My Project

Generated by Doxygen 1.10.0

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 <class_studentai.h>
```

Inheritance diagram for studentas:

Collaboration diagram for studentas:

Public Member Functions

- studentas ()
- studentas (const std::string &vardas, const std::string &pavarde, const std::vector< int > &ND, int EGZ)
- virtual std::string getVardas () const override
- virtual std::string getPavarde () const override
- ∼studentas ()
- studentas (const studentas &other)
- studentas (studentas &&other) noexcept
- studentas & operator= (const studentas &other)
- studentas & operator= (studentas &&other) noexcept
- std::vector< int > getND () const
- int getEGZ () const
- double getGalutinisV () const
- double getGalutinisM () const
- void setVardas (const std::string &newName)
- void setPavarde (const std::string &newSurname)
- void setND (const std::vector< int > &newND)
- void setEGZ (int newEGZ)
- void clearEverything ()

Public Member Functions inherited from zmogus

- zmogus ()
- zmogus (const std::string &vardas, const std::string &pavarde)
- ∼zmogus ()

Friends

- std::istream & operator>> (std::istream &is, studentas &s)
- std::ostream & operator<< (std::ostream &os, const studentas &s)

Additional Inherited Members

Protected Attributes inherited from zmogus

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

4.1.1 Detailed Description

Definition at line 41 of file class_studentai.h.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 studentas() [1/4]

```
studentas::studentas ( )  [inline]
```

Definition at line 71 of file class_studentai.h.

4.1.2.2 studentas() [2/4]

```
studentas::studentas (
              const std::string & vardas,
              const std::string & pavarde,
              const std::vector< int > & ND,
              int EGZ )  [inline]
```

Definition at line 75 of file class_studentai.h.

4.1.2.3 ~studentas()

```
studentas::~studentas ( )  [inline]
```

Definition at line 91 of file class_studentai.h.

4.1.2.4 studentas() [3/4]

```
studentas::studentas (
              const studentas & other )  [inline]
```

Definition at line 94 of file class_studentai.h.

4.1.2.5 studentas() [4/4]

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

Definition at line 105 of file class_studentai.h.

4.1.3 Member Function Documentation

4.1.3.1 clearEverything()

```
void studentas::clearEverything ( )  [inline]
```

Definition at line 221 of file class_studentai.h.

4.1.3.2 getEGZ()

```
int studentas::getEGZ ( ) const  [inline]
```

Definition at line 153 of file class_studentai.h.

Here is the caller graph for this function:

4.1.3.3 getGalutinisM()

```
double studentas::getGalutinisM ( ) const  [inline]
```

Definition at line 155 of file class_studentai.h.

Here is the caller graph for this function:

4.1.3.4 getGalutinisV()

```
double studentas::getGalutinisV ( ) const  [inline]
```

Definition at line 154 of file class_studentai.h.

Here is the caller graph for this function:

4.1.3.5 getND()

```
std::vector< int > studentas::getND ( ) const  [inline]
```

Definition at line 152 of file class_studentai.h.

Here is the caller graph for this function:

4.1.3.6 getPavarde()

```
virtual std::string studentas::getPavarde ( ) const  [inline], [override], [virtual]
```

Implements zmogus.

Definition at line 87 of file class_studentai.h.

Here is the caller graph for this function:

4.1.3.7 getVardas()

```
virtual std::string studentas::getVardas ( ) const  [inline], [override], [virtual]
```

Implements zmogus.

Definition at line 83 of file class_studentai.h.

Here is the caller graph for this function:

4.1.3.8 operator=() [1/2]

```
studentas & studentas::operator= (
             const studentas & other )  [inline]
```

Definition at line 117 of file class_studentai.h.

4.1.3.9 operator=() [2/2]

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

Definition at line 133 of file class_studentai.h.

4.1.3.10 setEGZ()

```
void studentas::setEGZ (
             int newEGZ )  [inline]
```

Definition at line 161 of file class_studentai.h.

Here is the caller graph for this function:

4.1.3.11 setND()

```
void studentas::setND (
             const std::vector< int > & newND )  [inline]
```

Definition at line 160 of file class_studentai.h.

Here is the caller graph for this function:

4.1.3.12 setPavarde()

```
void studentas::setPavarde (
             const std::string & newSurname )  [inline], [virtual]
```

Reimplemented from zmogus.

Definition at line 159 of file class_studentai.h.

Here is the caller graph for this function:

4.1.3.13 setVardas()

```
void studentas::setVardas (
             const std::string & newName )  [inline], [virtual]
```

Reimplemented from zmogus.

Definition at line 158 of file class_studentai.h.

Here is the caller graph for this function:

4.1.4 Friends And Related Symbol Documentation

4.1.4.1 operator<<

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

Definition at line 212 of file class_studentai.h.

4.1.4.2 operator>>

```
std::istream & operator>> (
             std::istream & is,
             studentas & s )  [friend]
```

Definition at line 167 of file class_studentai.h.

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

- class_studentai.h

4.2 zmogus Class Reference

```
#include <class_studentai.h>
```

Inheritance diagram for zmogus:

Public Member Functions

- zmogus ()
- zmogus (const std::string &vardas, const std::string &pavarde)
- ∼zmogus ()
- virtual std::string getVardas () const =0
- virtual std::string getPavarde () const =0
- virtual void setVardas (const std::string &newName)
- virtual void setPavarde (const std::string &newSurname)

Protected Attributes

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

4.2.1 Detailed Description

Definition at line 19 of file class_studentai.h.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 zmogus() [1/2]

```
zmogus::zmogus ( )  [inline]
```

Definition at line 25 of file class_studentai.h.

4.2.2.2 zmogus() [2/2]

```
zmogus::zmogus (
            const std::string & vardas,
            const std::string & pavarde )  [inline]
```

Definition at line 26 of file class_studentai.h.

4.2.2.3 ∼zmogus()

```
zmogus::∼zmogus ( )  [inline]
```

Definition at line 29 of file class_studentai.h.

4.2.3 Member Function Documentation

4.2.3.1 getPavarde()

```
virtual std::string zmogus::getPavarde ( ) const  [pure virtual]
```

Implemented in studentas.

4.2.3.2 getVardas()

```
virtual std::string zmogus::getVardas ( ) const  [pure virtual]
```

Implemented in studentas.

4.2.3.3 setPavarde()

```
virtual void zmogus::setPavarde (
             const std::string & newSurname )  [inline], [virtual]
```

Reimplemented in studentas.

Definition at line 36 of file class_studentai.h.

4.2.3.4 setVardas()

```
virtual void zmogus::setVardas (
             const std::string & newName )  [inline], [virtual]
```

Reimplemented in studentas.

Definition at line 35 of file class_studentai.h.

4.2.4 Member Data Documentation

4.2.4.1 pavarde

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

Definition at line 22 of file class_studentai.h.

4.2.4.2 vardas

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

Definition at line 21 of file class_studentai.h.

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

- class_studentai.h

Chapter 5

File Documentation

5.1 build/CMakeFiles/3.29.2/CompilerIdC/CMakeCCompilerId.c File Reference

Macros

- #define __has_include(x) 0
- #define COMPILER_ID ""
- #define STRINGIFY_HELPER(X) #X
- #define STRINGIFY(X) STRINGIFY_HELPER(X)
- #define PLATFORM_ID
- #define ARCHITECTURE_ID
- #define DEC(n)
- #define HEX(n)
- #define C_VERSION

Functions

- int main (int argc, char ∗argv[ ])

Variables

- char const ∗ info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
- char const ∗ info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
- char const ∗ info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
- const char ∗ info_language_standard_default
- const char ∗ info_language_extensions_default

5.1.1 Macro Definition Documentation

5.1.1.1 __has_include

```
#define __has_include(
                x ) 0
```

Definition at line 17 of file CMakeCCompilerId.c.

5.1.1.2 ARCHITECTURE_ID

```
#define ARCHITECTURE_ID
```

Definition at line 745 of file CMakeCCompilerId.c.

5.1.1.3 C_VERSION

```
#define C_VERSION
```

Definition at line 834 of file CMakeCCompilerId.c.

5.1.1.4 COMPILER_ID

```
#define COMPILER_ID ""
```

Definition at line 448 of file CMakeCCompilerId.c.

5.1.1.5 DEC

```
#define DEC(
             n )
```

Value:

```
  ('0' + (((n) / 10000000)%10)), \
  ('0' + (((n) / 1000000)%10)),  \
  ('0' + (((n) / 100000)%10)),   \
  ('0' + (((n) / 10000)%10)),    \
  ('0' + (((n) / 1000)%10)),     \
  ('0' + (((n) / 100)%10)),      \
  ('0' + (((n) / 10)%10)),       \
  ('0' +  ((n) % 10))
```

Definition at line 749 of file CMakeCCompilerId.c.

5.1.1.6 HEX

```
#define HEX(
             n )
```

Value:

```
  ('0' + ((n)»28 & 0xF)), \
  ('0' + ((n)»24 & 0xF)), \
  ('0' + ((n)»20 & 0xF)), \
  ('0' + ((n)»16 & 0xF)), \
  ('0' + ((n)»12 & 0xF)), \
  ('0' + ((n)»8  & 0xF)), \
  ('0' + ((n)»4  & 0xF)), \
  ('0' + ((n)     & 0xF))
```

Definition at line 760 of file CMakeCCompilerId.c.

5.1.1.7 PLATFORM_ID

```
#define PLATFORM_ID
```

Definition at line 579 of file CMakeCCompilerId.c.

5.1.1.8 STRINGIFY

```
#define STRINGIFY(
              X ) STRINGIFY_HELPER(X)
```

Definition at line 469 of file CMakeCCompilerId.c.

5.1.1.9 STRINGIFY_HELPER

```
#define STRINGIFY_HELPER(
              X ) #X
```

Definition at line 468 of file CMakeCCompilerId.c.

5.1.2 Function Documentation

5.1.2.1 main()

```
int main (
             int argc,
             char * argv[] )
```

Definition at line 868 of file CMakeCCompilerId.c.

5.1.3 Variable Documentation

5.1.3.1 info_arch

```
char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
```

Definition at line 826 of file CMakeCCompilerId.c.

5.1.3.2 info_compiler

```
char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
```

Definition at line 455 of file CMakeCCompilerId.c.

5.1.3.3 info_language_extensions_default

```
const char* info_language_extensions_default
```

Initial value:

```
= "INFO" ":" "extensions_default["




  "OFF"

"]"
```

Definition at line 850 of file CMakeCCompilerId.c.

5.1.3.4 info_language_standard_default

```
const char* info_language_standard_default
```

Initial value:

```
=
  "INFO" ":" "standard_default[" C_VERSION "]"
```

Definition at line 847 of file CMakeCCompilerId.c.

5.1.3.5 info_platform

```
char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
```

Definition at line 825 of file CMakeCCompilerId.c.

5.2 CMakeCCompilerId.c

Go to the documentation of this file.

```
#ifdef __cplusplus
# error "A C++ compiler has been selected for C."
#endif

#if defined(__18CXX)
# define ID_VOID_MAIN
#endif
#if defined(__CLASSIC_C__)
/* cv-qualifiers did not exist in K&R C */
# define const
# define volatile
#endif

#if !defined(__has_include)
/* If the compiler does not have __has_include, pretend the answer is
   always no.  */
#  define __has_include(x) 0
#endif


/* Version number components: V=Version, R=Revision, P=Patch
   Version date components:   YYYY=Year, MM=Month,   DD=Day  */

#if defined(__INTEL_COMPILER) || defined(__ICC)
# define COMPILER_ID "Intel"
# if defined(_MSC_VER)
#  define SIMULATE_ID "MSVC"
# endif
# if defined(__GNUC__)
#  define SIMULATE_ID "GNU"
# endif
  /* __INTEL_COMPILER = VRP prior to 2021, and then VVVV for 2021 and later,
     except that a few beta releases use the old format with V=2021.  */
# if __INTEL_COMPILER < 2021 || __INTEL_COMPILER == 202110 || __INTEL_COMPILER == 202111
#  define COMPILER_VERSION_MAJOR DEC(__INTEL_COMPILER/100)
#  define COMPILER_VERSION_MINOR DEC(__INTEL_COMPILER/10 % 10)
#  if defined(__INTEL_COMPILER_UPDATE)
#   define COMPILER_VERSION_PATCH DEC(__INTEL_COMPILER_UPDATE)
#  else
#   define COMPILER_VERSION_PATCH DEC(__INTEL_COMPILER   % 10)
#  endif
# else
#  define COMPILER_VERSION_MAJOR DEC(__INTEL_COMPILER)
#  define COMPILER_VERSION_MINOR DEC(__INTEL_COMPILER_UPDATE)
   /* The third version component from --version is an update index,
      but no macro is provided for it.  */
#  define COMPILER_VERSION_PATCH DEC(0)
# endif
# if defined(__INTEL_COMPILER_BUILD_DATE)
   /* __INTEL_COMPILER_BUILD_DATE = YYYYMMDD */
#  define COMPILER_VERSION_TWEAK DEC(__INTEL_COMPILER_BUILD_DATE)
# endif
# if defined(_MSC_VER)
   /* _MSC_VER = VVRR */
#  define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
#  define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
# endif
# if defined(__GNUC__)
#  define SIMULATE_VERSION_MAJOR DEC(__GNUC__)
# elif defined(__GNUG__)
#  define SIMULATE_VERSION_MAJOR DEC(__GNUG__)
# endif
# if defined(__GNUC_MINOR__)
#  define SIMULATE_VERSION_MINOR DEC(__GNUC_MINOR__)
# endif
# if defined(__GNUC_PATCHLEVEL__)
#  define SIMULATE_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
# endif

#elif (defined(__clang__) && defined(__INTEL_CLANG_COMPILER)) || defined(__INTEL_LLVM_COMPILER)
# define COMPILER_ID "IntelLLVM"
#if defined(_MSC_VER)
# define SIMULATE_ID "MSVC"
#endif
#if defined(__GNUC__)
# define SIMULATE_ID "GNU"
#endif
/* __INTEL_LLVM_COMPILER = VVVVRP prior to 2021.2.0, VVVVRRPP for 2021.2.0 and
 * later.  Look for 6 digit vs. 8 digit version number to decide encoding.
 * VVVV is no smaller than the current year when a version is released.
 */
#if __INTEL_LLVM_COMPILER < 1000000L
# define COMPILER_VERSION_MAJOR DEC(__INTEL_LLVM_COMPILER/100)
# define COMPILER_VERSION_MINOR DEC(__INTEL_LLVM_COMPILER/10 % 10)
# define COMPILER_VERSION_PATCH DEC(__INTEL_LLVM_COMPILER    % 10)
#else
# define COMPILER_VERSION_MAJOR DEC(__INTEL_LLVM_COMPILER/10000)
# define COMPILER_VERSION_MINOR DEC(__INTEL_LLVM_COMPILER/100 % 100)
# define COMPILER_VERSION_PATCH DEC(__INTEL_LLVM_COMPILER     % 100)
#endif
#if defined(_MSC_VER)
  /* _MSC_VER = VVRR */
# define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
# define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
#endif
#if defined(__GNUC__)
# define SIMULATE_VERSION_MAJOR DEC(__GNUC__)
#elif defined(__GNUG__)
# define SIMULATE_VERSION_MAJOR DEC(__GNUG__)
#endif
#if defined(__GNUC_MINOR__)
# define SIMULATE_VERSION_MINOR DEC(__GNUC_MINOR__)
#endif
#if defined(__GNUC_PATCHLEVEL__)
# define SIMULATE_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
#endif

#elif defined(__PATHCC__)
# define COMPILER_ID "PathScale"
# define COMPILER_VERSION_MAJOR DEC(__PATHCC__)
# define COMPILER_VERSION_MINOR DEC(__PATHCC_MINOR__)
# if defined(__PATHCC_PATCHLEVEL__)
#  define COMPILER_VERSION_PATCH DEC(__PATHCC_PATCHLEVEL__)
# endif

#elif defined(__BORLANDC__) && defined(__CODEGEARC_VERSION__)
```

```
# define COMPILER_ID "Embarcadero"
# define COMPILER_VERSION_MAJOR HEX(__CODEGEARC_VERSION__»24 & 0x00FF)
# define COMPILER_VERSION_MINOR HEX(__CODEGEARC_VERSION__»16 & 0x00FF)
# define COMPILER_VERSION_PATCH DEC(__CODEGEARC_VERSION__     & 0xFFFF)

#elif defined(__BORLANDC__)
# define COMPILER_ID "Borland"
  /* __BORLANDC__ = 0xVRR */
# define COMPILER_VERSION_MAJOR HEX(__BORLANDC__»8)
# define COMPILER_VERSION_MINOR HEX(__BORLANDC__ & 0xFF)

#elif defined(__WATCOMC__) && __WATCOMC__ < 1200
# define COMPILER_ID "Watcom"
   /* __WATCOMC__ = VVRR */
# define COMPILER_VERSION_MAJOR DEC(__WATCOMC__ / 100)
# define COMPILER_VERSION_MINOR DEC((__WATCOMC__ / 10) % 10)
# if (__WATCOMC__ % 10) > 0
#  define COMPILER_VERSION_PATCH DEC(__WATCOMC__ % 10)
# endif

#elif defined(__WATCOMC__)
# define COMPILER_ID "OpenWatcom"
   /* __WATCOMC__ = VVRP + 1100 */
# define COMPILER_VERSION_MAJOR DEC((__WATCOMC__ - 1100) / 100)
# define COMPILER_VERSION_MINOR DEC((__WATCOMC__ / 10) % 10)
# if (__WATCOMC__ % 10) > 0
#  define COMPILER_VERSION_PATCH DEC(__WATCOMC__ % 10)
# endif

#elif defined(__SUNPRO_C)
# define COMPILER_ID "SunPro"
# if __SUNPRO_C >= 0x5100
   /* __SUNPRO_C = 0xVRRP */
#  define COMPILER_VERSION_MAJOR HEX(__SUNPRO_C»12)
#  define COMPILER_VERSION_MINOR HEX(__SUNPRO_C»4 & 0xFF)
#  define COMPILER_VERSION_PATCH HEX(__SUNPRO_C    & 0xF)
# else
   /* __SUNPRO_CC = 0xVRP */
#  define COMPILER_VERSION_MAJOR HEX(__SUNPRO_C»8)
#  define COMPILER_VERSION_MINOR HEX(__SUNPRO_C»4 & 0xF)
#  define COMPILER_VERSION_PATCH HEX(__SUNPRO_C    & 0xF)
# endif

#elif defined(__HP_cc)
# define COMPILER_ID "HP"
  /* __HP_cc = VVRRPP */
# define COMPILER_VERSION_MAJOR DEC(__HP_cc/10000)
# define COMPILER_VERSION_MINOR DEC(__HP_cc/100 % 100)
# define COMPILER_VERSION_PATCH DEC(__HP_cc     % 100)

#elif defined(__DECC)
# define COMPILER_ID "Compaq"
  /* __DECC_VER = VVRRTPPPP */
# define COMPILER_VERSION_MAJOR DEC(__DECC_VER/10000000)
# define COMPILER_VERSION_MINOR DEC(__DECC_VER/100000  % 100)
# define COMPILER_VERSION_PATCH DEC(__DECC_VER         % 10000)

#elif defined(__IBMC__) && defined(__COMPILER_VER__)
# define COMPILER_ID "zOS"
  /* __IBMC__ = VRP */
# define COMPILER_VERSION_MAJOR DEC(__IBMC__/100)
# define COMPILER_VERSION_MINOR DEC(__IBMC__/10 % 10)
# define COMPILER_VERSION_PATCH DEC(__IBMC__    % 10)

#elif defined(__open_xl__) && defined(__clang__)
# define COMPILER_ID "IBMClang"
# define COMPILER_VERSION_MAJOR DEC(__open_xl_version__)
# define COMPILER_VERSION_MINOR DEC(__open_xl_release__)
# define COMPILER_VERSION_PATCH DEC(__open_xl_modification__)
# define COMPILER_VERSION_TWEAK DEC(__open_xl_ptf_fix_level__)


#elif defined(__ibmxl__) && defined(__clang__)
# define COMPILER_ID "XLClang"
# define COMPILER_VERSION_MAJOR DEC(__ibmxl_version__)
# define COMPILER_VERSION_MINOR DEC(__ibmxl_release__)
# define COMPILER_VERSION_PATCH DEC(__ibmxl_modification__)
# define COMPILER_VERSION_TWEAK DEC(__ibmxl_ptf_fix_level__)


#elif defined(__IBMC__) && !defined(__COMPILER_VER__) && __IBMC__ >= 800
# define COMPILER_ID "XL"
  /* __IBMC__ = VRP */
# define COMPILER_VERSION_MAJOR DEC(__IBMC__/100)
# define COMPILER_VERSION_MINOR DEC(__IBMC__/10 % 10)
# define COMPILER_VERSION_PATCH DEC(__IBMC__    % 10)

```

```
#elif defined(__IBMC__) && !defined(__COMPILER_VER__) && __IBMC__ < 800
# define COMPILER_ID "VisualAge"
  /* __IBMC__ = VRP */
# define COMPILER_VERSION_MAJOR DEC(__IBMC__/100)
# define COMPILER_VERSION_MINOR DEC(__IBMC__/10 % 10)
# define COMPILER_VERSION_PATCH DEC(__IBMC__    % 10)

#elif defined(__NVCOMPILER)
# define COMPILER_ID "NVHPC"
# define COMPILER_VERSION_MAJOR DEC(__NVCOMPILER_MAJOR__)
# define COMPILER_VERSION_MINOR DEC(__NVCOMPILER_MINOR__)
# if defined(__NVCOMPILER_PATCHLEVEL__)
#  define COMPILER_VERSION_PATCH DEC(__NVCOMPILER_PATCHLEVEL__)
# endif

#elif defined(__PGI)
# define COMPILER_ID "PGI"
# define COMPILER_VERSION_MAJOR DEC(__PGIC__)
# define COMPILER_VERSION_MINOR DEC(__PGIC_MINOR__)
# if defined(__PGIC_PATCHLEVEL__)
#  define COMPILER_VERSION_PATCH DEC(__PGIC_PATCHLEVEL__)
# endif

#elif defined(__clang__) && defined(__cray__)
# define COMPILER_ID "CrayClang"
# define COMPILER_VERSION_MAJOR DEC(__cray_major__)
# define COMPILER_VERSION_MINOR DEC(__cray_minor__)
# define COMPILER_VERSION_PATCH DEC(__cray_patchlevel__)
# define COMPILER_VERSION_INTERNAL_STR __clang_version__


#elif defined(_CRAYC)
# define COMPILER_ID "Cray"
# define COMPILER_VERSION_MAJOR DEC(_RELEASE_MAJOR)
# define COMPILER_VERSION_MINOR DEC(_RELEASE_MINOR)

#elif defined(__TI_COMPILER_VERSION__)
# define COMPILER_ID "TI"
  /* __TI_COMPILER_VERSION__ = VVVRRRPPP */
# define COMPILER_VERSION_MAJOR DEC(__TI_COMPILER_VERSION__/1000000)
# define COMPILER_VERSION_MINOR DEC(__TI_COMPILER_VERSION__/1000   % 1000)
# define COMPILER_VERSION_PATCH DEC(__TI_COMPILER_VERSION__        % 1000)

#elif defined(__CLANG_FUJITSU)
# define COMPILER_ID "FujitsuClang"
# define COMPILER_VERSION_MAJOR DEC(__FCC_major__)
# define COMPILER_VERSION_MINOR DEC(__FCC_minor__)
# define COMPILER_VERSION_PATCH DEC(__FCC_patchlevel__)
# define COMPILER_VERSION_INTERNAL_STR __clang_version__


#elif defined(__FUJITSU)
# define COMPILER_ID "Fujitsu"
# if defined(__FCC_version__)
#   define COMPILER_VERSION __FCC_version__
# elif defined(__FCC_major__)
#   define COMPILER_VERSION_MAJOR DEC(__FCC_major__)
#   define COMPILER_VERSION_MINOR DEC(__FCC_minor__)
#   define COMPILER_VERSION_PATCH DEC(__FCC_patchlevel__)
# endif
# if defined(__fcc_version)
#   define COMPILER_VERSION_INTERNAL DEC(__fcc_version)
# elif defined(__FCC_VERSION)
#   define COMPILER_VERSION_INTERNAL DEC(__FCC_VERSION)
# endif


#elif defined(__ghs__)
# define COMPILER_ID "GHS"
/* __GHS_VERSION_NUMBER = VVVVRP */
# ifdef __GHS_VERSION_NUMBER
# define COMPILER_VERSION_MAJOR DEC(__GHS_VERSION_NUMBER / 100)
# define COMPILER_VERSION_MINOR DEC(__GHS_VERSION_NUMBER / 10 % 10)
# define COMPILER_VERSION_PATCH DEC(__GHS_VERSION_NUMBER      % 10)
# endif

#elif defined(__TASKING__)
# define COMPILER_ID "Tasking"
  # define COMPILER_VERSION_MAJOR DEC(__VERSION__/1000)
  # define COMPILER_VERSION_MINOR DEC(__VERSION__ % 100)
# define COMPILER_VERSION_INTERNAL DEC(__VERSION__)

#elif defined(__ORANGEC__)
# define COMPILER_ID "OrangeC"
# define COMPILER_VERSION_MAJOR DEC(__ORANGEC_MAJOR__)
# define COMPILER_VERSION_MINOR DEC(__ORANGEC_MINOR__)
# define COMPILER_VERSION_PATCH DEC(__ORANGEC_PATCHLEVEL__)
```

```

#elif defined(__TINYC__)
# define COMPILER_ID "TinyCC"

#elif defined(__BCC__)
# define COMPILER_ID "Bruce"

#elif defined(__SCO_VERSION__)
# define COMPILER_ID "SCO"

#elif defined(__ARMCC_VERSION) && !defined(__clang__)
# define COMPILER_ID "ARMCC"
#if __ARMCC_VERSION >= 1000000
  /* __ARMCC_VERSION = VRRPPPP */
  # define COMPILER_VERSION_MAJOR DEC(__ARMCC_VERSION/1000000)
  # define COMPILER_VERSION_MINOR DEC(__ARMCC_VERSION/10000 % 100)
  # define COMPILER_VERSION_PATCH DEC(__ARMCC_VERSION     % 10000)
#else
  /* __ARMCC_VERSION = VRPPPP */
  # define COMPILER_VERSION_MAJOR DEC(__ARMCC_VERSION/100000)
  # define COMPILER_VERSION_MINOR DEC(__ARMCC_VERSION/10000 % 10)
  # define COMPILER_VERSION_PATCH DEC(__ARMCC_VERSION    % 10000)
#endif


#elif defined(__clang__) && defined(__apple_build_version__)
# define COMPILER_ID "AppleClang"
# if defined(_MSC_VER)
#  define SIMULATE_ID "MSVC"
# endif
# define COMPILER_VERSION_MAJOR DEC(__clang_major__)
# define COMPILER_VERSION_MINOR DEC(__clang_minor__)
# define COMPILER_VERSION_PATCH DEC(__clang_patchlevel__)
# if defined(_MSC_VER)
   /* _MSC_VER = VVRR */
#  define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
#  define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
# endif
# define COMPILER_VERSION_TWEAK DEC(__apple_build_version__)

#elif defined(__clang__) && defined(__ARMCOMPILER_VERSION)
# define COMPILER_ID "ARMClang"
  # define COMPILER_VERSION_MAJOR DEC(__ARMCOMPILER_VERSION/1000000)
  # define COMPILER_VERSION_MINOR DEC(__ARMCOMPILER_VERSION/10000 % 100)
  # define COMPILER_VERSION_PATCH DEC(__ARMCOMPILER_VERSION/100   % 100)
# define COMPILER_VERSION_INTERNAL DEC(__ARMCOMPILER_VERSION)

#elif defined(__clang__) && defined(__ti__)
# define COMPILER_ID "TIClang"
  # define COMPILER_VERSION_MAJOR DEC(__ti_major__)
  # define COMPILER_VERSION_MINOR DEC(__ti_minor__)
  # define COMPILER_VERSION_PATCH DEC(__ti_patchlevel__)
# define COMPILER_VERSION_INTERNAL DEC(__ti_version__)

#elif defined(__clang__)
# define COMPILER_ID "Clang"
# if defined(_MSC_VER)
#  define SIMULATE_ID "MSVC"
# endif
# define COMPILER_VERSION_MAJOR DEC(__clang_major__)
# define COMPILER_VERSION_MINOR DEC(__clang_minor__)
# define COMPILER_VERSION_PATCH DEC(__clang_patchlevel__)
# if defined(_MSC_VER)
   /* _MSC_VER = VVRR */
#  define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
#  define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
# endif

#elif defined(__LCC__) && (defined(__GNUC__) || defined(__GNUG__) || defined(__MCST__))
# define COMPILER_ID "LCC"
# define COMPILER_VERSION_MAJOR DEC(__LCC__ / 100)
# define COMPILER_VERSION_MINOR DEC(__LCC__ % 100)
# if defined(__LCC_MINOR__)
#  define COMPILER_VERSION_PATCH DEC(__LCC_MINOR__)
# endif
# if defined(__GNUC__) && defined(__GNUC_MINOR__)
#  define SIMULATE_ID "GNU"
#  define SIMULATE_VERSION_MAJOR DEC(__GNUC__)
#  define SIMULATE_VERSION_MINOR DEC(__GNUC_MINOR__)
#  if defined(__GNUC_PATCHLEVEL__)
#   define SIMULATE_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
#  endif
# endif

#elif defined(__GNUC__)
# define COMPILER_ID "GNU"
# define COMPILER_VERSION_MAJOR DEC(__GNUC__)
# if defined(__GNUC_MINOR__)
#  define COMPILER_VERSION_MINOR DEC(__GNUC_MINOR__)
# endif
# if defined(__GNUC_PATCHLEVEL__)
#  define COMPILER_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
# endif

#elif defined(_MSC_VER)
# define COMPILER_ID "MSVC"
  /* _MSC_VER = VVRR */
# define COMPILER_VERSION_MAJOR DEC(_MSC_VER / 100)
# define COMPILER_VERSION_MINOR DEC(_MSC_VER % 100)
# if defined(_MSC_FULL_VER)
#  if _MSC_VER >= 1400
    /* _MSC_FULL_VER = VVRRPPPPP */
#   define COMPILER_VERSION_PATCH DEC(_MSC_FULL_VER % 100000)
#  else
    /* _MSC_FULL_VER = VVRRPPPP */
#   define COMPILER_VERSION_PATCH DEC(_MSC_FULL_VER % 10000)
#  endif
# endif
# if defined(_MSC_BUILD)
#  define COMPILER_VERSION_TWEAK DEC(_MSC_BUILD)
# endif

#elif defined(_ADI_COMPILER)
# define COMPILER_ID "ADSP"
#if defined(__VERSIONNUM__)
  /* __VERSIONNUM__ = 0xVVRRPPTT */
#  define COMPILER_VERSION_MAJOR DEC(__VERSIONNUM__ » 24 & 0xFF)
#  define COMPILER_VERSION_MINOR DEC(__VERSIONNUM__ » 16 & 0xFF)
#  define COMPILER_VERSION_PATCH DEC(__VERSIONNUM__ » 8 & 0xFF)
#  define COMPILER_VERSION_TWEAK DEC(__VERSIONNUM__ & 0xFF)
#endif

#elif defined(__IAR_SYSTEMS_ICC__) || defined(__IAR_SYSTEMS_ICC)
# define COMPILER_ID "IAR"
# if defined(__VER__) && defined(__ICCARM__)
#  define COMPILER_VERSION_MAJOR DEC((__VER__) / 1000000)
#  define COMPILER_VERSION_MINOR DEC(((__VER__) / 1000) % 1000)
#  define COMPILER_VERSION_PATCH DEC((__VER__) % 1000)
#  define COMPILER_VERSION_INTERNAL DEC(__IAR_SYSTEMS_ICC__)
# elif defined(__VER__) && (defined(__ICCAVR__) || defined(__ICCRX__) || defined(__ICCRH850__) ||
defined(__ICCRL78__) || defined(__ICC430__) || defined(__ICCRISCV__) || defined(__ICCV850__) ||
defined(__ICC8051__) || defined(__ICCSTM8__))
#  define COMPILER_VERSION_MAJOR DEC((__VER__) / 100)
#  define COMPILER_VERSION_MINOR DEC((__VER__) - (((__VER__) / 100)*100))
#  define COMPILER_VERSION_PATCH DEC(__SUBVERSION__)
#  define COMPILER_VERSION_INTERNAL DEC(__IAR_SYSTEMS_ICC__)
# endif

#elif defined(__SDCC_VERSION_MAJOR) || defined(SDCC)
# define COMPILER_ID "SDCC"
# if defined(__SDCC_VERSION_MAJOR)
#  define COMPILER_VERSION_MAJOR DEC(__SDCC_VERSION_MAJOR)
#  define COMPILER_VERSION_MINOR DEC(__SDCC_VERSION_MINOR)
#  define COMPILER_VERSION_PATCH DEC(__SDCC_VERSION_PATCH)
# else
  /* SDCC = VRP */
#  define COMPILER_VERSION_MAJOR DEC(SDCC/100)
#  define COMPILER_VERSION_MINOR DEC(SDCC/10 % 10)
#  define COMPILER_VERSION_PATCH DEC(SDCC    % 10)
# endif


/* These compilers are either not known or too old to define an
  identification macro.  Try to identify the platform and guess that
  it is the native compiler.  */
#elif defined(__hpux) || defined(__hpua)
# define COMPILER_ID "HP"

#else /* unknown compiler */
# define COMPILER_ID ""
#endif

/* Construct the string literal in pieces to prevent the source from
   getting matched.  Store it in a pointer rather than an array
   because some compilers will just produce instructions to fill the
   array rather than assigning a pointer to a static array.  */
char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]";
#ifdef SIMULATE_ID
char const* info_simulate = "INFO" ":" "simulate[" SIMULATE_ID "]";
#endif

#ifdef __QNXNTO__
char const* qnxnto = "INFO" ":" "qnxnto[]";
#endif
```

```

#if defined(__CRAYXT_COMPUTE_LINUX_TARGET)
char const *info_cray = "INFO" ":" "compiler_wrapper[CrayPrgEnv]";
#endif

#define STRINGIFY_HELPER(X) #X
#define STRINGIFY(X) STRINGIFY_HELPER(X)

/* Identify known platforms by name.  */
#if defined(__linux) || defined(__linux__) || defined(linux)
# define PLATFORM_ID "Linux"

#elif defined(__MSYS__)
# define PLATFORM_ID "MSYS"

#elif defined(__CYGWIN__)
# define PLATFORM_ID "Cygwin"

#elif defined(__MINGW32__)
# define PLATFORM_ID "MinGW"

#elif defined(__APPLE__)
# define PLATFORM_ID "Darwin"

#elif defined(_WIN32) || defined(__WIN32__) || defined(WIN32)
# define PLATFORM_ID "Windows"

#elif defined(__FreeBSD__) || defined(__FreeBSD)
# define PLATFORM_ID "FreeBSD"

#elif defined(__NetBSD__) || defined(__NetBSD)
# define PLATFORM_ID "NetBSD"

#elif defined(__OpenBSD__) || defined(__OPENBSD)
# define PLATFORM_ID "OpenBSD"

#elif defined(__sun) || defined(sun)
# define PLATFORM_ID "SunOS"

#elif defined(_AIX) || defined(__AIX) || defined(__AIX__) || defined(__aix) || defined(__aix__)
# define PLATFORM_ID "AIX"

#elif defined(__hpux) || defined(__hpux__)
# define PLATFORM_ID "HP-UX"

#elif defined(__HAIKU__)
# define PLATFORM_ID "Haiku"

#elif defined(__BeOS) || defined(__BEOS__) || defined(_BEOS)
# define PLATFORM_ID "BeOS"

#elif defined(__QNX__) || defined(__QNXNTO__)
# define PLATFORM_ID "QNX"

#elif defined(__tru64) || defined(_tru64) || defined(__TRU64__)
# define PLATFORM_ID "Tru64"

#elif defined(__riscos) || defined(__riscos__)
# define PLATFORM_ID "RISCos"

#elif defined(__sinix) || defined(__sinix__) || defined(__SINIX__)
# define PLATFORM_ID "SINIX"

#elif defined(__UNIX_SV__)
# define PLATFORM_ID "UNIX_SV"

#elif defined(__bsdos__)
# define PLATFORM_ID "BSDOS"

#elif defined(_MPRAS) || defined(MPRAS)
# define PLATFORM_ID "MP-RAS"

#elif defined(__osf) || defined(__osf__)
# define PLATFORM_ID "OSF1"

#elif defined(_SCO_SV) || defined(SCO_SV) || defined(sco_sv)
# define PLATFORM_ID "SCO_SV"

#elif defined(__ultrix) || defined(__ultrix__) || defined(_ULTRIX)
# define PLATFORM_ID "ULTRIX"

#elif defined(__XENIX__) || defined(_XENIX) || defined(XENIX)
# define PLATFORM_ID "Xenix"

#elif defined(__WATCOMC__)
# if defined(__LINUX__)
#  define PLATFORM_ID "Linux"

# elif defined(__DOS__)
#  define PLATFORM_ID "DOS"

# elif defined(__OS2__)
#  define PLATFORM_ID "OS2"

# elif defined(__WINDOWS__)
#  define PLATFORM_ID "Windows3x"

# elif defined(__VXWORKS__)
#  define PLATFORM_ID "VxWorks"

# else /* unknown platform */
#  define PLATFORM_ID
# endif

#elif defined(__INTEGRITY)
# if defined(INT_178B)
#  define PLATFORM_ID "Integrity178"

# else /* regular Integrity */
#  define PLATFORM_ID "Integrity"
# endif

# elif defined(_ADI_COMPILER)
#  define PLATFORM_ID "ADSP"

#else /* unknown platform */
# define PLATFORM_ID

#endif

/* For windows compilers MSVC and Intel we can determine
   the architecture of the compiler being used.  This is because
   the compilers do not have flags that can change the architecture,
   but rather depend on which compiler is being used
*/
#if defined(_WIN32) && defined(_MSC_VER)
# if defined(_M_IA64)
#  define ARCHITECTURE_ID "IA64"

# elif defined(_M_ARM64EC)
#  define ARCHITECTURE_ID "ARM64EC"

# elif defined(_M_X64) || defined(_M_AMD64)
#  define ARCHITECTURE_ID "x64"

# elif defined(_M_IX86)
#  define ARCHITECTURE_ID "X86"

# elif defined(_M_ARM64)
#  define ARCHITECTURE_ID "ARM64"

# elif defined(_M_ARM)
#  if _M_ARM == 4
#   define ARCHITECTURE_ID "ARMV4I"
#  elif _M_ARM == 5
#   define ARCHITECTURE_ID "ARMV5I"
#  else
#   define ARCHITECTURE_ID "ARMV" STRINGIFY(_M_ARM)
#  endif

# elif defined(_M_MIPS)
#  define ARCHITECTURE_ID "MIPS"

# elif defined(_M_SH)
#  define ARCHITECTURE_ID "SHx"

# else /* unknown architecture */
#  define ARCHITECTURE_ID ""
# endif

#elif defined(__WATCOMC__)
# if defined(_M_I86)
#  define ARCHITECTURE_ID "I86"

# elif defined(_M_IX86)
#  define ARCHITECTURE_ID "X86"

# else /* unknown architecture */
#  define ARCHITECTURE_ID ""
# endif

#elif defined(__IAR_SYSTEMS_ICC__) || defined(__IAR_SYSTEMS_ICC)
# if defined(__ICCARM__)
#  define ARCHITECTURE_ID "ARM"

# elif defined(__ICCRX__)
#  define ARCHITECTURE_ID "RX"

# elif defined(__ICCRH850__)
#  define ARCHITECTURE_ID "RH850"

# elif defined(__ICCRL78__)
#  define ARCHITECTURE_ID "RL78"

# elif defined(__ICCRISCV__)
#  define ARCHITECTURE_ID "RISCV"

# elif defined(__ICCAVR__)
#  define ARCHITECTURE_ID "AVR"

# elif defined(__ICC430__)
#  define ARCHITECTURE_ID "MSP430"

# elif defined(__ICCV850__)
#  define ARCHITECTURE_ID "V850"

# elif defined(__ICC8051__)
#  define ARCHITECTURE_ID "8051"

# elif defined(__ICCSTM8__)
#  define ARCHITECTURE_ID "STM8"

# else /* unknown architecture */
#  define ARCHITECTURE_ID ""
# endif

#elif defined(__ghs__)
# if defined(__PPC64__)
#  define ARCHITECTURE_ID "PPC64"

# elif defined(__ppc__)
#  define ARCHITECTURE_ID "PPC"

# elif defined(__ARM__)
#  define ARCHITECTURE_ID "ARM"

# elif defined(__x86_64__)
#  define ARCHITECTURE_ID "x64"

# elif defined(__i386__)
#  define ARCHITECTURE_ID "X86"

# else /* unknown architecture */
#  define ARCHITECTURE_ID ""
# endif

#elif defined(__clang__) && defined(__ti__)
# if defined(__ARM_ARCH)
#  define ARCHITECTURE_ID "Arm"

# else /* unknown architecture */
#  define ARCHITECTURE_ID ""
# endif

#elif defined(__TI_COMPILER_VERSION__)
# if defined(__TI_ARM__)
#  define ARCHITECTURE_ID "ARM"

# elif defined(__MSP430__)
#  define ARCHITECTURE_ID "MSP430"

# elif defined(__TMS320C28XX__)
#  define ARCHITECTURE_ID "TMS320C28x"

# elif defined(__TMS320C6X__) || defined(_TMS320C6X)
#  define ARCHITECTURE_ID "TMS320C6x"

# else /* unknown architecture */
#  define ARCHITECTURE_ID ""
# endif

# elif defined(__ADSPSHARC__)
#  define ARCHITECTURE_ID "SHARC"

# elif defined(__ADSPBLACKFIN__)
#  define ARCHITECTURE_ID "Blackfin"

#elif defined(__TASKING__)

# if defined(__CTC__) || defined(__CPTC__)
#  define ARCHITECTURE_ID "TriCore"

# elif defined(__CMCS__)
#  define ARCHITECTURE_ID "MCS"

# elif defined(__CARM__)
#  define ARCHITECTURE_ID "ARM"

# elif defined(__CARC__)
#  define ARCHITECTURE_ID "ARC"

# elif defined(__C51__)
#  define ARCHITECTURE_ID "8051"

# elif defined(__CPCP__)
#  define ARCHITECTURE_ID "PCP"

# else
#  define ARCHITECTURE_ID ""
# endif

#else
#  define ARCHITECTURE_ID
#endif

/* Convert integer to decimal digit literals.  */
#define DEC(n)                   \
  ('0' + (((n) / 10000000)%10)), \
  ('0' + (((n) / 1000000)%10)),  \
  ('0' + (((n) / 100000)%10)),   \
  ('0' + (((n) / 10000)%10)),    \
  ('0' + (((n) / 1000)%10)),     \
  ('0' + (((n) / 100)%10)),      \
  ('0' + (((n) / 10)%10)),       \
  ('0' +  ((n) % 10))

/* Convert integer to hex digit literals.  */
#define HEX(n)             \
  ('0' + ((n)»28 & 0xF)), \
  ('0' + ((n)»24 & 0xF)), \
  ('0' + ((n)»20 & 0xF)), \
  ('0' + ((n)»16 & 0xF)), \
  ('0' + ((n)»12 & 0xF)), \
  ('0' + ((n)»8  & 0xF)), \
  ('0' + ((n)»4  & 0xF)), \
  ('0' + ((n)     & 0xF))

/* Construct a string literal encoding the version number. */
#ifdef COMPILER_VERSION
char const* info_version = "INFO" ":" "compiler_version[" COMPILER_VERSION "]";

/* Construct a string literal encoding the version number components. */
#elif defined(COMPILER_VERSION_MAJOR)
char const info_version[] = {
  'I', 'N', 'F', 'O', ':',
  'c','o','m','p','i','l','e','r','_','v','e','r','s','i','o','n','[',
  COMPILER_VERSION_MAJOR,
# ifdef COMPILER_VERSION_MINOR
  '.', COMPILER_VERSION_MINOR,
#  ifdef COMPILER_VERSION_PATCH
   '.', COMPILER_VERSION_PATCH,
#   ifdef COMPILER_VERSION_TWEAK
    '.', COMPILER_VERSION_TWEAK,
#   endif
#  endif
# endif
  ']','\0'};
#endif

/* Construct a string literal encoding the internal version number. */
#ifdef COMPILER_VERSION_INTERNAL
char const info_version_internal[] = {
  'I', 'N', 'F', 'O', ':',
  'c','o','m','p','i','l','e','r','_','v','e','r','s','i','o','n','_',
  'i','n','t','e','r','n','a','l','[',
  COMPILER_VERSION_INTERNAL,']','\0'};
#elif defined(COMPILER_VERSION_INTERNAL_STR)
char const* info_version_internal = "INFO" ":" "compiler_version_internal["
COMPILER_VERSION_INTERNAL_STR "]";
#endif

/* Construct a string literal encoding the version number components. */
#ifdef SIMULATE_VERSION_MAJOR
char const info_simulate_version[] = {
  'I', 'N', 'F', 'O', ':',
  's','i','m','u','l','a','t','e','_','v','e','r','s','i','o','n','[',
  SIMULATE_VERSION_MAJOR,
# ifdef SIMULATE_VERSION_MINOR
```

```
  '.', SIMULATE_VERSION_MINOR,
#  ifdef SIMULATE_VERSION_PATCH
   '.', SIMULATE_VERSION_PATCH,
#   ifdef SIMULATE_VERSION_TWEAK
    '.', SIMULATE_VERSION_TWEAK,
#   endif
#  endif
# endif
  ']','\0'};
#endif

/* Construct the string literal in pieces to prevent the source from
   getting matched.  Store it in a pointer rather than an array
   because some compilers will just produce instructions to fill the
   array rather than assigning a pointer to a static array.  */
char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]";
char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]";



#if !defined(__STDC__) && !defined(__clang__)
# if defined(_MSC_VER) || defined(__ibmxl__) || defined(__IBMC__)
#  define C_VERSION "90"
# else
#  define C_VERSION
# endif
#elif __STDC_VERSION__ > 201710L
# define C_VERSION "23"
#elif __STDC_VERSION__ >= 201710L
# define C_VERSION "17"
#elif __STDC_VERSION__ >= 201000L
# define C_VERSION "11"
#elif __STDC_VERSION__ >= 199901L
# define C_VERSION "99"
#else
# define C_VERSION "90"
#endif
const char* info_language_standard_default =
  "INFO" ":" "standard_default[" C_VERSION "]";

const char* info_language_extensions_default = "INFO" ":" "extensions_default["
#if (defined(__clang__) || defined(__GNUC__) || defined(__xlC__) ||           \
     defined(__TI_COMPILER_VERSION__)) &&                                     \
  !defined(__STRICT_ANSI__)
  "ON"
#else
  "OFF"
#endif
"]";

/*--------------------------------------------------------------------------*/

#ifdef ID_VOID_MAIN
void main() {}
#else
# if defined(__CLASSIC_C__)
int main(argc, argv) int argc; char *argv[];
# else
int main(int argc, char* argv[])
# endif
{
  int require = 0;
  require += info_compiler[argc];
  require += info_platform[argc];
  require += info_arch[argc];
#ifdef COMPILER_VERSION_MAJOR
  require += info_version[argc];
#endif
#ifdef COMPILER_VERSION_INTERNAL
  require += info_version_internal[argc];
#endif
#ifdef SIMULATE_ID
  require += info_simulate[argc];
#endif
#ifdef SIMULATE_VERSION_MAJOR
  require += info_simulate_version[argc];
#endif
#if defined(__CRAYXT_COMPUTE_LINUX_TARGET)
  require += info_cray[argc];
#endif
  require += info_language_standard_default[argc];
  require += info_language_extensions_default[argc];
  (void)argv;
  return require;
}
#endif
```

5.3 build/CMakeFiles/3.29.2/CompilerIdCXX/CMakeCXXCompilerId.cpp File Reference

Macros

- #define __has_include(x) 0
- #define COMPILER_ID ""
- #define STRINGIFY_HELPER(X) #X
- #define STRINGIFY(X) STRINGIFY_HELPER(X)
- #define PLATFORM_ID
- #define ARCHITECTURE_ID
- #define DEC(n)
- #define HEX(n)
- #define CXX_STD __cplusplus

Functions

- int main (int argc, char ∗argv[ ])

Variables

- char const ∗ info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
- char const ∗ info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
- char const ∗ info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
- const char ∗ info_language_standard_default
- const char ∗ info_language_extensions_default

5.3.1 Macro Definition Documentation

5.3.1.1 __has_include

```
#define __has_include(
              x ) 0
```

Definition at line 11 of file CMakeCXXCompilerId.cpp.

5.3.1.2 ARCHITECTURE_ID

```
#define ARCHITECTURE_ID
```

Definition at line 724 of file CMakeCXXCompilerId.cpp.

5.3.1.3 COMPILER_ID

```
#define COMPILER_ID ""
```

Definition at line 427 of file CMakeCXXCompilerId.cpp.

5.3.1.4 CXX_STD

```
#define CXX_STD __cplusplus
```

Definition at line 822 of file CMakeCXXCompilerId.cpp.

5.3.1.5 DEC

```
#define DEC(
               n )
```

Value:
```
  ('0' + (((n) / 10000000)%10)), \
  ('0' + (((n) / 1000000)%10)),  \
  ('0' + (((n) / 100000)%10)),   \
  ('0' + (((n) / 10000)%10)),    \
  ('0' + (((n) / 1000)%10)),     \
  ('0' + (((n) / 100)%10)),      \
  ('0' + (((n) / 10)%10)),       \
  ('0' +  ((n) % 10))
```

Definition at line 728 of file CMakeCXXCompilerId.cpp.

5.3.1.6 HEX

```
#define HEX(
               n )
```

Value:
```
  ('0' + ((n)»28 & 0xF)), \
  ('0' + ((n)»24 & 0xF)), \
  ('0' + ((n)»20 & 0xF)), \
  ('0' + ((n)»16 & 0xF)), \
  ('0' + ((n)»12 & 0xF)), \
  ('0' + ((n)»8  & 0xF)), \
  ('0' + ((n)»4  & 0xF)), \
  ('0' + ((n)     & 0xF))
```

Definition at line 739 of file CMakeCXXCompilerId.cpp.

5.3.1.7 PLATFORM_ID

```
#define PLATFORM_ID
```

Definition at line 558 of file CMakeCXXCompilerId.cpp.

5.3.1.8 STRINGIFY

```
#define STRINGIFY(
               X ) STRINGIFY_HELPER(X)
```

Definition at line 448 of file CMakeCXXCompilerId.cpp.

5.3.1.9 STRINGIFY_HELPER

```
#define STRINGIFY_HELPER(
              X ) #X
```

Definition at line 447 of file CMakeCXXCompilerId.cpp.

5.3.2 Function Documentation

5.3.2.1 main()

```
int main (
             int argc,
             char * argv[] )
```

Definition at line 853 of file CMakeCXXCompilerId.cpp.

5.3.3 Variable Documentation

5.3.3.1 info_arch

```
char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
```

Definition at line 805 of file CMakeCXXCompilerId.cpp.

5.3.3.2 info_compiler

```
char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
```

Definition at line 434 of file CMakeCXXCompilerId.cpp.

5.3.3.3 info_language_extensions_default

```
const char* info_language_extensions_default
```

Initial value:

```
= "INFO" ":" "extensions_default["




  "OFF"

"]"
```

Definition at line 841 of file CMakeCXXCompilerId.cpp.

5.3.3.4 info_language_standard_default

```
const char* info_language_standard_default
```

Initial value:

```
= "INFO" ":" "standard_default["









  "98"

"]"
```

Definition at line 825 of file CMakeCXXCompilerId.cpp.

5.3.3.5 info_platform

```
char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
```

Definition at line 804 of file CMakeCXXCompilerId.cpp.

5.4 CMakeCXXCompilerId.cpp

Go to the documentation of this file.

```
/* This source file must have a .cpp extension so that all C++ compilers
   recognize the extension without flags.  Borland does not know .cxx for
   example.  */
#ifndef __cplusplus
# error "A C compiler has been selected for C++."
#endif

#if !defined(__has_include)
/* If the compiler does not have __has_include, pretend the answer is
   always no.  */
#  define __has_include(x) 0
#endif


/* Version number components: V=Version, R=Revision, P=Patch
   Version date components:   YYYY=Year, MM=Month,   DD=Day  */

#if defined(__INTEL_COMPILER) || defined(__ICC)
# define COMPILER_ID "Intel"
# if defined(_MSC_VER)
#  define SIMULATE_ID "MSVC"
# endif
# if defined(__GNUC__)
#  define SIMULATE_ID "GNU"
# endif
  /* __INTEL_COMPILER = VRP prior to 2021, and then VVVV for 2021 and later,
     except that a few beta releases use the old format with V=2021.  */
# if __INTEL_COMPILER < 2021 || __INTEL_COMPILER == 202110 || __INTEL_COMPILER == 202111
#  define COMPILER_VERSION_MAJOR DEC(__INTEL_COMPILER/100)
#  define COMPILER_VERSION_MINOR DEC(__INTEL_COMPILER/10 % 10)
#  if defined(__INTEL_COMPILER_UPDATE)
#   define COMPILER_VERSION_PATCH DEC(__INTEL_COMPILER_UPDATE)
#  else
#   define COMPILER_VERSION_PATCH DEC(__INTEL_COMPILER   % 10)
#  endif
# else
#  define COMPILER_VERSION_MAJOR DEC(__INTEL_COMPILER)
#  define COMPILER_VERSION_MINOR DEC(__INTEL_COMPILER_UPDATE)
   /* The third version component from --version is an update index,
      but no macro is provided for it.  */
```

```
#  define COMPILER_VERSION_PATCH DEC(0)
# endif
# if defined(__INTEL_COMPILER_BUILD_DATE)
   /* __INTEL_COMPILER_BUILD_DATE = YYYYMMDD */
#  define COMPILER_VERSION_TWEAK DEC(__INTEL_COMPILER_BUILD_DATE)
# endif
# if defined(_MSC_VER)
   /* _MSC_VER = VVRR */
#  define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
#  define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
# endif
# if defined(__GNUC__)
#  define SIMULATE_VERSION_MAJOR DEC(__GNUC__)
# elif defined(__GNUG__)
#  define SIMULATE_VERSION_MAJOR DEC(__GNUG__)
# endif
# if defined(__GNUC_MINOR__)
#  define SIMULATE_VERSION_MINOR DEC(__GNUC_MINOR__)
# endif
# if defined(__GNUC_PATCHLEVEL__)
#  define SIMULATE_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
# endif

#elif (defined(__clang__) && defined(__INTEL_CLANG_COMPILER)) || defined(__INTEL_LLVM_COMPILER)
# define COMPILER_ID "IntelLLVM"
#if defined(_MSC_VER)
# define SIMULATE_ID "MSVC"
#endif
#if defined(__GNUC__)
# define SIMULATE_ID "GNU"
#endif
/* __INTEL_LLVM_COMPILER = VVVVRP prior to 2021.2.0, VVVVRRPP for 2021.2.0 and
 * later.  Look for 6 digit vs. 8 digit version number to decide encoding.
 * VVVV is no smaller than the current year when a version is released.
 */
#if __INTEL_LLVM_COMPILER < 1000000L
# define COMPILER_VERSION_MAJOR DEC(__INTEL_LLVM_COMPILER/100)
# define COMPILER_VERSION_MINOR DEC(__INTEL_LLVM_COMPILER/10 % 10)
# define COMPILER_VERSION_PATCH DEC(__INTEL_LLVM_COMPILER    % 10)
#else
# define COMPILER_VERSION_MAJOR DEC(__INTEL_LLVM_COMPILER/10000)
# define COMPILER_VERSION_MINOR DEC(__INTEL_LLVM_COMPILER/100 % 100)
# define COMPILER_VERSION_PATCH DEC(__INTEL_LLVM_COMPILER     % 100)
#endif
#if defined(_MSC_VER)
  /* _MSC_VER = VVRR */
# define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
# define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
#endif
#if defined(__GNUC__)
# define SIMULATE_VERSION_MAJOR DEC(__GNUC__)
#elif defined(__GNUG__)
# define SIMULATE_VERSION_MAJOR DEC(__GNUG__)
#endif
#if defined(__GNUC_MINOR__)
# define SIMULATE_VERSION_MINOR DEC(__GNUC_MINOR__)
#endif
#if defined(__GNUC_PATCHLEVEL__)
# define SIMULATE_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
#endif

#elif defined(__PATHCC__)
# define COMPILER_ID "PathScale"
# define COMPILER_VERSION_MAJOR DEC(__PATHCC__)
# define COMPILER_VERSION_MINOR DEC(__PATHCC_MINOR__)
# if defined(__PATHCC_PATCHLEVEL__)
#  define COMPILER_VERSION_PATCH DEC(__PATHCC_PATCHLEVEL__)
# endif

#elif defined(__BORLANDC__) && defined(__CODEGEARC_VERSION__)
# define COMPILER_ID "Embarcadero"
# define COMPILER_VERSION_MAJOR HEX(__CODEGEARC_VERSION__»24 & 0x00FF)
# define COMPILER_VERSION_MINOR HEX(__CODEGEARC_VERSION__»16 & 0x00FF)
# define COMPILER_VERSION_PATCH DEC(__CODEGEARC_VERSION__     & 0xFFFF)

#elif defined(__BORLANDC__)
# define COMPILER_ID "Borland"
  /* __BORLANDC__ = 0xVRR */
# define COMPILER_VERSION_MAJOR HEX(__BORLANDC__»8)
# define COMPILER_VERSION_MINOR HEX(__BORLANDC__ & 0xFF)

#elif defined(__WATCOMC__) && __WATCOMC__ < 1200
# define COMPILER_ID "Watcom"
   /* __WATCOMC__ = VVRR */
# define COMPILER_VERSION_MAJOR DEC(__WATCOMC__ / 100)
# define COMPILER_VERSION_MINOR DEC((__WATCOMC__ / 10) % 10)
# if (__WATCOMC__ % 10) > 0
#  define COMPILER_VERSION_PATCH DEC(__WATCOMC__ % 10)
# endif

#elif defined(__WATCOMC__)
# define COMPILER_ID "OpenWatcom"
   /* __WATCOMC__ = VVRP + 1100 */
# define COMPILER_VERSION_MAJOR DEC((__WATCOMC__ - 1100) / 100)
# define COMPILER_VERSION_MINOR DEC((__WATCOMC__ / 10) % 10)
# if (__WATCOMC__ % 10) > 0
#  define COMPILER_VERSION_PATCH DEC(__WATCOMC__ % 10)
# endif

#elif defined(__SUNPRO_CC)
# define COMPILER_ID "SunPro"
# if __SUNPRO_CC >= 0x5100
   /* __SUNPRO_CC = 0xVRRP */
#  define COMPILER_VERSION_MAJOR HEX(__SUNPRO_CC»12)
#  define COMPILER_VERSION_MINOR HEX(__SUNPRO_CC»4 & 0xFF)
#  define COMPILER_VERSION_PATCH HEX(__SUNPRO_CC    & 0xF)
# else
   /* __SUNPRO_CC = 0xVRP */
#  define COMPILER_VERSION_MAJOR HEX(__SUNPRO_CC»8)
#  define COMPILER_VERSION_MINOR HEX(__SUNPRO_CC»4 & 0xF)
#  define COMPILER_VERSION_PATCH HEX(__SUNPRO_CC    & 0xF)
# endif

#elif defined(__HP_aCC)
# define COMPILER_ID "HP"
  /* __HP_aCC = VVRRPP */
# define COMPILER_VERSION_MAJOR DEC(__HP_aCC/10000)
# define COMPILER_VERSION_MINOR DEC(__HP_aCC/100 % 100)
# define COMPILER_VERSION_PATCH DEC(__HP_aCC     % 100)

#elif defined(__DECCXX)
# define COMPILER_ID "Compaq"
  /* __DECCXX_VER = VVRRTPPPP */
# define COMPILER_VERSION_MAJOR DEC(__DECCXX_VER/10000000)
# define COMPILER_VERSION_MINOR DEC(__DECCXX_VER/100000  % 100)
# define COMPILER_VERSION_PATCH DEC(__DECCXX_VER         % 10000)

#elif defined(__IBMCPP__) && defined(__COMPILER_VER__)
# define COMPILER_ID "zOS"
  /* __IBMCPP__ = VRP */
# define COMPILER_VERSION_MAJOR DEC(__IBMCPP__/100)
# define COMPILER_VERSION_MINOR DEC(__IBMCPP__/10 % 10)
# define COMPILER_VERSION_PATCH DEC(__IBMCPP__    % 10)

#elif defined(__open_xl__) && defined(__clang__)
# define COMPILER_ID "IBMClang"
# define COMPILER_VERSION_MAJOR DEC(__open_xl_version__)
# define COMPILER_VERSION_MINOR DEC(__open_xl_release__)
# define COMPILER_VERSION_PATCH DEC(__open_xl_modification__)
# define COMPILER_VERSION_TWEAK DEC(__open_xl_ptf_fix_level__)


#elif defined(__ibmxl__) && defined(__clang__)
# define COMPILER_ID "XLClang"
# define COMPILER_VERSION_MAJOR DEC(__ibmxl_version__)
# define COMPILER_VERSION_MINOR DEC(__ibmxl_release__)
# define COMPILER_VERSION_PATCH DEC(__ibmxl_modification__)
# define COMPILER_VERSION_TWEAK DEC(__ibmxl_ptf_fix_level__)


#elif defined(__IBMCPP__) && !defined(__COMPILER_VER__) && __IBMCPP__ >= 800
# define COMPILER_ID "XL"
  /* __IBMCPP__ = VRP */
# define COMPILER_VERSION_MAJOR DEC(__IBMCPP__/100)
# define COMPILER_VERSION_MINOR DEC(__IBMCPP__/10 % 10)
# define COMPILER_VERSION_PATCH DEC(__IBMCPP__    % 10)

#elif defined(__IBMCPP__) && !defined(__COMPILER_VER__) && __IBMCPP__ < 800
# define COMPILER_ID "VisualAge"
  /* __IBMCPP__ = VRP */
# define COMPILER_VERSION_MAJOR DEC(__IBMCPP__/100)
# define COMPILER_VERSION_MINOR DEC(__IBMCPP__/10 % 10)
# define COMPILER_VERSION_PATCH DEC(__IBMCPP__    % 10)

#elif defined(__NVCOMPILER)
# define COMPILER_ID "NVHPC"
# define COMPILER_VERSION_MAJOR DEC(__NVCOMPILER_MAJOR__)
# define COMPILER_VERSION_MINOR DEC(__NVCOMPILER_MINOR__)
# if defined(__NVCOMPILER_PATCHLEVEL__)
#  define COMPILER_VERSION_PATCH DEC(__NVCOMPILER_PATCHLEVEL__)
# endif

#elif defined(__PGI)
# define COMPILER_ID "PGI"
# define COMPILER_VERSION_MAJOR DEC(__PGIC__)
# define COMPILER_VERSION_MINOR DEC(__PGIC_MINOR__)
# if defined(__PGIC_PATCHLEVEL__)
#  define COMPILER_VERSION_PATCH DEC(__PGIC_PATCHLEVEL__)
# endif

#elif defined(__clang__) && defined(__cray__)
# define COMPILER_ID "CrayClang"
# define COMPILER_VERSION_MAJOR DEC(__cray_major__)
# define COMPILER_VERSION_MINOR DEC(__cray_minor__)
# define COMPILER_VERSION_PATCH DEC(__cray_patchlevel__)
# define COMPILER_VERSION_INTERNAL_STR __clang_version__


#elif defined(_CRAYC)
# define COMPILER_ID "Cray"
# define COMPILER_VERSION_MAJOR DEC(_RELEASE_MAJOR)
# define COMPILER_VERSION_MINOR DEC(_RELEASE_MINOR)

#elif defined(__TI_COMPILER_VERSION__)
# define COMPILER_ID "TI"
  /* __TI_COMPILER_VERSION__ = VVVRRRPPP */
# define COMPILER_VERSION_MAJOR DEC(__TI_COMPILER_VERSION__/1000000)
# define COMPILER_VERSION_MINOR DEC(__TI_COMPILER_VERSION__/1000   % 1000)
# define COMPILER_VERSION_PATCH DEC(__TI_COMPILER_VERSION__        % 1000)

#elif defined(__CLANG_FUJITSU)
# define COMPILER_ID "FujitsuClang"
# define COMPILER_VERSION_MAJOR DEC(__FCC_major__)
# define COMPILER_VERSION_MINOR DEC(__FCC_minor__)
# define COMPILER_VERSION_PATCH DEC(__FCC_patchlevel__)
# define COMPILER_VERSION_INTERNAL_STR __clang_version__


#elif defined(__FUJITSU)
# define COMPILER_ID "Fujitsu"
# if defined(__FCC_version__)
#   define COMPILER_VERSION __FCC_version__
# elif defined(__FCC_major__)
#   define COMPILER_VERSION_MAJOR DEC(__FCC_major__)
#   define COMPILER_VERSION_MINOR DEC(__FCC_minor__)
#   define COMPILER_VERSION_PATCH DEC(__FCC_patchlevel__)
# endif
# if defined(__fcc_version)
#   define COMPILER_VERSION_INTERNAL DEC(__fcc_version)
# elif defined(__FCC_VERSION)
#   define COMPILER_VERSION_INTERNAL DEC(__FCC_VERSION)
# endif


#elif defined(__ghs__)
# define COMPILER_ID "GHS"
/* __GHS_VERSION_NUMBER = VVVVRP */
# ifdef __GHS_VERSION_NUMBER
# define COMPILER_VERSION_MAJOR DEC(__GHS_VERSION_NUMBER / 100)
# define COMPILER_VERSION_MINOR DEC(__GHS_VERSION_NUMBER / 10 % 10)
# define COMPILER_VERSION_PATCH DEC(__GHS_VERSION_NUMBER      % 10)
# endif

#elif defined(__TASKING__)
# define COMPILER_ID "Tasking"
  # define COMPILER_VERSION_MAJOR DEC(__VERSION__/1000)
  # define COMPILER_VERSION_MINOR DEC(__VERSION__ % 100)
# define COMPILER_VERSION_INTERNAL DEC(__VERSION__)

#elif defined(__ORANGEC__)
# define COMPILER_ID "OrangeC"
# define COMPILER_VERSION_MAJOR DEC(__ORANGEC_MAJOR__)
# define COMPILER_VERSION_MINOR DEC(__ORANGEC_MINOR__)
# define COMPILER_VERSION_PATCH DEC(__ORANGEC_PATCHLEVEL__)

#elif defined(__SCO_VERSION__)
# define COMPILER_ID "SCO"

#elif defined(__ARMCC_VERSION) && !defined(__clang__)
# define COMPILER_ID "ARMCC"
#if __ARMCC_VERSION >= 1000000
  /* __ARMCC_VERSION = VRRPPPP */
  # define COMPILER_VERSION_MAJOR DEC(__ARMCC_VERSION/1000000)
  # define COMPILER_VERSION_MINOR DEC(__ARMCC_VERSION/10000 % 100)
  # define COMPILER_VERSION_PATCH DEC(__ARMCC_VERSION     % 10000)
#else
  /* __ARMCC_VERSION = VRPPPP */
  # define COMPILER_VERSION_MAJOR DEC(__ARMCC_VERSION/100000)
  # define COMPILER_VERSION_MINOR DEC(__ARMCC_VERSION/10000 % 10)
  # define COMPILER_VERSION_PATCH DEC(__ARMCC_VERSION    % 10000)
#endif
```

```


#elif defined(__clang__) && defined(__apple_build_version__)
# define COMPILER_ID "AppleClang"
# if defined(_MSC_VER)
#  define SIMULATE_ID "MSVC"
# endif
# define COMPILER_VERSION_MAJOR DEC(__clang_major__)
# define COMPILER_VERSION_MINOR DEC(__clang_minor__)
# define COMPILER_VERSION_PATCH DEC(__clang_patchlevel__)
# if defined(_MSC_VER)
   /* _MSC_VER = VVRR */
#  define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
#  define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
# endif
# define COMPILER_VERSION_TWEAK DEC(__apple_build_version__)

#elif defined(__clang__) && defined(__ARMCOMPILER_VERSION)
# define COMPILER_ID "ARMClang"
  # define COMPILER_VERSION_MAJOR DEC(__ARMCOMPILER_VERSION/1000000)
  # define COMPILER_VERSION_MINOR DEC(__ARMCOMPILER_VERSION/10000 % 100)
  # define COMPILER_VERSION_PATCH DEC(__ARMCOMPILER_VERSION/100   % 100)
# define COMPILER_VERSION_INTERNAL DEC(__ARMCOMPILER_VERSION)

#elif defined(__clang__) && defined(__ti__)
# define COMPILER_ID "TIClang"
  # define COMPILER_VERSION_MAJOR DEC(__ti_major__)
  # define COMPILER_VERSION_MINOR DEC(__ti_minor__)
  # define COMPILER_VERSION_PATCH DEC(__ti_patchlevel__)
# define COMPILER_VERSION_INTERNAL DEC(__ti_version__)

#elif defined(__clang__)
# define COMPILER_ID "Clang"
# if defined(_MSC_VER)
#  define SIMULATE_ID "MSVC"
# endif
# define COMPILER_VERSION_MAJOR DEC(__clang_major__)
# define COMPILER_VERSION_MINOR DEC(__clang_minor__)
# define COMPILER_VERSION_PATCH DEC(__clang_patchlevel__)
# if defined(_MSC_VER)
   /* _MSC_VER = VVRR */
#  define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
#  define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
# endif

#elif defined(__LCC__) && (defined(__GNUC__) || defined(__GNUG__) || defined(__MCST__))
# define COMPILER_ID "LCC"
# define COMPILER_VERSION_MAJOR DEC(__LCC__ / 100)
# define COMPILER_VERSION_MINOR DEC(__LCC__ % 100)
# if defined(__LCC_MINOR__)
#  define COMPILER_VERSION_PATCH DEC(__LCC_MINOR__)
# endif
# if defined(__GNUC__) && defined(__GNUC_MINOR__)
#  define SIMULATE_ID "GNU"
#  define SIMULATE_VERSION_MAJOR DEC(__GNUC__)
#  define SIMULATE_VERSION_MINOR DEC(__GNUC_MINOR__)
#  if defined(__GNUC_PATCHLEVEL__)
#   define SIMULATE_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
#  endif
# endif

#elif defined(__GNUC__) || defined(__GNUG__)
# define COMPILER_ID "GNU"
# if defined(__GNUC__)
#  define COMPILER_VERSION_MAJOR DEC(__GNUC__)
# else
#  define COMPILER_VERSION_MAJOR DEC(__GNUG__)
# endif
# if defined(__GNUC_MINOR__)
#  define COMPILER_VERSION_MINOR DEC(__GNUC_MINOR__)
# endif
# if defined(__GNUC_PATCHLEVEL__)
#  define COMPILER_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
# endif

#elif defined(_MSC_VER)
# define COMPILER_ID "MSVC"
  /* _MSC_VER = VVRR */
# define COMPILER_VERSION_MAJOR DEC(_MSC_VER / 100)
# define COMPILER_VERSION_MINOR DEC(_MSC_VER % 100)
# if defined(_MSC_FULL_VER)
#  if _MSC_VER >= 1400
    /* _MSC_FULL_VER = VVRRPPPPP */
#   define COMPILER_VERSION_PATCH DEC(_MSC_FULL_VER % 100000)
#  else
    /* _MSC_FULL_VER = VVRRPPPP */
#   define COMPILER_VERSION_PATCH DEC(_MSC_FULL_VER % 10000)
#  endif
# endif
# if defined(_MSC_BUILD)
#  define COMPILER_VERSION_TWEAK DEC(_MSC_BUILD)
# endif

#elif defined(_ADI_COMPILER)
# define COMPILER_ID "ADSP"
#if defined(__VERSIONNUM__)
  /* __VERSIONNUM__ = 0xVVRRPPTT */
#  define COMPILER_VERSION_MAJOR DEC(__VERSIONNUM__ » 24 & 0xFF)
#  define COMPILER_VERSION_MINOR DEC(__VERSIONNUM__ » 16 & 0xFF)
#  define COMPILER_VERSION_PATCH DEC(__VERSIONNUM__ » 8 & 0xFF)
#  define COMPILER_VERSION_TWEAK DEC(__VERSIONNUM__ & 0xFF)
#endif

#elif defined(__IAR_SYSTEMS_ICC__) || defined(__IAR_SYSTEMS_ICC)
# define COMPILER_ID "IAR"
# if defined(__VER__) && defined(__ICCARM__)
#  define COMPILER_VERSION_MAJOR DEC((__VER__) / 1000000)
#  define COMPILER_VERSION_MINOR DEC(((__VER__) / 1000) % 1000)
#  define COMPILER_VERSION_PATCH DEC((__VER__) % 1000)
#  define COMPILER_VERSION_INTERNAL DEC(__IAR_SYSTEMS_ICC__)
# elif defined(__VER__) && (defined(__ICCAVR__) || defined(__ICCRX__) || defined(__ICCRH850__) ||
      defined(__ICCRL78__) || defined(__ICC430__) || defined(__ICCRISCV__) || defined(__ICCV850__) ||
      defined(__ICC8051__) || defined(__ICCSTM8__))
#  define COMPILER_VERSION_MAJOR DEC((__VER__) / 100)
#  define COMPILER_VERSION_MINOR DEC((__VER__) - (((__VER__) / 100)*100))
#  define COMPILER_VERSION_PATCH DEC(__SUBVERSION__)
#  define COMPILER_VERSION_INTERNAL DEC(__IAR_SYSTEMS_ICC__)
# endif


/* These compilers are either not known or too old to define an
  identification macro.  Try to identify the platform and guess that
  it is the native compiler.  */
#elif defined(__hpux) || defined(__hpua)
# define COMPILER_ID "HP"

#else /* unknown compiler */
# define COMPILER_ID ""
#endif

/* Construct the string literal in pieces to prevent the source from
   getting matched.  Store it in a pointer rather than an array
   because some compilers will just produce instructions to fill the
   array rather than assigning a pointer to a static array.  */
char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]";
#ifdef SIMULATE_ID
char const* info_simulate = "INFO" ":" "simulate[" SIMULATE_ID "]";
#endif

#ifdef __QNXNTO__
char const* qnxnto = "INFO" ":" "qnxnto[]";
#endif

#if defined(__CRAYXT_COMPUTE_LINUX_TARGET)
char const *info_cray = "INFO" ":" "compiler_wrapper[CrayPrgEnv]";
#endif

#define STRINGIFY_HELPER(X) #X
#define STRINGIFY(X) STRINGIFY_HELPER(X)

/* Identify known platforms by name.  */
#if defined(__linux) || defined(__linux__) || defined(linux)
# define PLATFORM_ID "Linux"

#elif defined(__MSYS__)
# define PLATFORM_ID "MSYS"

#elif defined(__CYGWIN__)
# define PLATFORM_ID "Cygwin"

#elif defined(__MINGW32__)
# define PLATFORM_ID "MinGW"

#elif defined(__APPLE__)
# define PLATFORM_ID "Darwin"

#elif defined(_WIN32) || defined(__WIN32__) || defined(WIN32)
# define PLATFORM_ID "Windows"

#elif defined(__FreeBSD__) || defined(__FreeBSD)
# define PLATFORM_ID "FreeBSD"

#elif defined(__NetBSD__) || defined(__NetBSD)
# define PLATFORM_ID "NetBSD"
```

```

#elif defined(__OpenBSD__) || defined(__OPENBSD)
# define PLATFORM_ID "OpenBSD"

#elif defined(__sun) || defined(sun)
# define PLATFORM_ID "SunOS"

#elif defined(_AIX) || defined(__AIX) || defined(__AIX__) || defined(__aix) || defined(__aix__)
# define PLATFORM_ID "AIX"

#elif defined(__hpux) || defined(__hpux__)
# define PLATFORM_ID "HP-UX"

#elif defined(__HAIKU__)
# define PLATFORM_ID "Haiku"

#elif defined(__BeOS) || defined(__BEOS__) || defined(_BEOS)
# define PLATFORM_ID "BeOS"

#elif defined(__QNX__) || defined(__QNXNTO__)
# define PLATFORM_ID "QNX"

#elif defined(__tru64) || defined(_tru64) || defined(__TRU64__)
# define PLATFORM_ID "Tru64"

#elif defined(__riscos) || defined(__riscos__)
# define PLATFORM_ID "RISCos"

#elif defined(__sinix) || defined(__sinix__) || defined(__SINIX__)
# define PLATFORM_ID "SINIX"

#elif defined(__UNIX_SV__)
# define PLATFORM_ID "UNIX_SV"

#elif defined(__bsdos__)
# define PLATFORM_ID "BSDOS"

#elif defined(_MPRAS) || defined(MPRAS)
# define PLATFORM_ID "MP-RAS"

#elif defined(__osf) || defined(__osf__)
# define PLATFORM_ID "OSF1"

#elif defined(_SCO_SV) || defined(SCO_SV) || defined(sco_sv)
# define PLATFORM_ID "SCO_SV"

#elif defined(__ultrix) || defined(__ultrix__) || defined(_ULTRIX)
# define PLATFORM_ID "ULTRIX"

#elif defined(__XENIX__) || defined(_XENIX) || defined(XENIX)
# define PLATFORM_ID "Xenix"

#elif defined(__WATCOMC__)
# if defined(__LINUX__)
#  define PLATFORM_ID "Linux"

# elif defined(__DOS__)
#  define PLATFORM_ID "DOS"

# elif defined(__OS2__)
#  define PLATFORM_ID "OS2"

# elif defined(__WINDOWS__)
#  define PLATFORM_ID "Windows3x"

# elif defined(__VXWORKS__)
#  define PLATFORM_ID "VxWorks"

# else /* unknown platform */
#  define PLATFORM_ID
# endif

#elif defined(__INTEGRITY)
# if defined(INT_178B)
#  define PLATFORM_ID "Integrity178"

# else /* regular Integrity */
#  define PLATFORM_ID "Integrity"
# endif

# elif defined(_ADI_COMPILER)
#  define PLATFORM_ID "ADSP"

#else /* unknown platform */
# define PLATFORM_ID

#endif
```

```

/* For windows compilers MSVC and Intel we can determine
   the architecture of the compiler being used.  This is because
   the compilers do not have flags that can change the architecture,
   but rather depend on which compiler is being used
*/
#if defined(_WIN32) && defined(_MSC_VER)
# if defined(_M_IA64)
#  define ARCHITECTURE_ID "IA64"

# elif defined(_M_ARM64EC)
#  define ARCHITECTURE_ID "ARM64EC"

# elif defined(_M_X64) || defined(_M_AMD64)
#  define ARCHITECTURE_ID "x64"

# elif defined(_M_IX86)
#  define ARCHITECTURE_ID "X86"

# elif defined(_M_ARM64)
#  define ARCHITECTURE_ID "ARM64"

# elif defined(_M_ARM)
#  if _M_ARM == 4
#   define ARCHITECTURE_ID "ARMV4I"
#  elif _M_ARM == 5
#   define ARCHITECTURE_ID "ARMV5I"
#  else
#   define ARCHITECTURE_ID "ARMV" STRINGIFY(_M_ARM)
#  endif

# elif defined(_M_MIPS)
#  define ARCHITECTURE_ID "MIPS"

# elif defined(_M_SH)
#  define ARCHITECTURE_ID "SHx"

# else /* unknown architecture */
#  define ARCHITECTURE_ID ""
# endif

#elif defined(__WATCOMC__)
# if defined(_M_I86)
#  define ARCHITECTURE_ID "I86"

# elif defined(_M_IX86)
#  define ARCHITECTURE_ID "X86"

# else /* unknown architecture */
#  define ARCHITECTURE_ID ""
# endif

#elif defined(__IAR_SYSTEMS_ICC__) || defined(__IAR_SYSTEMS_ICC)
# if defined(__ICCARM__)
#  define ARCHITECTURE_ID "ARM"

# elif defined(__ICCRX__)
#  define ARCHITECTURE_ID "RX"

# elif defined(__ICCRH850__)
#  define ARCHITECTURE_ID "RH850"

# elif defined(__ICCRL78__)
#  define ARCHITECTURE_ID "RL78"

# elif defined(__ICCRISCV__)
#  define ARCHITECTURE_ID "RISCV"

# elif defined(__ICCAVR__)
#  define ARCHITECTURE_ID "AVR"

# elif defined(__ICC430__)
#  define ARCHITECTURE_ID "MSP430"

# elif defined(__ICCV850__)
#  define ARCHITECTURE_ID "V850"

# elif defined(__ICC8051__)
#  define ARCHITECTURE_ID "8051"

# elif defined(__ICCSTM8__)
#  define ARCHITECTURE_ID "STM8"

# else /* unknown architecture */
#  define ARCHITECTURE_ID ""
# endif

```

```
#elif defined(__ghs__)
# if defined(__PPC64__)
#  define ARCHITECTURE_ID "PPC64"

# elif defined(__ppc__)
#  define ARCHITECTURE_ID "PPC"

# elif defined(__ARM__)
#  define ARCHITECTURE_ID "ARM"

# elif defined(__x86_64__)
#  define ARCHITECTURE_ID "x64"

# elif defined(__i386__)
#  define ARCHITECTURE_ID "X86"

# else /* unknown architecture */
#  define ARCHITECTURE_ID ""
# endif

#elif defined(__clang__) && defined(__ti__)
# if defined(__ARM_ARCH)
#  define ARCHITECTURE_ID "Arm"

# else /* unknown architecture */
#  define ARCHITECTURE_ID ""
# endif

#elif defined(__TI_COMPILER_VERSION__)
# if defined(__TI_ARM__)
#  define ARCHITECTURE_ID "ARM"

# elif defined(__MSP430__)
#  define ARCHITECTURE_ID "MSP430"

# elif defined(__TMS320C28XX__)
#  define ARCHITECTURE_ID "TMS320C28x"

# elif defined(__TMS320C6X__) || defined(_TMS320C6X)
#  define ARCHITECTURE_ID "TMS320C6x"

# else /* unknown architecture */
#  define ARCHITECTURE_ID ""
# endif

# elif defined(__ADSPSHARC__)
#  define ARCHITECTURE_ID "SHARC"

# elif defined(__ADSPBLACKFIN__)
#  define ARCHITECTURE_ID "Blackfin"

#elif defined(__TASKING__)

# if defined(__CTC__) || defined(__CPTC__)
#  define ARCHITECTURE_ID "TriCore"

# elif defined(__CMCS__)
#  define ARCHITECTURE_ID "MCS"

# elif defined(__CARM__)
#  define ARCHITECTURE_ID "ARM"

# elif defined(__CARC__)
#  define ARCHITECTURE_ID "ARC"

# elif defined(__C51__)
#  define ARCHITECTURE_ID "8051"

# elif defined(__CPCP__)
#  define ARCHITECTURE_ID "PCP"

# else
#  define ARCHITECTURE_ID ""
# endif

#else
#  define ARCHITECTURE_ID
#endif

/* Convert integer to decimal digit literals.  */
#define DEC(n)                   \
  ('0' + (((n) / 10000000)%10)), \
  ('0' + (((n) / 1000000)%10)),  \
  ('0' + (((n) / 100000)%10)),   \
  ('0' + (((n) / 10000)%10)),    \
  ('0' + (((n) / 1000)%10)),     \
  ('0' + (((n) / 100)%10)),      \
  ('0' + (((n) / 10)%10)),       \
  ('0' +  ((n) % 10))

/* Convert integer to hex digit literals.  */
#define HEX(n)                \
  ('0' + ((n)»28 & 0xF)), \
  ('0' + ((n)»24 & 0xF)), \
  ('0' + ((n)»20 & 0xF)), \
  ('0' + ((n)»16 & 0xF)), \
  ('0' + ((n)»12 & 0xF)), \
  ('0' + ((n)»8  & 0xF)), \
  ('0' + ((n)»4  & 0xF)), \
  ('0' + ((n)     & 0xF))

/* Construct a string literal encoding the version number. */
#ifdef COMPILER_VERSION
char const* info_version = "INFO" ":" "compiler_version[" COMPILER_VERSION "]";

/* Construct a string literal encoding the version number components. */
#elif defined(COMPILER_VERSION_MAJOR)
char const info_version[] = {
  'I', 'N', 'F', 'O', ':',
  'c','o','m','p','i','l','e','r','_','v','e','r','s','i','o','n','[',
  COMPILER_VERSION_MAJOR,
# ifdef COMPILER_VERSION_MINOR
  '.', COMPILER_VERSION_MINOR,
#  ifdef COMPILER_VERSION_PATCH
   '.', COMPILER_VERSION_PATCH,
#   ifdef COMPILER_VERSION_TWEAK
    '.', COMPILER_VERSION_TWEAK,
#   endif
#  endif
# endif
  ']','\0'};
#endif

/* Construct a string literal encoding the internal version number. */
#ifdef COMPILER_VERSION_INTERNAL
char const info_version_internal[] = {
  'I', 'N', 'F', 'O', ':',
  'c','o','m','p','i','l','e','r','_','v','e','r','s','i','o','n','_',
  'i','n','t','e','r','n','a','l','[',
  COMPILER_VERSION_INTERNAL,']','\0'};
#elif defined(COMPILER_VERSION_INTERNAL_STR)
char const* info_version_internal = "INFO" ":" "compiler_version_internal["
COMPILER_VERSION_INTERNAL_STR "]";
#endif

/* Construct a string literal encoding the version number components. */
#ifdef SIMULATE_VERSION_MAJOR
char const info_simulate_version[] = {
  'I', 'N', 'F', 'O', ':',
  's','i','m','u','l','a','t','e','_','v','e','r','s','i','o','n','[',
  SIMULATE_VERSION_MAJOR,
# ifdef SIMULATE_VERSION_MINOR
  '.', SIMULATE_VERSION_MINOR,
#  ifdef SIMULATE_VERSION_PATCH
   '.', SIMULATE_VERSION_PATCH,
#   ifdef SIMULATE_VERSION_TWEAK
    '.', SIMULATE_VERSION_TWEAK,
#   endif
#  endif
# endif
  ']','\0'};
#endif

/* Construct the string literal in pieces to prevent the source from
   getting matched.  Store it in a pointer rather than an array
   because some compilers will just produce instructions to fill the
   array rather than assigning a pointer to a static array.  */
char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]";
char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]";



#if defined(__INTEL_COMPILER) && defined(_MSVC_LANG) && _MSVC_LANG < 201403L
#  if defined(__INTEL_CXX11_MODE__)
#    if defined(__cpp_aggregate_nsdmi)
#      define CXX_STD 201402L
#    else
#      define CXX_STD 201103L
#    endif
#  else
#    define CXX_STD 199711L
#  endif
#elif defined(_MSC_VER) && defined(_MSVC_LANG)
#  define CXX_STD _MSVC_LANG
```

```
#else
#  define CXX_STD __cplusplus
#endif

const char* info_language_standard_default = "INFO" ":" "standard_default["
#if CXX_STD > 202002L
  "23"
#elif CXX_STD > 201703L
  "20"
#elif CXX_STD >= 201703L
  "17"
#elif CXX_STD >= 201402L
  "14"
#elif CXX_STD >= 201103L
  "11"
#else
  "98"
#endif
"]";

const char* info_language_extensions_default = "INFO" ":" "extensions_default["
#if (defined(__clang__) || defined(__GNUC__) || defined(__xlC__) ||           \
     defined(__TI_COMPILER_VERSION__)) &&                                     \
  !defined(__STRICT_ANSI__)
  "ON"
#else
  "OFF"
#endif
"]";

/*--------------------------------------------------------------------------*/

int main(int argc, char* argv[])
{
  int require = 0;
  require += info_compiler[argc];
  require += info_platform[argc];
  require += info_arch[argc];
#ifdef COMPILER_VERSION_MAJOR
  require += info_version[argc];
#endif
#ifdef COMPILER_VERSION_INTERNAL
  require += info_version_internal[argc];
#endif
#ifdef SIMULATE_ID
  require += info_simulate[argc];
#endif
#ifdef SIMULATE_VERSION_MAJOR
  require += info_simulate_version[argc];
#endif
#if defined(__CRAYXT_COMPUTE_LINUX_TARGET)
  require += info_cray[argc];
#endif
  require += info_language_standard_default[argc];
  require += info_language_extensions_default[argc];
  (void)argv;
  return require;
}
```

5.5 class_funkcijos.cpp File Reference

```
#include "class_studentai.h"
#include "class_funkcijos.h"
```

Include dependency graph for class_funkcijos.cpp:

5.6 class_funkcijos.cpp

Go to the documentation of this file.

```
#include "class_studentai.h"
#include "class_funkcijos.h"


void Netinkamas_Ivestis(std::string Problema)
{
```

```cpp
    std::cin.clear();
    std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
    std::cerr « Problema;
}


std::random_device rd;
std::mt19937 generuoti(rd());
std::uniform_int_distribution<int> nd_kiekis(5, 20);
std::uniform_int_distribution<int> dis(1, 10);
std::uniform_int_distribution<int> dis_lytis(0, 1);

std::vector<std::string> vardaiV = { "Jonas", "Petras", "Antanas", "Juozas", "Kazys", "Darius", "Linas", "Tomas", "Giedrius", "Marius" };
std::vector<std::string> vardaiM = { "Ona", "Maryte", "Aldona", "Gabija", "Dalia", "Danute", "Asta", "Rasa", "Nijole", "Aiste", "Gabriele" };
std::vector<std::string> pavardesV = { "Jonaitis", "Petraitis", "Antanaitis", "Juozaitis", "Kaziukaitis", "Dariukaitis", "Linaitis", "Tomaitis", "Giedraitis", "Mariukaitis" };
std::vector<std::string> pavardesM = { "Jonaite", "Petraityte", "Antanaite", "Juozaite", "Kaziukaite", "Dariukaite", "Linaite", "Tomaite", "Giedraite", "Mariukaite", "Antaniene", "Jonaitiene", "Antaniene" };

int lytis = dis_lytis(generuoti);

void GeneruotiNDPazymius(studentas& S, int ND_kiekis)
{
    std::vector<int> pazymiai;
    for (int i = 0; i < ND_kiekis; ++i) {
        pazymiai.push_back(dis(generuoti));
    }
    S.setND(pazymiai);
    S.setEGZ(dis(generuoti));
}

void GeneruotiVardus(studentas& S)
{
    int lytis = dis_lytis(generuoti);

    if (lytis == 0)
    {
        S.setVardas(vardaiV[dis(generuoti) % 10]);
        S.setPavarde(pavardesV[dis(generuoti) % 10]);
    }

    else
    {
        S.setVardas(vardaiM[dis(generuoti) % 10]);
        S.setPavarde(pavardesM[dis(generuoti) % 10]);
    }

}

void GeneruotiFailus(int reserveDydis, std::string& G_Failo_Vieta)
{
    int nd_kiekis_gen = nd_kiekis(generuoti);

    std::ofstream GFailas(G_Failo_Vieta);

    if (!GFailas.is_open())
    {
        std::cout « "Nepavyko atidaryti failo " « G_Failo_Vieta « std::endl;
        return;
    }

    //headline spausdinimas
    GFailas « std::left « std::setw(20) « "Pavarde" « std::setw(20) « "Vardas";

    for (int i = 0; i < nd_kiekis_gen; i++)
    {
        GFailas « std::left « std::setw(7) « "ND" + std::to_string(i + 1);
    }

    GFailas « std::setw(5) « "Egz." « std::endl;

    //studentu duomenu spasudiniams
    for (int i = 0; i < reserveDydis; i++)
    {
        GFailas « std::left « std::setw(20) « "Pavarde" + std::to_string(i + 1) « std::left « std::setw(20) « "Vardas" + std::to_string(i + 1);
        for (int j = 0; j < nd_kiekis_gen; j++)
        {
            GFailas « std::setw(7) « dis(generuoti);
        }
        GFailas « std::setw(7) « dis(generuoti);
        GFailas « "\n";
    }

    GFailas.close();

}


void Ivesti_Pazymius(studentas& S)
{
    std::vector<int> pazymiai;
    char TaipNePaz;
    std::cout « "\nIveskite namu darbu pazymi: ";
    int pazymys;
    do
    {
        while (true)
        {
            try
            {
                std::cin » pazymys;
                if (std::cin.fail() || std::cin.peek() != '\n' || pazymys < 1 || pazymys > 10)
                {
                    throw std::invalid_argument("Netinkama ivestis. Iveskite sveikaji skaiciu nuo 1 iki 10. ");
                }

                if (pazymys >= 1 && pazymys <= 10)
                {
                    pazymiai.push_back(pazymys);// pridedamas pazymis i vektoriu
                }
                break;
            }

            catch (const std::invalid_argument& paz)
            {
                Netinkamas_Ivestis(paz.what());
            }
        }

        std::cout « "Ar norite ivesti dar viena pazymi? (iveskite T, jei taip , N, jei ne): ";
        while (true)
        {
            try
            {
                std::cin » TaipNePaz;
                if ((std::cin.fail() || std::cin.peek() != '\n') || (TaipNePaz != 'T' && TaipNePaz != 'N'))
                {
                    throw std::invalid_argument("Netinkama ivestis. Iveskite T arba N.  ");
                }

                break;
            }
            catch (const std::invalid_argument& tpp)
            {
                Netinkamas_Ivestis(tpp.what());
            }
        }

        if (TaipNePaz == 'T')
            std::cout « std::endl « "Iveskite namu darbu pazymi: ";

    } while (TaipNePaz == 'T');

    S.setND(pazymiai);

    int egz;
    std::cout « std::endl « "Iveskite egzamino pazymi: ";
    while (true)
    {
        try
        {
            std::cin » egz;
            if (std::cin.fail() || std::cin.peek() != '\n' || egz < 1 || egz > 10)
            {
                throw std::invalid_argument("Netinkama ivestis. Iveskite sveikaji skaiciu nuo 1 iki 10. ");
            }
            break;
        }
        catch (const std::invalid_argument& e)
        {
            Netinkamas_Ivestis(e.what());
        }
    }
    S.setEGZ(egz);
}

void Ivesti_Varda(studentas& S)
```

```
{
    std::string vardas, pavarde;
    std::cout « std::endl « "Iveskite studento varda: ";
    while (true)
    {
        try
        {
            std::cin » vardas;
            if (std::cin.fail() || std::cin.peek() != '\n' || !all_of(vardas.begin(), vardas.end(), ::isalpha))
            {
                throw std::invalid_argument("Netinkama ivestis. Iveskite varda, sudaryta tik is raidziu. ");
            }

            break;
        }
        catch (const std::invalid_argument& v)
        {
            Netinkamas_Ivestis(v.what());
        }
    }

    std::cout « std::endl « "Iveskite studento pavarde: ";
    while (true)
    {
        try
        {
            std::cin » pavarde;
            if (std::cin.fail() || std::cin.peek() != '\n' || !all_of(pavarde.begin(), pavarde.end(), ::isalpha))
            {
                throw std::invalid_argument("Netinkama ivestis. Iveskite pavarde, sudaryta tik is raidziu. ");
            }

            break;
        }
        catch (const std::invalid_argument& pv)
        {
            Netinkamas_Ivestis(pv.what());
        }
    }
    S.setVardas(vardas);
    S.setPavarde(pavarde);
}


std::vector<studentas> Nuskaityti_Is_Failo(const std::string& Failo_Pavadinimas, int reserveDydis)
{
    // Pradedamas skaiciuti laikas
    auto start = std::chrono::high_resolution_clock::now();

    std::ifstream file(Failo_Pavadinimas);
    std::vector<studentas> studentai;
    studentai.reserve(reserveDydis);  // Ið anksto rezervuojama atmintis

    if (!file.is_open())
    {
        std::cerr « "Klaida atidarant faila " « Failo_Pavadinimas « std::endl;
        return studentai;
    }
    // Praleidziama pirma header eilute
    std::string header;
    std::getline(file, header);

    std::string eilute;
    while (std::getline(file, eilute))
    {
        std::istringstream iss(eilute);
        studentas s;
        if (iss » s)
        {
            studentai.push_back(s);
        }
        else
        {
            std::cerr « "Klaida nuskaitant duomenis is eilutes: " « eilute « std::endl;
        }
    }

    // Baigia skaiciuoti laika
    auto end = std::chrono::high_resolution_clock::now();

    //Apskaiciuoja laika
    auto duration = std::chrono::duration_cast<std::chrono::duration<double»(end - start);

    file.close();
    std::cout « "\nFailo nuskaitymas uztruko " « duration.count() « " sek." « std::endl;
    return studentai;
}


void Rikiuoti_Duomenis(std::vector<studentas>& S)
{

    // Rusiavimo pasirinkimai
    std::cout « std::endl « "Rikiuoti pagal:\n 1. Varda\n 2. Pavarde\n 3. Galutini bala, apskaiciuota su mediana\n 4. Galutini bala, apskaiciuota su vidurkiu\n Iveskite pasirinkimo numeri: ";
    int Rusiavimo_Pasirinkimas;
    while (true)
    {
        try
        {
            std::cin » Rusiavimo_Pasirinkimas;

            if (std::cin.fail() || std::cin.peek() != '\n' || Rusiavimo_Pasirinkimas < 1 || Rusiavimo_Pasirinkimas > 4)
            {
                throw std::invalid_argument("Netinkama ivestis. Iveskite sveikaji skaiciu nuo 1 iki 4.");
            }
            break;
        }
        catch (const std::invalid_argument& rp)
        {
            Netinkamas_Ivestis(rp.what());


        }

    }
    // Pradedamas skaiciuoti laikas
    auto RikiavimoPradzia = std::chrono::high_resolution_clock::now();
    switch (Rusiavimo_Pasirinkimas)
    {
    case 1:
        std::sort(S.begin(), S.end(), [](const studentas& a, const studentas& b)
            {
                return a.getVardas() < b.getVardas(); // Rûðiuojama pagal vardà
            });

        break;
    case 2:
        std::sort(S.begin(), S.end(), [](const studentas& a, const studentas& b)
            {
                return a.getPavarde() < b.getPavarde(); // Rûðiuojama pagal pavarde
            });
        break;
    case 3:
        std::sort(S.begin(), S.end(), [](const studentas& a, const studentas& b)
            {
                return a.getGalutinisM() < b.getGalutinisM(); // Rûðiuojama pagal GalutiniM
            });
        break;
    case 4:
        std::sort(S.begin(), S.end(), [](const studentas& a, const studentas& b)
            {
                return a.getGalutinisV() < b.getGalutinisV(); // Rûðiuojama pagal GalutiniV
            });
        break;
    }
    // Baigia skaiciuoti laika
    auto RikiavimoPabaiga = std::chrono::high_resolution_clock::now();

    //Apskaiciuoja laika
    auto Rikiavimotrukme = std::chrono::duration_cast<std::chrono::duration<double»(RikiavimoPabaiga - RikiavimoPradzia);

    std::cout « "\nRikiavimas didejancia tvarka pagal pasirinkta kriteriju uztruko " « Rikiavimotrukme.count() « " sek." « std::endl;
}

void Skirstyti_Studentus(std::vector<studentas>& S, std::vector<studentas>& N, std::vector<studentas>& G, int Strategija)
{

    std::cout « "\nAr norite studentus surusiuoti pagal mediana ar vidurki? M jei mediana, V jei vidurki: ";
    char RusiavimoPasirinkimas;
    while (true)
    {
        try
        {
            std::cin » RusiavimoPasirinkimas;
            if (std::cin.fail() || std::cin.peek() != ’\n’ || (RusiavimoPasirinkimas != ’V’ && RusiavimoPasirinkimas != ’M’))
            {
                throw std::invalid_argument("Netinkama ivestis. Iveskite M arba V: ");
            }
            break;
        }
        catch (const std::invalid_argument& rp)
        {
            Netinkamas_Ivestis(rp.what());
        }
    }

    // Pradedamas skaiciuti laikas
    auto RusavimoPradzia = std::chrono::high_resolution_clock::now();


    if (Strategija == 1)
    {

        for (auto& studentas : S)
        {
            if (RusiavimoPasirinkimas == ’V’)
            {
                if (studentas.getGalutinisV() < 5)
                    N.push_back(studentas);
                else
                    G.push_back(studentas);
            }
            else if (RusiavimoPasirinkimas == ’M’)
            {
                if (studentas.getGalutinisM() < 5)
                    N.push_back(studentas);
                else
                    G.push_back(studentas);
            }
        }
    }
    if (Strategija == 2)
    {

        auto i = S.begin();
        while (i != S.end())
        {
            if (RusiavimoPasirinkimas == ’V’)
            {
                if (i->getGalutinisV() < 5)
                {
                    N.push_back(*i);
                    i = S.erase(i);
                    continue;
                }
            }
            else if (RusiavimoPasirinkimas == ’M’)
            {
                if (i->getGalutinisM() < 5)
                {
                    N.push_back(*i);
                    i = S.erase(i);
                    continue;
                }
            }
            ++i;
        }
    }

    if (Strategija == 3)
    {
        auto i = std::remove_if(S.begin(), S.end(), [&](const auto& studentas)
            {
                bool istrinti = false;
                if (RusiavimoPasirinkimas == ’V’)
                {
                    if (studentas.getGalutinisV() < 5)
                    {
                        istrinti = true;
                    }
                }
                else if (RusiavimoPasirinkimas == ’M’)
                {
                    if (studentas.getGalutinisM() < 5)
                    {
                        istrinti = true;
                    }
                }
                if (istrinti)
                {
                    N.push_back(studentas);
                }
                return istrinti;
            });

        S.erase(i, S.end());

    }

    // Baigia skaiciuoti laika
    auto RusaivimoPabaiga = std::chrono::high_resolution_clock::now();

    //Apskaiciuoja laika
    auto Rusiavimotrukme = std::chrono::duration_cast<std::chrono::duration<double»(RusaivimoPabaiga -
RusavimoPradzia);

    std::cout « "\nRusiavimas i galvocius ir nepazangius uztruko " « Rusiavimotrukme.count() « " sek."
« std::endl;
}

void Spausdinti_Rezultatus(const std::vector<studentas>& N, const std::vector<studentas>& G)
{
    std::ofstream Galvociu_failas("Galvociai.txt");
    if (!Galvociu_failas.is_open())
    {
        std::cerr « "Klaida atidarant rezultatu faila" « std::endl;
        return;
    }
    int i = 0;
    for (auto& studentas : G)
    {
        if (i == 0)
            Galvociu_failas « std::setw(7) « "Nr." « std::setw(20) « "Pavarde" « std::setw(20) «
"Vardas" « std::setw(20) « "Galutinis (Vid.)" « std::setw(20) « "Galutinis (Med.)" « std::endl «
std::setfill(’-’) « std::setw(90) « "-" « std::setfill(’ ’) « std::endl;

        Galvociu_failas « std::setw(7) « i + 1 « studentas;
        i++;

    }
    Galvociu_failas.close();

    std::ofstream Nepazangiuju_failas("Nepazangus.txt");
    if (!Nepazangiuju_failas.is_open())
    {
        std::cerr « "Klaida atidarant rezultatu faila" « std::endl;
        return;
    }
    i = 0;
    for (auto& studentas : N)
    {
        if (i == 0)
            Nepazangiuju_failas « std::setw(7) « "Nr." « std::setw(20) « "Pavarde" « std::setw(20) «
"Vardas" « std::setw(20) « "Galutinis (Vid.)" « std::setw(20) « "Galutinis (Med.)" « std::endl «
std::setfill(’-’) « std::setw(90) « "-" « std::setfill(’ ’) « std::endl;

        Nepazangiuju_failas « std::setw(7) « i + 1 « studentas;
        i++;

    }
    Nepazangiuju_failas.close();

    std::cout « std::endl « "Rezultatai atspausdinti" « std::endl;
}

void Testavimas()
{
    // Testuojamas default konstruktorius
    {
        std::cout « "\n1. Testuojamas default konstruktorius\n\n";
        studentas s;
        std::cout « std::endl;
    }

    // Testuojamas parametrizuotas konstruktorius
    {
        std::cout « "\n2.  Testuojamas parametrizuotas konstruktorius\n\n";
        std::string vardas = "Jonas";
        std::string pavarde = "Jonaitis";
        std::vector<int> nd = { 5, 7, 8 };
        int egz = 9;
        studentas s(vardas, pavarde, nd, egz);
        std::cout « std::endl;
    }

    // Testuojamas copy  konstruktorius
    {
        std::cout « "\n3. Testuojamas copy konstruktorius\n\n";
        studentas s1("Petras", "Petraitis", { 10, 9, 8 }, 10);
        studentas s2 = s1;
        std::cout « std::endl;
    }

    // Testuojamas move konstruktorius
    {
        std::cout « "\n4. Testuojamas move konstruktorius\n\n";
        studentas s1("Kazys", "Kazlauskas", { 6, 5, 7 }, 8);
        studentas s2 = std::move(s1);
        std::cout « std::endl;
    }

    // Testuojamas kopijavimo priskyrimo operatorius
    {
        std::cout « "\n5. Testuojamas copy priskyrimo operatorius\n\n";
        studentas s1, s2;
        s2 = s1;
        std::cout « std::endl;
    }

    // Testuojamas move priskyrimo operatorius
    {
        std::cout « "\n6. Testuojamas move priskyrimo operatorius\n\n";
        studentas s3, s2;
        s3 = std::move(s2);
        std::cout « std::endl;
    }

    // Destruktoriaus patikrinimas
    {
        std::cout « "\n7. Destruktoriaus patikrinimas\n\n";
        // Sukuriamas dynamic studentas
        studentas* s1 = new studentas();

        // I ji pridedamepazymius
        s1->setND({ 10, 9, 8 });

        // Istriname
        delete s1;

        // Sukuriam nauja
        studentas s2;

        // Patikrinam ar jame nebeliko s1 pazymiu
        assert(s2.getND().empty());
        std::cout « std::endl;

    }
    // Testuojamas ivesties operatorius
    {
        std::cout « "\n8. Testuojamas ivesties operatorius\n\n";
        std::vector<int> I = { 5, 6, 7, 8 };//toki ND vector turi gauti
        std::istringstream iss("Mindaugas Mindaugaitis 5 6 7 8 9");
        studentas s;
        iss » s;
        assert(s.getVardas() == "Mindaugas");
        assert(s.getPavarde() == "Mindaugaitis");
        assert(s.getND() == I);
        assert(s.getEGZ() == 9);
        std::cout « std::endl;
    }
    // Testuojamas ivesties operatorius
    {
        std::cout « "\n9. Testuojamas ivesties operatorius\n\n";
        std::istringstream iss("Lina Linaityte 4 5 9 9");
        studentas s;
        iss » s;
        std::ostringstream oss;
        oss « s;
        std::string tikimasi = "           Linaityte                Lina                 7.8 7.4\n";
        assert(oss.str() == tikimasi);
        std::cout « std::endl;
    }

    //realizuota abstrakti klasë zmogus, jos objektø kûrimas negalimas (pademonstruota).
    {
        //zmogus z;

    }
}
```

5.7 class_funkcijos.h File Reference

```
#include "class_studentai.h"
```

Include dependency graph for class_funkcijos.h: This graph shows which files directly or indirectly include this file:

Functions

- void Netinkamas_Ivestis (std::string Problema)
- void GeneruotiNDPazymius (studentas &S, int ND_kiekis)
- void GeneruotiVardus (studentas &S)
- void GeneruotiFailus (int reserveDydis, std::string &failoPav)
- void Ivesti_Pazymius (studentas &S)
- void Ivesti_Varda (studentas &S)
- std::vector< studentas > Nuskaityti_Is_Failo (const std::string &Failo_Pavadinimas, int reserveDydis)
- bool VarduRikiavimas (const studentas &a, const studentas &b)
- bool PavardziuRikiavimas (const studentas &a, const studentas &b)
- bool MedianuRikiavimas (const studentas &a, const studentas &b)
- bool VidurkiuRikiavimas (const studentas &a, const studentas &b)
- void Rikiuoti_Duomenis (std::vector< studentas > &S)
- void Skirstyti_Studentus (std::vector< studentas > &S, std::vector< studentas > &N, std::vector< studentas > &G, int Strategija)
- void Spausdinti_Rezultatus (const std::vector< studentas > &N, const std::vector< studentas > &G)
- void Testavimas ()

5.7.1 Function Documentation

5.7.1.1 GeneruotiFailus()

```
void GeneruotiFailus (
             int reserveDydis,
             std::string & failoPav )
```

Definition at line 59 of file class_funkcijos.cpp.

Here is the call graph for this function: Here is the caller graph for this function:

5.7.1.2 GeneruotiNDPazymius()

```
void GeneruotiNDPazymius (
             studentas & S,
             int ND_kiekis )
```

Definition at line 29 of file class_funkcijos.cpp.

Here is the call graph for this function: Here is the caller graph for this function:

5.7.1.3 GeneruotiVardus()

```
void GeneruotiVardus (
            studentas & S )
```

Definition at line 40 of file class_funkcijos.cpp.

Here is the call graph for this function: Here is the caller graph for this function:

5.7.1.4 Ivesti_Pazymius()

```
void Ivesti_Pazymius (
            studentas & S )
```

Definition at line 99 of file class_funkcijos.cpp.

Here is the call graph for this function: Here is the caller graph for this function:

5.7.1.5 Ivesti_Varda()

```
void Ivesti_Varda (
            studentas & S )
```

Definition at line 177 of file class_funkcijos.cpp.

Here is the call graph for this function: Here is the caller graph for this function:

5.7.1.6 MedianuRikiavimas()

```
bool MedianuRikiavimas (
            const studentas & a,
            const studentas & b )
```

5.7.1.7 Netinkamas_Ivestis()

```
void Netinkamas_Ivestis (
            std::string Problema )
```

Definition at line 6 of file class_funkcijos.cpp.

Here is the caller graph for this function:

5.7.1.8 Nuskaityti_Is_Failo()

```
std::vector< studentas > Nuskaityti_Is_Failo (
            const std::string & Failo_Pavadinimas,
            int reserveDydis )
```

Definition at line 223 of file class_funkcijos.cpp.

Here is the caller graph for this function:

5.7.1.9 PavardziuRikiavimas()

```
bool PavardziuRikiavimas (
             const studentas & a,
             const studentas & b )
```

5.7.1.10 Rikiuoti_Duomenis()

```
void Rikiuoti_Duomenis (
             std::vector< studentas > & S )
```

Definition at line 269 of file class_funkcijos.cpp.

Here is the call graph for this function: Here is the caller graph for this function:

5.7.1.11 Skirstyti_Studentus()

```
void Skirstyti_Studentus (
             std::vector< studentas > & S,
             std::vector< studentas > & N,
             std::vector< studentas > & G,
             int Strategija )
```

Definition at line 335 of file class_funkcijos.cpp.

Here is the call graph for this function: Here is the caller graph for this function:

5.7.1.12 Spausdinti_Rezultatus()

```
void Spausdinti_Rezultatus (
             const std::vector< studentas > & N,
             const std::vector< studentas > & G )
```

Definition at line 450 of file class_funkcijos.cpp.

Here is the caller graph for this function:

5.7.1.13 Testavimas()

```
void Testavimas ( )
```

Definition at line 491 of file class_funkcijos.cpp.

Here is the call graph for this function: Here is the caller graph for this function:

5.7.1.14 VarduRikiavimas()

```
bool VarduRikiavimas (
             const studentas & a,
             const studentas & b )
```

5.7.1.15 VidurkiuRikiavimas()

```
bool VidurkiuRikiavimas (
            const studentas & a,
            const studentas & b )
```

5.8 class_funkcijos.h

Go to the documentation of this file.

```
#ifndef CLASS_FUNKCIJOS_H
#define CLASS_FUNKCIJOS_H
#include "class_studentai.h"

void Netinkamas_Ivestis(std::string Problema);

void GeneruotiNDPazymius(studentas& S, int ND_kiekis);
void GeneruotiVardus(studentas& S);
void GeneruotiFailus(int reserveDydis, std::string& failoPav);

void Ivesti_Pazymius(studentas& S);
void Ivesti_Varda(studentas& S);

std::vector<studentas> Nuskaityti_Is_Failo(const std::string& Failo_Pavadinimas, int reserveDydis);

bool VarduRikiavimas(const studentas& a, const studentas& b);
bool PavardziuRikiavimas(const studentas& a, const studentas& b);
bool MedianuRikiavimas(const studentas& a, const studentas& b);
bool VidurkiuRikiavimas(const studentas& a, const studentas& b);
void Rikiuoti_Duomenis(std::vector<studentas>& S);

void Skirstyti_Studentus(std::vector<studentas>& S, std::vector<studentas>& N, std::vector<studentas>& G, int Strategija);

void Spausdinti_Rezultatus(const std::vector<studentas>& N, const std::vector<studentas>& G);

void Testavimas();

#endif
```

5.9 class_main.cpp File Reference

```
#include "class_studentai.h"
#include "class_funkcijos.h"
```

Include dependency graph for class_main.cpp:

Functions

- int main ()

Variables

- char TaipNe

5.9.1 Function Documentation

5.9.1.1 main()

```
int main ( )
```

Definition at line 6 of file class_main.cpp.

Here is the call graph for this function:

5.9.2 Variable Documentation

5.9.2.1 TaipNe

```
char TaipNe
```

Definition at line 4 of file class_main.cpp.

5.10 class_main.cpp

Go to the documentation of this file.

```
00001 #include "class_studentai.h"
00002 #include "class_funkcijos.h"
00003
00004 char TaipNe;
00005 namespace fs = std::filesystem;
00006 int main()
00007 {
00008     int Pasirinkimas, Strategija;
00009     std::vector<studentas> S;
00010     std::vector<studentas> N;//nuskriaustieji
00011     std::vector<studentas> G;//galvociai
00012
00013     std::cout « "Pasirinkite veiksma:\n 1. Suvesti visus studentu duomenis\n 2. Sugeneruoti tik
      studentu pazymius\n 3. Sugeneruoti studentu vardus ir pazymius\n 4. Nuskaityti studentu duomenis nuo
      failo\n 5. Generuoti failus\n 6. Baigti darba\n 7. Testuoti \n Iveskite pasirinkimo numeri: ";
00014
00015     while (true)
00016     {
00017         try
00018         {
00019             std::cin » Pasirinkimas;
00020             if (std::cin.fail() || std::cin.peek() != '\n' || Pasirinkimas < 1 || Pasirinkimas > 7)
00021             {
00022                 throw std::invalid_argument("Netinkama ivestis. Iveskite sveikaji skaiciu nuo 1 iki 7.
      ");
00023             }
00024             break;
00025         }
00026         catch (const std::invalid_argument& p)
00027         {
00028             Netinkamas_Ivestis(p.what());
00029         }
00030     }
00031     if (Pasirinkimas >= 1 && Pasirinkimas <= 5)
00032     {
00033         std::cout « "\nKuria strategija norite naudoti: 1, 2, 3: ";
00034         while (true)
00035         {
00036             try
00037             {
00038                 std::cin » Strategija;
00039                 if (std::cin.fail() || std::cin.peek() != '\n' || Strategija < 1 || Strategija > 3)
00040                 {
00041                     throw std::invalid_argument("Netinkama ivestis. Iveskite skaiciu nuo 1 iki 3: ");
00042                 }
00043                 break;
00044             }
00045             catch (const std::invalid_argument& s)
00046             {
00047                 Netinkamas_Ivestis(s.what());
00048             }
00049         }
00050
00051         std::cout « std::endl « "VECTOR " « Strategija « " STRATEGIJA\n\n";
00052
00053     }
00054
00055     if (Pasirinkimas == 1)
00056     {
00057         studentas naujas;
00058         do
00059         {
00060             Ivesti_Varda(naujas);
00061
00062             Ivesti_Pazymius(naujas);
00063
```

```
            S.push_back(naujas);// pridedamas studentas i vektoriu

            std::cout « std::endl « "Ar norite ivesti " « S.size() + 1 « " studenta ? (T jei taip, N -
      ne) : ";
            while (true)
            {
                try
                {
                    std::cin » TaipNe;
                    if ((std::cin.fail() || std::cin.peek() != '\n') || (TaipNe != 'T' && TaipNe !=
      'N'))
                    {
                        throw std::invalid_argument("Netinkama ivestis. Iveskite T arba N.  ");
                    }
                    break;
                }
                catch (const std::invalid_argument& tp)
                {
                    Netinkamas_Ivestis(tp.what());
                }
            }

        } while (TaipNe == 'T');

        Rikiuoti_Duomenis(S);
        Skirstyti_Studentus(S, N, G, Strategija);

        if (Strategija == 1)
            Spausdinti_Rezultatus(N, G);

        if (Strategija == 2)
            Spausdinti_Rezultatus(N, S);

        if (Strategija == 3)
            Spausdinti_Rezultatus(N, S);
    }

    if (Pasirinkimas == 2)
    {

        studentas naujas;
        do
        {
            Ivesti_Varda(naujas);

            std::cout « std::endl « "Kiek namu darbu pazymiu norite sugeneruoti: ";
            int ND_kiekis;
            while (true)
            {
                try
                {
                    std::cin » ND_kiekis;
                    if (std::cin.fail() || std::cin.peek() != '\n')
                    {
                        throw std::invalid_argument("Netinkama ivestis. Iveskite sveikaji skaiciu. ");
                    }
                    break;
                }
                catch (const std::invalid_argument& ndk)
                {
                    Netinkamas_Ivestis(ndk.what());
                }
            }


            GeneruotiNDPazymius(naujas, ND_kiekis);

            S.push_back(naujas); // pridedamas studentas i vektoriu
            std::cout « std::endl « "Ar norite ivesti " « S.size() + 1 « " studenta ? (T jei taip, N -
      ne) : ";
            while (true)
            {
                try
                {
                    std::cin » TaipNe;
                    if ((std::cin.fail() || std::cin.peek() != '\n') || (TaipNe != 'T' && TaipNe !=
      'N'))
                    {
                        throw std::invalid_argument("Netinkama ivestis. Iveskite T arba N.  ");
                    }
                    break;
                }
                catch (const std::invalid_argument& tp)
                {
                    Netinkamas_Ivestis(tp.what());
                }
            }


        } while (TaipNe == 'T');

        Rikiuoti_Duomenis(S);
        Skirstyti_Studentus(S, N, G, Strategija);

        if (Strategija == 1)
            Spausdinti_Rezultatus(N, G);

        if (Strategija == 2)
            Spausdinti_Rezultatus(N, S);

        if (Strategija == 3)
            Spausdinti_Rezultatus(N, S);
    }

    if (Pasirinkimas == 3)
    {

        studentas naujas;

        std::cout « std::endl « "Kiek studentu norite sugeneruoti: ";
        int Studentu_kiekis;
        while (true)
        {
            try
            {
                std::cin » Studentu_kiekis;
                if (std::cin.fail() || std::cin.peek() != '\n')
                {
                    throw std::invalid_argument("Netinkama ivestis. Iveskite sveikaji skaiciu. ");
                }
                break;
            }
            catch (const std::invalid_argument& sk)
            {
                Netinkamas_Ivestis(sk.what());


            }

        }

        std::cout « std::endl « "Kiek namu darbu pazymiu norite sugeneruoti: ";
        int ND_kiekis;
        while (true)
        {
            try
            {
                std::cin » ND_kiekis;
                if (std::cin.fail() || std::cin.peek() != '\n')
                {
                    throw std::invalid_argument("Netinkama ivestis. Iveskite sveikaji skaiciu. ");
                }
                break;
            }
            catch (const std::invalid_argument& ndk)
            {
                Netinkamas_Ivestis(ndk.what());


            }
        }

        for (int i = 0; i < Studentu_kiekis; ++i)
        {

            GeneruotiVardus(naujas);

            GeneruotiNDPazymius(naujas, ND_kiekis);


            S.push_back(naujas); // pridedamas studentas i vektoriu
        }

        Rikiuoti_Duomenis(S);

        Skirstyti_Studentus(S, N, G, Strategija);

        if (Strategija == 1)
            Spausdinti_Rezultatus(N, G);

        if (Strategija == 2)
            Spausdinti_Rezultatus(N, S);

        if (Strategija == 3)
```

```
            Spausdinti_Rezultatus(N, S);
    }

    if (Pasirinkimas == 4)
    {
        int Failo_Pasirinkimas;
        std::string Failas;

        while (true)
        {
            // Parinkimo meniu ir failo pasirinkimas
            std::cout « "\nPasirinkite, is kurio failo norite nuskaityti duomenis:\n 1. 1 000\n 2. 10 000\n 3. 100 000\n 4. 1 000 000\n 5. 10 000 000 \n Iveskite pasirinkimo numeri: ";

            std::cin » Failo_Pasirinkimas;
            // Tikrinimas ar įvestis yra tinkama
            if (std::cin.fail() || std::cin.peek() != '\n' || Failo_Pasirinkimas < 1 || Failo_Pasirinkimas > 5)
            {
                throw std::invalid_argument("Netinkama ivestis. Iveskite sveikaji skaiciu nuo 1 iki 5.");
            }
            int reserveDydis = 0;
            // Nuskaitomas pasirinkto failo kelias
            switch (Failo_Pasirinkimas)
            {
            case 1:
                Failas = "Studentai1000.txt";
                reserveDydis = 1000;
                break;
            case 2:
                Failas = "Studentai10000.txt";
                reserveDydis = 10000;
                break;
            case 3:
                Failas = "Studentai100000.txt";
                reserveDydis = 100000;
                break;
            case 4:
                Failas = "Studentai1000000.txt";
                reserveDydis = 1000000;
                break;

            case 5:
                Failas = "Studentai10000000.txt";
                reserveDydis = 10000000;
                break;

            }
            // Tikrinimas ar pasirinktas failas egzistuoja
            if (!fs::exists(Failas))
            {
                Netinkamas_Ivestis("Pasirinktas failas neegzistuoja. Pasirinkite kita faila. ");
                std::cout « "\n";
                continue;

            }
            // Nuskaitymas duomenų iš pasirinkto failo
            S = Nuskaityti_Is_Failo(Failas, reserveDydis);
            Rikiuoti_Duomenis(S);
            Skirstyti_Studentus(S, N, G, Strategija);

            if (Strategija == 1)
                Spausdinti_Rezultatus(N, G);

            if (Strategija == 2)
                Spausdinti_Rezultatus(N, S);

            if (Strategija == 3)
                Spausdinti_Rezultatus(N, S);
            break; // Išeiti iš ciklo, kai buna pasirinktas tinkamas failas
        }

    }

    if (Pasirinkimas == 5)
    {
        std::cout « "Pasirinkite kiek studentu norite sugeneruoti:\n 1. 1 000\n 2. 10 000\n 3. 100 000\n 4. 1 000 000\n 5. 10 000 000 \n Iveskite pasirinkimo numeri: ";

        std::string G_Failas;
        int G_Failo_Pasirinkimas;
        int reserveDydis = 0;
        while (true)
        {
            try
            {
                std::cin » G_Failo_Pasirinkimas;
                if (std::cin.fail() || std::cin.peek() != '\n' || G_Failo_Pasirinkimas < 1 ||
G_Failo_Pasirinkimas > 5)
                {
                    throw std::invalid_argument("Netinkama ivestis. Iveskite sveikaji skaiciu nuo 1
iki 5. ");
                }
                break;
            }
            catch (const std::invalid_argument& pfg)
            {
                Netinkamas_Ivestis(pfg.what());
            }
        }


        switch (G_Failo_Pasirinkimas)
        {
        case 1:
            G_Failas = "Studentai1000.txt";
            reserveDydis = 1000;
            break;
        case 2:
            G_Failas = "Studentai10000.txt";
            reserveDydis = 10000;
            break;
        case 3:
            G_Failas = "Studentai100000.txt";
            reserveDydis = 100000;
            break;
        case 4:
            G_Failas = "Studentai1000000.txt";
            reserveDydis = 1000000;
            break;

        case 5:
            G_Failas = "Studentai10000000.txt";
            reserveDydis = 10000000;
            break;

        }

        //jei failas jau egzistuoja, tiesiog nuskaitoma nuo jo
        if (fs::exists(G_Failas))
        {
            std::cout « "Pasirinktas failas jau egzistuoja, dabar nuo jo bus nuskaitoma";
            S = Nuskaityti_Is_Failo(G_Failas, reserveDydis);
        }
        else //jei failas neegzistuoja, tai ji sugeneruoja ir tada nuskaito
        {
            std::cout « "Pasirinktas failas neegzistuoja, jis generuojamas";
            GeneruotiFailus(reserveDydis, G_Failas);
            S = Nuskaityti_Is_Failo(G_Failas, reserveDydis);
        }
        Rikiuoti_Duomenis(S);
        Skirstyti_Studentus(S, N, G, Strategija);

        if (Strategija == 1)
            Spausdinti_Rezultatus(N, G);

        if (Strategija == 2)
            Spausdinti_Rezultatus(N, S);

        if (Strategija == 3)
            Spausdinti_Rezultatus(N, S);
    }

    if (Pasirinkimas == 6)
        std::cout « "\n" « "Darbas baigtas";

    if (Pasirinkimas == 7)
        Testavimas();



    std::cout « "\n";
    return 0;
}
```

5.11 class_studentai.h File Reference

```
#include <chrono>
#include <string>
#include <vector>
#include <iostream>
#include <sstream>
#include <fstream>
#include <stdexcept>
#include <limits>
#include <filesystem>
#include <algorithm>
#include <numeric>
#include <random>
#include <cassert>
```

Include dependency graph for class_studentai.h: This graph shows which files directly or indirectly include this file:

Classes

- class zmogus
- class studentas

5.12 class_studentai.h

Go to the documentation of this file.

```
#ifndef CLASS_STUDENTAI_H
#define CLASS_STUDENTAI_H

#include <chrono>
#include <string>
#include <vector>
#include <iostream>
#include <sstream>
#include <fstream>
#include <stdexcept>
#include <limits>
#include <filesystem>
#include <algorithm>
#include <numeric>
#include <random>
#include <cassert>

//ZMOGUS
class zmogus {
protected:
    std::string vardas;
    std::string pavarde;
public:
    //Konstruktorius
    zmogus() : vardas("Bevardis"), pavarde("Bepavardis") { /* std::cout « "Suveike zmogus default
konstruktorius\n"; */ }
    zmogus(const std::string& vardas, const std::string& pavarde)
        : vardas(vardas), pavarde(pavarde) {}

    ~zmogus() {}

    virtual std::string getVardas() const = 0;
    virtual std::string getPavarde() const = 0;

    // Setter'iai
    virtual void setVardas(const std::string& newName) { vardas = newName; }
    virtual void setPavarde(const std::string& newSurname) { pavarde = newSurname; }

};

//STUDENTAS
class  studentas : public zmogus {
private:

    std::vector<int> ND;
    int EGZ;
    double GalutinisV;
    double GalutinisM;
    void ApskaiciuotiGalutinius()
    {
        if (!ND.empty())
        {
            GalutinisV = 0.4 * std::accumulate(ND.begin(), ND.end(), 0.0) / ND.size() + 0.6 * EGZ;
            if (ND.size() > 1)
            {
                std::vector<int> sortedND = ND;
                std::sort(sortedND.begin(), sortedND.end());
                size_t mid = sortedND.size() / 2;
                GalutinisM = 0.4 * (sortedND.size() % 2 == 0 ? (sortedND[mid - 1] + sortedND[mid]) /
2.0 : sortedND[mid]) + 0.6 * EGZ;
            }
            else
                GalutinisM = 0.4 * ND[0] + 0.6 * EGZ;
        }
        else
        {
            // Jei ND tuščias
            GalutinisV = GalutinisM = 0.6 * EGZ;
        }
    }

public:
    studentas() : EGZ(0), ND(), GalutinisV(0), GalutinisM(0) {
        //std::cout « "Suveike studentas default konstruktorius\n";
    }

    studentas(const std::string& vardas, const std::string& pavarde, const std::vector<int>& ND, int
EGZ)
        : zmogus(vardas, pavarde), ND(ND), EGZ(EGZ) {
        ApskaiciuotiGalutinius();
        //std::cout « "Suveike parametrizuotas konstruktorius\n";
    }


    // Implementuojame abstrakčius metodus
    virtual std::string getVardas() const override {
        return vardas;
    }

    virtual std::string getPavarde() const override {
        return pavarde;
    }
    // Destruktorius
    ~studentas() { ND.clear();  /*std::cout « "Suveike destruktorius\n";*/ }

    // Copy konstruktorius
    studentas(const studentas& other)
    {
        vardas = other.vardas;
        pavarde = other.pavarde;
        ND = other.ND;
        EGZ = other.EGZ;
        GalutinisV = other.GalutinisV;
        GalutinisM = other.GalutinisM;
        //std::cout « "Suveike copy konstruktorius\n";
    }
    // Move konstruktorius
    studentas(studentas&& other) noexcept
    {
        vardas = std::move(other.vardas);
        pavarde = std::move(other.pavarde);
        ND = std::move(other.ND);
        EGZ = std::move(other.EGZ);
        GalutinisV = std::move(other.GalutinisV);
        GalutinisM = std::move(other.GalutinisM);
        other.clearEverything();
        //std::cout « "Suveike move konstruktorius\n";
    }
    // Copy priskyrimo operatorius
    studentas& operator=(const studentas& other)
    {

        if (this != &other)
        {
            vardas = other.vardas;
            pavarde = other.pavarde;
            ND = other.ND;
            EGZ = other.EGZ;
            GalutinisV = other.GalutinisV;
            GalutinisM = other.GalutinisM;
            //std::cout « "Suveike copy priskyrimo operatorius\n";
        }
        return *this;
    }
    // Move priskyrimo operatorius
    studentas& operator=(studentas&& other) noexcept
    {

        if (this != &other)
        {
            vardas = std::move(other.vardas);
            pavarde = std::move(other.pavarde);
            ND = std::move(other.ND);
            EGZ = std::move(other.EGZ);
            GalutinisV = std::move(other.GalutinisV);
            GalutinisM = std::move(other.GalutinisM);
            other.clearEverything();
            //std::cout « "Suveike move priskyrimo operatorius\n";
        }
        return *this;
    }


    // Getter'iai
    std::vector<int> getND() const { return ND; }
    int getEGZ() const { return EGZ; }
    double getGalutinisV() const { return GalutinisV; }
    double getGalutinisM() const { return GalutinisM; }

    // Setter'iai
    void setVardas(const std::string& newName) { vardas = newName; }
    void setPavarde(const std::string& newSurname) { pavarde = newSurname; }
    void setND(const std::vector<int>& newND) { ND = newND; ApskaiciuotiGalutinius(); }
    void setEGZ(int newEGZ) {
        EGZ = newEGZ;
        ApskaiciuotiGalutinius();
    }


    friend std::istream& operator»(std::istream& is, studentas& s)
    {
        s.vardas.clear();
        s.pavarde.clear();
        s.ND.clear();
        s.EGZ = 0;


        if (!(is » s.vardas » s.pavarde))
        {
            return is;
        }

        int pazymys;
        std::vector<int> NDpazymiai;
        while (is » pazymys)
        {
            NDpazymiai.push_back(pazymys);
        }

        // Patikrina, ar pasiekė failo pabaigą
        if (is.eof()) {
            is.clear();
        }
        // Jei įvedimo operacija nepavyko
        else if (is.fail()) {

            is.clear();
            std::string unused;
            std::getline(is, unused);
            return is;
        }

        if (!NDpazymiai.empty())
        {
            s.EGZ = NDpazymiai.back();
            NDpazymiai.pop_back();
            s.ND = NDpazymiai;
        }

        s.ApskaiciuotiGalutinius();
        //std::cout « "Suveike ivesties operatorius\n";
        return is;
    }

    friend std::ostream& operator«(std::ostream& os, const studentas& s)
    {
        os « std::setw(20) « s.pavarde « std::setw(20) « s.vardas « std::setw(20) «
```

```
std::setprecision(3) « s.GalutinisV « std::setw(20) « std::setprecision(3) « s.GalutinisM « std::endl;
        //std::cout « "Suveike isvesties operatorius\n";
        return os;

    }


    void clearEverything()
    {
        this->vardas.clear();
        this->pavarde.clear();
        this->ND.clear();
        this->EGZ = 0;
        this->GalutinisV = 0;
        this->GalutinisM = 0;

    }
};

#endif
```

Index