27
5.4.1.1 Zmogus() [1/2] . . . 27
5.4.1.2 Zmogus() [2/2] . . . 27
5.4.1.3 ∼Zmogus() . . . 27
5.4.2 Metodų Dokumentacija . . . 27
5.4.2.1 getPavarde() . . . 27
5.4.2.2 getVardas() . . . 28
5.4.2.3 setPavarde() . . . 28
5.4.2.4 setVardas() . . . 28

**6 Failo Dokumentacija** **29**
6.1 pazymiai.h . . . 29
6.2 timer.h . . . 30
6.3 vector.h . . . 30

**Rodyklė** **33**

# skyrius 1

# VU_OP_3uzd-MS

## 1.1 v.pradinė

### 1.1.1 v.pradine.cpp

Studentų ir nd kiekis "hard-coded"\ Naudojami c masyvai\ Pasirenkama tarp 3 būdų įvesti duomenis:

- Įvesti duomenis ranka
- Generuoti pažymius
- Generuoti pažymius ir vardus

## 1.2 v0.1

Studentų ir nd kiekis dinamiškas\ Masyvų duomenų spauzdinimas\ Pasirinkimas tarp vidurkio ir medianos skaičiavimo

### 1.2.1 cmasyvas.cpp

Naudojami tik c masyvai

### 1.2.2 vector.cpp

Naudojami <vector> objektai

## 1.3 v0.2

Galimybė duomenis įvesti per failą\ Išspauzdinamų studnetų kiekio pasirinkimas\ Rikiavimas pagal skirtingus parametrus

## 1.4 v0.3

Programa paskirstyta į kelis .cpp failus\ Sukurtas programą "apjungentis", struktūras apibreženti header failas\ Naudojami try-catch blokai vartotojo įvesčiai (sukurta bendra funkcija tai įgyvendinti)

## 1.5 v0.4

Duomenų (failų) generavimas\ Sugeneruotų studentų išvedimas į du skirtingus konteinerius (geri/blogi)\ Bendras programos veikimo laikas, kai generuojami 5 failai su nurodytasi dydžiais (15Nd): 118.9s (5 testų vidurkis)

## 1.6 v1.0

trijų tipų konteineriai\ pritaikytos trys strategijos\ Atlikta spartumo diagnostika\ "..." Laiko tarpas didesnis negu 10min

Processor Intel Core i7-11800H 2.30 GHz\ Installed RAM 16,0 GB (15,8 GB usable)\ System type 64-bit Ubuntu 22.04.3 LTS on Windows 10 x86_64\ Storage 512GB SSD

Instrukcija kaip paleisti koda UNIX sistemoje:

1. Atidarę sistemos komandinę eilutę šiame aplanke įveskite `nano pazymiai.h`, modifikuokite failą taip, kad pirmos trys programos eilutės atrodytų kaip vienas iš pateiktų pavyzdžių:\ Vector versijai:
   ```
   #ifndef Container
   #define Container std::vector
   #endif
   ```
   List versijai:
   ```
   #ifndef Container
   #define Container std::list
   #endif
   ```
   Deque versijai:
   ```
   #ifndef Container
   #define Container std::deque
   #endif
   ```
   Uždarykite teksto retdaktorių su "Ctrl" ir "C" mygtukais vienu metu, išsaugokite failą (spauskite "Y" ir ENTER)\
2. Sistemos komandinėje eilutėje įrašykite `make`
3. Kad paleistumėte programą į komadinę eilutę įveskite: `./pazymiai` ir sekite instrukcijas programoje.

## 1.7 v1.1

Vietoje struktūros naudojama klasė.\

norint sukompiliuoti paleidžiamus failus su skirtingoms optimizavimo vėliavoms:\ `make O1`, `make O2`, `make O3`

## 1.8 v1.2

Implementuotos "rule of five" funkcijos\ Realizuotas kodas, kuris patikrina naujų operatorių, konstruktorių veikimą\ Įvestis:\ Duomenų įvestis rankiniu ir automatiniu būdu įvyksta, kai duomenų skaitymui yra naudojamas `istream` objektas, o tada žiūrima pagal vartotojo pasirinkta programos veikimą (globalų kintamąjį `input↩Option`).\ Duomenų įvestis iš failo įvyksta, kai duomenų skaitymui naudojamas `istringstream` objektas.

Išvesties:\ Duomenų išvedimas per konsolę įvyksta, kai duomenų išvedimui yra naudojamas `ostream` objektas. Duomenų išvedimas į failą įvyksta, kai duomenų išvedimui yra naudojamas `ofstream` objektas.

## 1.9 v1.5

Sukurta abstrakti Zmogus klasė\ Perdaryti konstruktoriai\

## 1.10 v2.0

Naudojamas Google test framework'as\ Parašyti porą "testinių" testų\

### 1.10.1 instrukcija kaip paleisti programą UNIX(MacOS, Linux,...) sistemoje

1. Sistemos komandinėje eilutėje įrašykite `make`, ENTER
2. Kad paleistumėte programą, į komadinę eilutę įveskite: `./pazymiai` ir sekite instrukcijas programoje. Kad paleistumėte programos testą, į komadinę eilutę įveskite: `./test_pazymiai`.

# skyrius 2

# Hierarchijos Indeksas

## 2.1 Klasių hierarchija

Šis paveldėjimo sąrašas yra beveik surikiuotas abėcėlės tvarka:


Timer . . . . . 18
Vector< T > . . . . . 19
Zmogus . . . . . 27
    Studentas . . . . . 11

# skyrius 3

# Klasės Indeksas

## 3.1 Klasės

Klasės, struktūros, sąjungos ir sąsajos su trumpais aprašymais:


Studentas . . . . . 11
Timer . . . . . 18
Vector< T > . . . . . 19
Zmogus . . . . . 27

# skyrius 4

# Failo Indeksas

## 4.1 Failai

Visų dokumentuotų failų sąrašas su trumpais aprašymais:

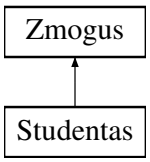
pazymiai.h . . . . . 29
timer.h . . . . . 30
vector.h . . . . . 30

# skyrius 5

# Klasės Dokumentacija

## 5.1 Studentas Klasė

Paveldimumo diagrama Studentas:




**Vieši Metodai**

- Studentas (const string &vardas, const string &pavarde)
- Studentas (const string &vardas, const string &pavarde, int egz, double vidurkis, double mediana, vector< int > nd)
- Studentas (const Studentas &tmpStud)
- Studentas (Studentas &&tmpStud) noexcept
- void ndAppend (int balas)
- void ndResize (int size)
- int ndSk () const
- int getNd (int index)
- void setNd (int index, int value)
- void lastNdtoEgz ()
- void setEgz (int value)
- double getEgz () const
- double getVidurkis () const
- double getMediana () const
- void skaiciuotiVid ()
- void skaiciuotiMed ()
- void clearEverything ()
- Studentas & operator= (const Studentas &tmpStud)
- Studentas & operator= (Studentas &&tmpStud)
- bool operator== (const Studentas &other) const
- bool operator!= (const Studentas &other) const
- virtual void whoAmI ()

## Vieši Metodai inherited from Zmogus

- Zmogus (const string &vard, const string &pavard)
- string getVardas () const
- string getPavarde () const
- void setVardas (string vard)
- void setPavarde (string pav)

### Draugai

- istringstream & operator>> (istringstream &filename, Studentas &tmpStud)
- istream & operator>> (istream &manual, Studentas &tmpStud)
- ostream & operator<< (ostream &console, const Studentas &tmpStud)
- ofstream & operator<< (ofstream &filename, const Studentas &tmpStud)

### Additional Inherited Members

## Apsaugoti Atributai inherited from Zmogus

- string **vardas**
- string **pavarde**

### 5.1.1 Konstruktoriaus ir Destruktoriaus Dokumentacija

#### 5.1.1.1 Studentas() [1/5]

```
Studentas::Studentas ()
00012                     : Zmogus(),
00013     egz(0),
00014     nd({}),
00015     vidurkis(0.0),
00016     mediana(0.0) {}
```

#### 5.1.1.2 Studentas() [2/5]

```
Studentas::Studentas (
              const string & vardas,
              const string & pavarde)
00018                                                          : Zmogus(vardas, pavarde) {
00019     egz = 0;
00020     nd = {};
00021     vidurkis = 0.0;
00022     mediana = 0.0;
00023 }
```

#### 5.1.1.3 Studentas() [3/5]

```
Studentas::Studentas (
              const string & vardas,
              const string & pavarde,
              int egz,
              double vidurkis,
              double mediana,
              vector< int > nd)  [inline]
00043
      : Zmogus(vardas, pavarde),
00044             egz(egz), vidurkis(vidurkis), mediana(mediana), nd(nd) {}
```

#### 5.1.1.4 Studentas() [4/5]

```
Studentas::Studentas (
            const Studentas & tmpStud)
00025                                       : Zmogus(tmpStud.vardas, tmpStud.pavarde) {
00026     nd = tmpStud.nd;
00027     egz = tmpStud.egz;
00028     mediana = tmpStud.mediana;
00029     vidurkis = tmpStud.vidurkis;
00030     // cout « "Kopijavimo konstruktorius suveike" « endl;
00031 }
```

#### 5.1.1.5 Studentas() [5/5]

```
Studentas::Studentas (
            Studentas && tmpStud)  [noexcept]
00033                                          : Zmogus(move(tmpStud.vardas), move(tmpStud.pavarde))
      {
00034     nd = move(tmpStud.nd);
00035     egz = move(tmpStud.egz);
00036     mediana = move(tmpStud.mediana);
00037     vidurkis = move(tmpStud.vidurkis);
00038
00039     tmpStud.clearEverything();
00040     // cout « "Perkelimo konstruktorius suveike" « endl;
00041 }
```

#### 5.1.1.6 ~Studentas()

```
Studentas::~Studentas ()  [inline]
00065                     {
00066             nd.clear();
00067         }
```

### 5.1.2 Metodų Dokumentacija

#### 5.1.2.1 clearEverything()

```
void Studentas::clearEverything ()  [inline]
00069 { this->vardas.clear(); this->pavarde.clear(); this->nd.clear(); this->egz = 0; this->mediana = 0.0;
      this->vidurkis = 0.0; }
```

#### 5.1.2.2 getEgz()

```
double Studentas::getEgz () const  [inline]
00057 {return egz;}
```

#### 5.1.2.3 getMediana()

```
double Studentas::getMediana () const  [inline]
00061 {return mediana;}
```

#### 5.1.2.4 getNd()

```
int Studentas::getNd (
            int index)  [inline]
00051 {return nd.at(index);}
```

#### 5.1.2.5 getVidurkis()

```
double Studentas::getVidurkis () const  [inline]
00060 {return vidurkis;}
```

#### 5.1.2.6 lastNdtoEgz()

```
void Studentas::lastNdtoEgz ()
00267                                {
00268     int lastIndex = (int)this->nd.size() - 1;
00269     egz = nd.at(lastIndex);
00270     nd.resize(lastIndex);
00271 }
```

#### 5.1.2.7 ndAppend()

```
void Studentas::ndAppend (
             int balas)  [inline]
00048 {nd.push_back(balas);} //prideda viena namu darba.
```

#### 5.1.2.8 ndResize()

```
void Studentas::ndResize (
             int size)  [inline]
00049 {nd.resize(size);}
```

#### 5.1.2.9 ndSk()

```
int Studentas::ndSk () const  [inline]
00050 {return nd.size();}
```

#### 5.1.2.10 operator"!=()

```
bool Studentas::operator!= (
             const Studentas & other) const
00242                                                    {
00243     return !(*this == other);
00244 }
```

#### 5.1.2.11 operator=() [1/2]

```
Studentas & Studentas::operator= (
             const Studentas & tmpStud)
00043                                                     {
00044     if(this != &tmpStud){
00045         vardas = tmpStud.vardas;
00046         pavarde = tmpStud.pavarde;
00047         nd = tmpStud.nd;
00048         egz = tmpStud.egz;
00049         mediana = tmpStud.mediana;
00050         vidurkis = tmpStud.vidurkis;
00051     }
00052     // cout « "Priskyrimo operatorius suveike" « endl;
00053     return *this;
00054 }
```

#### 5.1.2.12 operator=() [2/2]

```
Studentas & Studentas::operator= (
            Studentas && tmpStud)
00056                                                   {
00057     if(this != &tmpStud) {
00058         vardas = move(tmpStud.vardas);
00059         pavarde = move(tmpStud.pavarde);
00060         nd = move(tmpStud.nd);
00061         egz = move(tmpStud.egz);
00062         mediana = move(tmpStud.mediana);
00063         vidurkis = move(tmpStud.vidurkis);
00064         tmpStud.clearEverything();
00065     }
00066     // cout << "Perkelimo operatorius suveike" << endl;
00067     return *this;
00068 }
```

#### 5.1.2.13 operator==()

```
bool Studentas::operator== (
            const Studentas & other) const
00234                                                       {
00235     if (nd.size() != other.ndSk()) return false;
00236     for(int i = 0; i < nd.size(); i++) {
00237         if(nd[i] != other.nd.at(i)) return false;
00238     }
00239     return vardas == other.getVardas() && pavarde == other.getPavarde() && egz == other.getEgz();
00240 }
```

#### 5.1.2.14 setEgz()

```
void Studentas::setEgz (
            int value)  [inline]
00056 {egz = value;}
```

#### 5.1.2.15 setNd()

```
void Studentas::setNd (
            int index,
            int value)
00207                                           {
00208     if(index > abs((int)this->nd.size())) throw out_of_range("Bondote nustatyti nd elementą už jo
    ribų");
00209     else if(index >= 0) this->nd.at(index) = value;
00210     else this->nd.at((int)this->nd.size() + index) = value;
00211 }
```

#### 5.1.2.16 skaiciuotiMed()

```
void Studentas::skaiciuotiMed ()
00222                                {
00223     int ndSk = nd.size();
00224     double vidurys;
00225     vector<int> ndCopy(ndSk);
00226     ndCopy = nd;
00227     sort(ndCopy.begin(), ndCopy.end());
00228     vidurys = ndSk % 2 == 0 ? (ndCopy[ndSk / 2 - 1] + ndCopy[ndSk / 2]) / 2.0 : ndCopy[ndSk / 2];
00229     vidurys = round(vidurys * 100.0) / 100.0;
00230     vidurys = vidurys * 0.4 + egz * 0.6;
00231     mediana = vidurys;
00232 }
```

#### 5.1.2.17 skaiciuotiVid()

```
void Studentas::skaiciuotiVid ()
00213                             {
00214     double agreguotas = 0;
00215     for(auto &balas : nd) agreguotas += balas;
00216     agreguotas /= (double)nd.size();
00217     agreguotas = agreguotas * 0.4 + egz * 0.6;
00218     agreguotas = round(agreguotas * 100.0) / 100.0; // iki simtuju
00219     vidurkis = agreguotas;
00220 }
```

#### 5.1.2.18 whoAmI()

```
virtual void Studentas::whoAmI ()  [inline], [virtual]
```

Realizuoja Zmogus.

```
00081 {cout « "Studentas klasė" « endl;}
```

### 5.1.3 Draugiškų Ir Susijusių Funkcijų Dokumentacija

#### 5.1.3.1 operator<< [1/2]

```
ofstream & operator<< (
             ofstream & filename,
             const Studentas & tmpStud)  [friend]
00196                                                          {
00197     // stringstream RF;
00198     filename « left « setw(24) « tmpStud.getVardas() « setw(24) « tmpStud.getPavarde() «
00199     setw(10) « fixed « setprecision(2) « tmpStud.getVidurkis() « fixed « setw(10) «
      tmpStud.getMediana();
00200
00201     //cout « "As esu isvedimo i faila operatoriuje «" « endl;
00202     // filename « RF.str();
00203     // RF.clear();
00204     return filename;
00205 }
```

#### 5.1.3.2 operator<< [2/2]

```
ostream & operator<< (
             ostream & console,
             const Studentas & tmpStud)  [friend]
00188                                                        {
00189     console « left « setw(24) « tmpStud.getVardas() « setw(24) « tmpStud.getPavarde() «
00190     setw(10) « fixed « setprecision(2) « tmpStud.getVidurkis() « fixed « setw(10) «
      tmpStud.getMediana();
00191     //cout « "As esu isvedimo i konsole operatoriuje «" « endl;
00192     return console;
00193
00194 }
```

#### 5.1.3.3 operator>> [1/2]

```
istream & operator>> (
            istream & manual,
            Studentas & tmpStud)  [friend]
00070                                                                {
00071     string line = "";
00072     tmpStud.ndResize(ndSk);
00073     if (inputOption == 2 || inputOption == 3) {
00074         cout « endl;
00075         int i = 0;
00076         while(true) {
00077             tmpStud.setNd(i, 1 + rand() % 11);
00078             cout « "Sugeneruotas " « i+1 « "-as namų darbas" « endl;
00079             i++;
00080             if (i == ndSk) {
00081                 if(taipArNe("\nPridėti dar vieną namų darbą? (ENTER - Taip, 'Ne'/'N' - Ne): ")) break;
00082                 ndSk += 1;
00083                 tmpStud.ndAppend(0);
00084             }
00085         }
00086         tmpStud.setEgz(rand() % 11);
00087
00088     }
00089     if(inputOption == 1 || inputOption == 2) {
00090         string line;
00091
00092         cout « endl;
00093         cout « "Vardas: ";
00094         manual » line;
00095         tmpStud.setVardas(line);
00096
00097         cout « "Pavarde: ";
00098         manual » line;
00099         tmpStud.setPavarde(line);
00100     }
00101
00102     if (inputOption == 1) {
00103         int i = 0;
00104         while (true) {
00105             int grade;
00106             while(true) {
00107                 try {
00108                     cout « "\n" « i + 1 « " ND įvertinimas (0-10): ";
00109                     manual » grade;
00110                     if(!cin.good() || grade < 1 || grade > 10) {
00111                         throw invalid_argument("Netinkama įvestis. Įveskite skaičių nuo 1 iki 10");
00112                     }
00113                     tmpStud.setNd(i, grade);
00114                     break;
00115                 } catch(invalid_argument &e) {
00116                     cerr « e.what() « endl;
00117                     cin.clear();
00118                     cin.ignore(numeric_limits<streamsize>::max(), '\n');
00119                     continue;
00120                 }
00121             }
00122             i++;
00123             if (i < ndSk) continue;
00124             else if(!taipArNe("Pridėti dar vieną namų darbą? (ENTER - Taip, 'Ne'/'N' - Ne): ")) {
00125                 ndSk++;
00126                 tmpStud.ndResize(ndSk);
00127                 continue;
00128             } else {
00129                 while (true) {
00130                     try {
00131                         cout « "Egzamino įvertinimas: ";
00132                         manual » grade;
00133                         if(!manual.good() || grade < 1 || grade > 10) {
00134                             throw invalid_argument("Netinkama įvestis. Įveskite skaičių nuo 1 iki
      10");
00135                         }
00136                         tmpStud.setEgz(grade);
00137                         break;
00138                     } catch(invalid_argument &e) {
00139                         cerr « e.what() « endl;
00140                         manual.clear();
00141                         manual.ignore(numeric_limits<streamsize>::max(), '\n');
00142                         continue;
00143                     }
00144                 }
00145                 break;
00146             }
00147         }
00148     }
```

```
00149
00150     if (inputOption == 3) {
00151         tmpStud.setVardas("Vardenis_" + to_string(studSk));
00152         tmpStud.setPavarde("Pavardenis_" + to_string(studSk));
00153
00154         cout « "Vardas bei pavardė sugeneruoti" « endl;
00155     }
00156
00157     tmpStud.skaiciuotiVid();
00158     tmpStud.skaiciuotiMed();
00159
00160     return manual;
00161 }
```

#### 5.1.3.4 operator>> [2/2]

```
istringstream & operator>> (
             istringstream & filename,
             Studentas & tmpStud)  [friend]
00163                                                                    {
00164     string vardas, pavarde;
00165     int ndSk;
00166
00167     if (!(filename » vardas » pavarde)) {
00168         cerr « "Nepavyko nuskaityti vardo ir pavardes" « endl;
00169     }
00170
00171     tmpStud.setVardas(vardas);
00172     tmpStud.setPavarde(pavarde);
00173
00174     int pazymys;
00175     while (filename » pazymys) {
00176         tmpStud.ndAppend(pazymys);
00177     }
00178
00179     tmpStud.lastNdtoEgz();
00180
00181
00182     tmpStud.skaiciuotiVid();
00183     tmpStud.skaiciuotiMed();
00184     //cout « "As esu ivedimo is failo operatoriuje »" « endl;
00185     return filename;
00186 }
```

Dokumentacija šiai klasei sugeneruota iš šių failų:

- pazymiai.h
- StudentaiClass.cpp

## 5.2 Timer Klasė

### Vieši Metodai

- void reset ()
- double elapsed () const

### 5.2.1 Konstruktoriaus ir Destruktoriaus Dokumentacija

#### 5.2.1.1 Timer()

```
Timer::Timer ()  [inline]
00006 : start{std::chrono::high_resolution_clock::now()} {}
```

### 5.2.2 Metodų Dokumentacija

#### 5.2.2.1 elapsed()

```
double Timer::elapsed () const  [inline]
00010                           {return
      std::chrono::duration<double>(std::chrono::high_resolution_clock::now() - start).count();
00011     }
```

#### 5.2.2.2 reset()

```
void Timer::reset ()  [inline]
00007                 {
00008       start = std::chrono::high_resolution_clock::now();
00009     }
```

Dokumentacija šiai klasei sugeneruota iš šio failo:

- timer.h

## 5.3 Vector< T > Klasė Šablonas

**Vieši Metodai**

- Vector (size_t initial_size)
- Vector (std::initializer_list< T > init)
- Vector (const Vector &other)
- Vector (Vector &&other) noexcept
- Vector & operator= (const Vector &other)
- Vector & operator= (Vector &&other) noexcept
- T & operator[ ] (size_t index)
- const T & operator[ ] (size_t index) const
- T & at (size_t index)
- const T & at (size_t index) const
- T & front ()
- const T & front () const
- T & back ()
- const T & back () const
- T * begin () noexcept
- const T * begin () const noexcept
- T * end () noexcept
- const T * end () const noexcept
- bool empty () const noexcept
- size_t size () const noexcept
- void reserve (size_t new_cap)
- size_t capacity () const noexcept
- void shrink_to_fit ()
- void assign (size_t n, const T &value)
- void clear () noexcept
- void push_back (const T &value)
- void push_back (T &&value)
- void pop_back ()
- void resize (size_t count)
- void resize (size_t count, const T &value)
- void insert (size_t index, const T &value)
- void insert_range (size_t index, size_t n, const T &value)
- void emplace (size_t index, T &&value)
- void emplace_back (T &&value)
- bool operator== (const Vector &other) const
- bool operator!= (const Vector &other) const

### 5.3.1 Konstruktoriaus ir Destruktoriaus Dokumentacija

#### 5.3.1.1 Vector() [1/5]

```
template<typename T >
Vector< T >::Vector ()
00017 : data_(nullptr), size_(0), capacity_(0) {};
```

#### 5.3.1.2 Vector() [2/5]

```
template<typename T >
Vector< T >::Vector (
            size_t initial_size)  [explicit]
00021     : size_(initial_size), capacity_(nextCapacity(initial_size)), data_(new
      T[nextCapacity(initial_size)]) {
00022     for (size_t i = 0; i < size_; i++) data_[i] = T();
00023 }
```

#### 5.3.1.3 Vector() [3/5]

```
template<typename T >
Vector< T >::Vector (
            std::initializer_list< T > init)
00027     : size_(init.size()), capacity_(nextCapacity(init.size())), data_(new
      T[nextCapacity(init.size())]) {
00028     std::copy(init.begin(), init.end(), data_);
00029 }
```

#### 5.3.1.4 Vector() [4/5]

```
template<typename T >
Vector< T >::Vector (
            const Vector< T > & other)
00033     : size_(other.size_), capacity_(other.capacity_), data_(new T[other.capacity_]) {
00034     std::copy(other.data_, other.data_ + other.size_, data_);
00035 }
```

#### 5.3.1.5 Vector() [5/5]

```
template<typename T >
Vector< T >::Vector (
            Vector< T > && other)  [noexcept]
00039     : data_(other.data_), size_(other.size_), capacity_(other.capacity_) {
00040     other.data_ = nullptr;
00041     other.size_ = 0;
00042     other.capacity_ = 0;
00043 }
```

#### 5.3.1.6 ~Vector()

```
template<typename T >
Vector< T >::~Vector ()
00046                   {
00047     delete[] data_;
00048 }
```

## 5.3.2 Metodų Dokumentacija

### 5.3.2.1 assign()

```
template<typename T >
void Vector< T >::assign (
             size_t n,
             const T & value)
00225                                                {
00226     if (n > capacity_) reserve(n);
00227     for (size_t i = 0; i < n; i++) {
00228         data_[i] = value;
00229     }
00230     size_ = n;
00231 }
```

### 5.3.2.2 at() [1/2]

```
template<typename T >
T & Vector< T >::at (
             size_t index)
00064                              {
00065     if (index >= size_) {
00066         throw std::out_of_range("Index out of range");
00067     }
00068     return data_[index];
00069 }
```

### 5.3.2.3 at() [2/2]

```
template<typename T >
const T & Vector< T >::at (
             size_t index) const
00072                                           {
00073     if (index >= size_) {
00074         throw std::out_of_range("Index out of range");
00075     }
00076     return data_[index];
00077 }
```

### 5.3.2.4 back() [1/2]

```
template<typename T >
T & Vector< T >::back ()
00090                       {
00091     return data_[size_ - 1];
00092 }
```

### 5.3.2.5 back() [2/2]

```
template<typename T >
const T & Vector< T >::back () const
00095                                {
00096     return data_[size_ - 1];
00097 }
```

#### 5.3.2.6 begin() [1/2]

```
template<typename T >
const T * Vector< T >::begin () const  [noexcept]
00119 {return data_;}
```

#### 5.3.2.7 begin() [2/2]

```
template<typename T >
T * Vector< T >::begin ()  [noexcept]
00117 {return data_;}
```

#### 5.3.2.8 capacity()

```
template<typename T >
size_t Vector< T >::capacity () const  [noexcept]
00110                                        {
00111     return capacity_;
00112 }
```

#### 5.3.2.9 clear()

```
template<typename T >
void Vector< T >::clear ()  [noexcept]
00201                                {
00202     size_ = 0;
00203 }
```

#### 5.3.2.10 emplace()

```
template<typename T >
void Vector< T >::emplace (
             size_t index,
             T && value)
00256                                                {
00257     if (index > size_) throw std::out_of_range("Index out of range");
00258     if (size_ == capacity_) reserve(nextCapacity(size_ + 1));
00259
00260     for (size_t i = size_; i > index; --i) {
00261         data_[i] = std::move(data_[i - 1]);
00262     }
00263     data_[index] = std::move(value);
00264     size_++;
00265 }
```

#### 5.3.2.11 emplace_back()

```
template<typename T >
void Vector< T >::emplace_back (
             T && value)
00268                                       {
00269     if (size_ == capacity_) reserve(nextCapacity(size_ + 1));
00270     data_[size_++] = std::move(value);
00271 }
```

#### 5.3.2.12 empty()

```
template<typename T >
bool Vector< T >::empty () const  [noexcept]
00105                                    {
00106     return size_ == 0;
00107 }
```

#### 5.3.2.13 end() [1/2]

```
template<typename T >
const T * Vector< T >::end () const  [noexcept]
00123 {return data_ + size_;}
```

#### 5.3.2.14 end() [2/2]

```
template<typename T >
T * Vector< T >::end ()  [noexcept]
00121 {return data_ + size_;}
```

#### 5.3.2.15 front() [1/2]

```
template<typename T >
T & Vector< T >::front ()
00080                    {
00081     return data_[0];
00082 }
```

#### 5.3.2.16 front() [2/2]

```
template<typename T >
const T & Vector< T >::front () const
00085                              {
00086     return data_[0];
00087 }
```

#### 5.3.2.17 insert()

```
template<typename T >
void Vector< T >::insert (
             size_t index,
             const T & value)
00234                                                {
00235     if (size_ == capacity_) reserve(nextCapacity(size_ + 1));
00236     for (size_t i = size_; i > index; i--) {
00237         data_[i] = data_[i - 1];
00238     }
00239     data_[index] = value;
00240     size_++;
00241 }
```

#### 5.3.2.18 insert_range()

```
template<typename T >
void Vector< T >::insert_range (
            size_t index,
            size_t n,
            const T & value)
00244                                                                    {
00245     if (size_ + n > capacity_) reserve(nextCapacity(size_ + n));
00246     for (size_t i = size_ + n - 1; i > index + n - 1; i--) {
00247         data_[i] = data_[i - n];
00248     }
00249     for (size_t i = index; i < index + n; i++) {
00250         data_[i] = value;
00251     }
00252     size_ += n;
00253 }
```

#### 5.3.2.19 operator"!=()

```
template<typename T >
bool Vector< T >::operator!= (
            const Vector< T > & other) const
00282                                                  {
00283     return !(*this == other);
00284 }
```

#### 5.3.2.20 operator=() [1/2]

```
template<typename T >
Vector< T > & Vector< T >::operator= (
            const Vector< T > & other)
00127                                                  {
00128     if (this != &other) {
00129         if (other.size_ > capacity_) {
00130             T* new_data = new T[nextCapacity(other.size_)];
00131             std::copy(other.data_, other.data_ + other.size_, new_data);
00132             delete[] data_;
00133             data_ = new_data;
00134             capacity_ = nextCapacity(other.size_);
00135         } else {
00136             std::copy(other.data_, other.data_ + other.size_, data_);
00137         }
00138         size_ = other.size_;
00139     }
00140     return *this;
00141 }
```

#### 5.3.2.21 operator=() [2/2]

```
template<typename T >
Vector< T > & Vector< T >::operator= (
            Vector< T > && other)  [noexcept]
00145                                                       {
00146     if (this != &other) {
00147         delete[] data_;
00148         data_ = other.data_;
00149         size_ = other.size_;
00150         capacity_ = other.capacity_;
00151
00152         other.data_ = nullptr;
00153         other.size_ = 0;
00154         other.capacity_ = 0;
00155     }
00156     return *this;
00157 }
```

#### 5.3.2.22 operator==()

```
template<typename T >
bool Vector< T >::operator== (
            const Vector< T > & other) const
00274                                                   {
00275     if (size_ != other.size_) {
00276         return false;
00277     }
00278     return std::equal(data_, data_ + size_, other.data_);
00279 }
```

#### 5.3.2.23 operator[]() [1/2]

```
template<typename T >
T & Vector< T >::operator[] (
            size_t index)
00054                                   {
00055     return data_[index];
00056 }
```

#### 5.3.2.24 operator[]() [2/2]

```
template<typename T >
const T & Vector< T >::operator[] (
            size_t index) const
00059                                                   {
00060     return data_[index];
00061 }
```

#### 5.3.2.25 pop_back()

```
template<typename T >
void Vector< T >::pop_back ()
00207                               {
00208     if (size_ > 0) {
00209         --size_;
00210     }
00211 }
```

#### 5.3.2.26 push_back() [1/2]

```
template<typename T >
void Vector< T >::push_back (
            const T & value)
00187                                           {
00188     if (size_ == capacity_) reserve(nextCapacity(size_ + 1));
00189     data_[size_++] = value;
00190 }
```

#### 5.3.2.27 push_back() [2/2]

```
template<typename T >
void Vector< T >::push_back (
            T && value)
00194                                       {
00195     if (size_ == capacity_) reserve(nextCapacity(size_ + 1));
00196     data_[size_++] = std::move(value);
00197 }
```

#### 5.3.2.28 reserve()

```
template<typename T >
void Vector< T >::reserve (
            size_t new_cap)
00160                                       {
00161     if (new_cap > capacity_) {
00162         T* new_data = new T[new_cap];
00163         std::move(data_, data_ + size_, new_data);
00164         delete[] data_;
00165         data_ = new_data;
00166         capacity_ = new_cap;
00167     }
00168 }
```

#### 5.3.2.29 resize() [1/2]

```
template<typename T >
void Vector< T >::resize (
            size_t count)
00182                                     {
00183     resize(count, T());
00184 }
```

#### 5.3.2.30 resize() [2/2]

```
template<typename T >
void Vector< T >::resize (
            size_t count,
            const T & value)
00171                                                       {
00172     if (count > size_) {
00173         if (count > capacity_) {
00174             reserve(count);
00175         }
00176         std::fill(data_ + size_, data_ + count, value);
00177     }
00178     size_ = count;
00179 }
```

#### 5.3.2.31 shrink_to_fit()

```
template<typename T >
void Vector< T >::shrink_to_fit ()
00214                                {
00215     if (size_ < capacity_) {
00216         T* new_data = new T[size_];
00217         std::move(data_, data_ + size_, new_data);
00218         delete[] data_;
00219         data_ = new_data;
00220         capacity_ = size_;
00221     }
00222 }
```

#### 5.3.2.32 size()

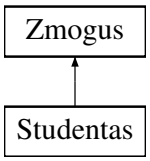
```
template<typename T >
size_t Vector< T >::size () const  [noexcept]
00100                                       {
00101     return size_;
00102 }
```

Dokumentacija šiai klasei sugeneruota iš šių failų:

- vector.h
- vector.cpp

## 5.4 Zmogus Klasė

Paveldimumo diagrama Zmogus:




**Vieši Metodai**

- Zmogus (const string &vard, const string &pavard)
- string getVardas () const
- string getPavarde () const
- void setVardas (string vard)
- void setPavarde (string pav)
- virtual void **whoAmI** ()=0

**Apsaugoti Atributai**

- string **vardas**
- string **pavarde**

### 5.4.1 Konstruktoriaus ir Destruktoriaus Dokumentacija

#### 5.4.1.1 Zmogus() [1/2]

```
Zmogus::Zmogus ()  [inline]
00023 : vardas("BeVardis"), pavarde("BePavardis") {}
```

#### 5.4.1.2 Zmogus() [2/2]

```
Zmogus::Zmogus (
            const string & vard,
            const string & pavard)  [inline]
00024 : vardas(vard), pavarde(pavard) {}
```

#### 5.4.1.3 ~Zmogus()

```
virtual Zmogus::~Zmogus ()  [inline], [virtual]
00025 {vardas.clear(); pavarde.clear();}
```

### 5.4.2 Metodų Dokumentacija

#### 5.4.2.1 getPavarde()

```
string Zmogus::getPavarde () const  [inline]
00028 { return pavarde; }
```

#### 5.4.2.2 getVardas()

```
string Zmogus::getVardas () const  [inline]
00027 { return vardas; }
```

#### 5.4.2.3 setPavarde()

```
void Zmogus::setPavarde (
             string pav)  [inline]
00030 { this->pavarde = pav; }
```

#### 5.4.2.4 setVardas()

```
void Zmogus::setVardas (
             string vard)  [inline]
00029 { this->vardas = vard; }
```

Dokumentacija šiai klasei sugeneruota iš šio failo:

- pazymiai.h

# skyrius 6

# Failo Dokumentacija

## 6.1 pazymiai.h

```
00001 #ifndef Container
00002 #include "vector.h"
00003 #define Container Vector
00004 #endif
00005
00006 #ifndef PAZYMIAI_H
00007 #define PAZYMIAI_H
00008
00009 #include <string>
00010 #include <vector>
00011 // #include <list>
00012 // #include <deque>
00013 #include "timer.h"
00014
00015 extern int ndSk, studSk, inputOption;
00016
00017 using namespace std;
00018
00019 class Zmogus {
00020     protected:
00021         string vardas, pavarde;
00022     public:
00023         Zmogus() : vardas("BeVardis"), pavarde("BePavardis") {}
00024         Zmogus(const string &vard, const string &pavard) : vardas(vard), pavarde(pavard) {}
00025         virtual ~Zmogus() {vardas.clear(); pavarde.clear();}
00026
00027         inline string getVardas() const { return vardas; }
00028         inline string getPavarde() const { return pavarde; }
00029         void setVardas(string vard) { this->vardas = vard; }
00030         void setPavarde(string pav) { this->pavarde = pav; }
00031         virtual void whoAmI() = 0;
00032 };
00033
00034 class Studentas : public Zmogus {
00035     private:
00036         int egz;
00037         double vidurkis;
00038         double mediana;
00039         vector<int> nd;
00040     public:
00041         Studentas();
00042         Studentas(const string &vardas, const string &pavarde);
00043         Studentas(const string &vardas, const string &pavarde, int egz, double vidurkis, double
      mediana, vector<int> nd) : Zmogus(vardas, pavarde),
00044             egz(egz), vidurkis(vidurkis), mediana(mediana), nd(nd) {}
00045         Studentas(const Studentas &tmpStud);
00046         Studentas(Studentas &&tmpStud) noexcept;
00047
00048         void ndAppend(int balas) {nd.push_back(balas);} //prideda viena namu darba.
00049         void ndResize(int size) {nd.resize(size);}
00050         inline int ndSk() const {return nd.size();}
00051         inline int getNd(int index) {return nd.at(index);}
00052         void setNd(int index, int value);
00053
00054         void lastNdtoEgz();
00055
00056         inline void setEgz(int value) {egz = value;}
00057         inline double getEgz() const {return egz;}
```

```
00058
00059
00060         inline double getVidurkis() const {return vidurkis;}
00061         inline double getMediana() const {return mediana;}
00062         void skaiciuotiVid();
00063         void skaiciuotiMed();
00064
00065         ~Studentas() {
00066             nd.clear();
00067         }
00068
00069         inline void clearEverything() { this->vardas.clear(); this->pavarde.clear(); this->nd.clear();
      this->egz = 0; this->mediana = 0.0; this->vidurkis = 0.0; }
00070
00071         Studentas& operator=(const Studentas &tmpStud);
00072         Studentas& operator=(Studentas &&tmpStud);
00073         friend istringstream& operator»(istringstream& filename, Studentas &tmpStud);
00074         friend istream& operator»(istream& manual, Studentas &tmpStud);
00075         friend ostream& operator«(ostream& console, const Studentas &tmpStud);
00076         friend ofstream& operator«(ofstream& filename, const Studentas &tmpStud);
00077
00078         bool operator==(const Studentas& other) const;
00079         bool operator!=(const Studentas& other) const;
00080
00081         virtual void whoAmI() {cout « "Studentas klasė" « endl;}
00082 };
00083
00084 void pasirinktiEiga(string msg, int* option, int max);
00085 bool taipArNe(string uzklausa);
00086 int countNd(string &line);
00087
00088 void rikiuotiPagalParametra(Container<Studentas> &studentai, int option);
00089 void equalOutNdSk(Container<Studentas> &studentai, int ndSk);
00090 void sortAndAddToFile(Container<Studentas> &kietekai, Container<Studentas> &vargsiukai, int option);
00091 void atspauzdintiMasyvoInfo(Container<Studentas> &studentai);
00092
00093 #endif
```

## 6.2 timer.h

```
00001 #include <chrono>
00002 class Timer {
00003   private:
00004     std::chrono::time_point<std::chrono::high_resolution_clock> start;
00005   public:
00006     Timer() : start{std::chrono::high_resolution_clock::now()} {}
00007     void reset() {
00008       start = std::chrono::high_resolution_clock::now();
00009     }
00010     double elapsed() const {return
      std::chrono::duration<double>(std::chrono::high_resolution_clock::now() - start).count();
00011     }
00012 };
```

## 6.3 vector.h

```
00001 #ifndef VECTOR_H
00002 #define VECTOR_H
00003
00004 #include <algorithm>
00005 #include <initializer_list>
00006
00007 template<typename T>
00008 class Vector {
00009     private:
00010         T* data_;
00011         size_t size_;
00012         size_t capacity_;
00013
00014     public:
00015         Vector();
00016         explicit Vector(size_t initial_size);
00017         Vector(std::initializer_list<T> init);
00018         Vector(const Vector& other);
00019         Vector(Vector&& other) noexcept;
00020         ~Vector();
00021
00022         Vector& operator=(const Vector& other);
00023         Vector& operator=(Vector&& other) noexcept;
```

```
00024
00025         T& operator[](size_t index);
00026         const T& operator[](size_t index) const;
00027         T& at(size_t index);
00028         const T& at(size_t index) const;
00029         T& front();
00030         const T& front() const;
00031         T& back();
00032         const T& back() const;
00033
00034         T* begin() noexcept;
00035         const T* begin() const noexcept;
00036         T* end() noexcept;
00037         const T* end() const noexcept;
00038
00039         bool empty() const noexcept;
00040         size_t size() const noexcept;
00041         void reserve(size_t new_cap);
00042         size_t capacity() const noexcept;
00043         void shrink_to_fit();
00044         void assign(size_t n, const T& value);
00045
00046         void clear() noexcept;
00047         void push_back(const T& value);
00048         void push_back(T&& value);
00049         void pop_back();
00050         void resize(size_t count);
00051         void resize(size_t count, const T& value);
00052         void insert(size_t index, const T& value);
00053         void insert_range(size_t index, size_t n, const T& value);
00054         void emplace(size_t index, T&& value);
00055         void emplace_back(T&& value);
00056
00057         bool operator==(const Vector& other) const;
00058         bool operator!=(const Vector& other) const;
00059 };
00060
00061 size_t nextCapacity(const size_t &size);
00062
00063 #endif
```

# Rodyklė


~Studentas
    Studentas, 13
~Vector
    Vector< T >, 20
~Zmogus
    Zmogus, 27

assign
    Vector< T >, 21
at
    Vector< T >, 21

back
    Vector< T >, 21
begin
    Vector< T >, 21, 22

capacity
    Vector< T >, 22
clear
    Vector< T >, 22
clearEverything
    Studentas, 13

elapsed
    Timer, 19
emplace
    Vector< T >, 22
emplace_back
    Vector< T >, 22
empty
    Vector< T >, 22
end
    Vector< T >, 23

front
    Vector< T >, 23

getEgz
    Studentas, 13
getMediana
    Studentas, 13
getNd
    Studentas, 13
getPavarde
    Zmogus, 27
getVardas
    Zmogus, 27
getVidurkis
    Studentas, 13

insert
    Vector< T >, 23
insert_range
    Vector< T >, 23

lastNdtoEgz
    Studentas, 14

ndAppend
    Studentas, 14
ndResize
    Studentas, 14
ndSk
    Studentas, 14

operator!=
    Studentas, 14
    Vector< T >, 24
operator<<
    Studentas, 16
operator>>
    Studentas, 16, 18
operator=
    Studentas, 14
    Vector< T >, 24
operator==
    Studentas, 15
    Vector< T >, 24
operator[]
    Vector< T >, 25

pop_back
    Vector< T >, 25
push_back
    Vector< T >, 25

reserve
    Vector< T >, 25
reset
    Timer, 19
resize
    Vector< T >, 26

setEgz
    Studentas, 15
setNd
    Studentas, 15
setPavarde
    Zmogus, 28
setVardas

Zmogus, 28
shrink_to_fit
Vector< T >, 26
size
Vector< T >, 26
skaiciuotiMed
Studentas, 15
skaiciuotiVid
Studentas, 15
Studentas, 11
~Studentas, 13
clearEverything, 13
getEgz, 13
getMediana, 13
getNd, 13
getVidurkis, 13
lastNdtoEgz, 14
ndAppend, 14
ndResize, 14
ndSk, 14
operator!=, 14
operator<<, 16
operator>>, 16, 18
operator=, 14
operator==, 15
setEgz, 15
setNd, 15
skaiciuotiMed, 15
skaiciuotiVid, 15
Studentas, 12, 13
whoAmI, 16

Timer, 18
elapsed, 19
reset, 19
Timer, 18

Vector
Vector< T >, 20
Vector< T >, 19
~Vector, 20
assign, 21
at, 21
back, 21
begin, 21, 22
capacity, 22
clear, 22
emplace, 22
emplace_back, 22
empty, 22
end, 23
front, 23
insert, 23
insert_range, 23
operator!=, 24
operator=, 24
operator==, 24
operator[], 25
pop_back, 25
push_back, 25
reserve, 25
resize, 26
shrink_to_fit, 26
size, 26
Vector, 20
VU_OP_3uzd-MS, 1

whoAmI
Studentas, 16

Zmogus, 27
~Zmogus, 27
getPavarde, 27
getVardas, 27
setPavarde, 28
setVardas, 28
Zmogus, 27