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.

```
00001 #ifdef __cplusplus
00002 # error "A C++ compiler has been selected for C."
00003 #endif
00004
00005 #if defined(__18CXX)
00006 # define ID_VOID_MAIN
00007 #endif
00008 #if defined(__CLASSIC_C__)
00009 /* cv-qualifiers did not exist in K&R C */
00010 # define const
00011 # define volatile
00012 #endif
00013
00014 #if !defined(__has_include)
00015 /* If the compiler does not have __has_include, pretend the answer is
00016    always no.  */
00017 #  define __has_include(x) 0
00018 #endif
00019
00020
00021 /* Version number components: V=Version, R=Revision, P=Patch
00022    Version date components:   YYYY=Year, MM=Month,   DD=Day  */
00023
00024 #if defined(__INTEL_COMPILER) || defined(__ICC)
00025 # define COMPILER_ID "Intel"
00026 # if defined(_MSC_VER)
00027 #  define SIMULATE_ID "MSVC"
00028 # endif
00029 # if defined(__GNUC__)
```

```
00030 #  define SIMULATE_ID "GNU"
00031 # endif
00032   /* __INTEL_COMPILER = VRP prior to 2021, and then VVVV for 2021 and later,
00033      except that a few beta releases use the old format with V=2021.  */
00034 # if __INTEL_COMPILER < 2021 || __INTEL_COMPILER == 202110 || __INTEL_COMPILER == 202111
00035 #  define COMPILER_VERSION_MAJOR DEC(__INTEL_COMPILER/100)
00036 #  define COMPILER_VERSION_MINOR DEC(__INTEL_COMPILER/10 % 10)
00037 #  if defined(__INTEL_COMPILER_UPDATE)
00038 #   define COMPILER_VERSION_PATCH DEC(__INTEL_COMPILER_UPDATE)
00039 #  else
00040 #   define COMPILER_VERSION_PATCH DEC(__INTEL_COMPILER   % 10)
00041 #  endif
00042 # else
00043 #  define COMPILER_VERSION_MAJOR DEC(__INTEL_COMPILER)
00044 #  define COMPILER_VERSION_MINOR DEC(__INTEL_COMPILER_UPDATE)
00045    /* The third version component from --version is an update index,
00046       but no macro is provided for it.  */
00047 #  define COMPILER_VERSION_PATCH DEC(0)
00048 # endif
00049 # if defined(__INTEL_COMPILER_BUILD_DATE)
00050    /* __INTEL_COMPILER_BUILD_DATE = YYYYMMDD */
00051 #  define COMPILER_VERSION_TWEAK DEC(__INTEL_COMPILER_BUILD_DATE)
00052 # endif
00053 # if defined(_MSC_VER)
00054    /* _MSC_VER = VVRR */
00055 #  define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
00056 #  define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
00057 # endif
00058 # if defined(__GNUC__)
00059 #  define SIMULATE_VERSION_MAJOR DEC(__GNUC__)
00060 # elif defined(__GNUG__)
00061 #  define SIMULATE_VERSION_MAJOR DEC(__GNUG__)
00062 # endif
00063 # if defined(__GNUC_MINOR__)
00064 #  define SIMULATE_VERSION_MINOR DEC(__GNUC_MINOR__)
00065 # endif
00066 # if defined(__GNUC_PATCHLEVEL__)
00067 #  define SIMULATE_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
00068 # endif
00069
00070 #elif (defined(__clang__) && defined(__INTEL_CLANG_COMPILER)) || defined(__INTEL_LLVM_COMPILER)
00071 # define COMPILER_ID "IntelLLVM"
00072 #if defined(_MSC_VER)
00073 # define SIMULATE_ID "MSVC"
00074 #endif
00075 #if defined(__GNUC__)
00076 # define SIMULATE_ID "GNU"
00077 #endif
00078 /* __INTEL_LLVM_COMPILER = VVVVRP prior to 2021.2.0, VVVVRRPP for 2021.2.0 and
00079  * later.  Look for 6 digit vs. 8 digit version number to decide encoding.
00080  * VVVV is no smaller than the current year when a version is released.
00081  */
00082 #if __INTEL_LLVM_COMPILER < 1000000L
00083 # define COMPILER_VERSION_MAJOR DEC(__INTEL_LLVM_COMPILER/100)
00084 # define COMPILER_VERSION_MINOR DEC(__INTEL_LLVM_COMPILER/10 % 10)
00085 # define COMPILER_VERSION_PATCH DEC(__INTEL_LLVM_COMPILER    % 10)
00086 #else
00087 # define COMPILER_VERSION_MAJOR DEC(__INTEL_LLVM_COMPILER/10000)
00088 # define COMPILER_VERSION_MINOR DEC(__INTEL_LLVM_COMPILER/100 % 100)
00089 # define COMPILER_VERSION_PATCH DEC(__INTEL_LLVM_COMPILER     % 100)
00090 #endif
00091 #if defined(_MSC_VER)
00092   /* _MSC_VER = VVRR */
00093 # define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
00094 # define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
00095 #endif
00096 #if defined(__GNUC__)
00097 # define SIMULATE_VERSION_MAJOR DEC(__GNUC__)
00098 #elif defined(__GNUG__)
00099 # define SIMULATE_VERSION_MAJOR DEC(__GNUG__)
00100 #endif
00101 #if defined(__GNUC_MINOR__)
00102 # define SIMULATE_VERSION_MINOR DEC(__GNUC_MINOR__)
00103 #endif
00104 #if defined(__GNUC_PATCHLEVEL__)
00105 # define SIMULATE_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
00106 #endif
00107
00108 #elif defined(__PATHCC__)
00109 # define COMPILER_ID "PathScale"
00110 # define COMPILER_VERSION_MAJOR DEC(__PATHCC__)
00111 # define COMPILER_VERSION_MINOR DEC(__PATHCC_MINOR__)
00112 # if defined(__PATHCC_PATCHLEVEL__)
00113 #  define COMPILER_VERSION_PATCH DEC(__PATHCC_PATCHLEVEL__)
00114 # endif
00115
00116 #elif defined(__BORLANDC__) && defined(__CODEGEARC_VERSION__)
```

```
00117 # define COMPILER_ID "Embarcadero"
00118 # define COMPILER_VERSION_MAJOR HEX(__CODEGEARC_VERSION__»24 & 0x00FF)
00119 # define COMPILER_VERSION_MINOR HEX(__CODEGEARC_VERSION__»16 & 0x00FF)
00120 # define COMPILER_VERSION_PATCH DEC(__CODEGEARC_VERSION__     & 0xFFFF)
00121
00122 #elif defined(__BORLANDC__)
00123 # define COMPILER_ID "Borland"
00124   /* __BORLANDC__ = 0xVRR */
00125 # define COMPILER_VERSION_MAJOR HEX(__BORLANDC__»8)
00126 # define COMPILER_VERSION_MINOR HEX(__BORLANDC__ & 0xFF)
00127
00128 #elif defined(__WATCOMC__) && __WATCOMC__ < 1200
00129 # define COMPILER_ID "Watcom"
00130    /* __WATCOMC__ = VVRR */
00131 # define COMPILER_VERSION_MAJOR DEC(__WATCOMC__ / 100)
00132 # define COMPILER_VERSION_MINOR DEC((__WATCOMC__ / 10) % 10)
00133 # if (__WATCOMC__ % 10) > 0
00134 #  define COMPILER_VERSION_PATCH DEC(__WATCOMC__ % 10)
00135 # endif
00136
00137 #elif defined(__WATCOMC__)
00138 # define COMPILER_ID "OpenWatcom"
00139    /* __WATCOMC__ = VVRP + 1100 */
00140 # define COMPILER_VERSION_MAJOR DEC((__WATCOMC__ - 1100) / 100)
00141 # define COMPILER_VERSION_MINOR DEC((__WATCOMC__ / 10) % 10)
00142 # if (__WATCOMC__ % 10) > 0
00143 #  define COMPILER_VERSION_PATCH DEC(__WATCOMC__ % 10)
00144 # endif
00145
00146 #elif defined(__SUNPRO_C)
00147 # define COMPILER_ID "SunPro"
00148 # if __SUNPRO_C >= 0x5100
00149    /* __SUNPRO_C = 0xVRRP */
00150 #  define COMPILER_VERSION_MAJOR HEX(__SUNPRO_C»12)
00151 #  define COMPILER_VERSION_MINOR HEX(__SUNPRO_C»4 & 0xFF)
00152 #  define COMPILER_VERSION_PATCH HEX(__SUNPRO_C    & 0xF)
00153 # else
00154    /* __SUNPRO_CC = 0xVRP */
00155 #  define COMPILER_VERSION_MAJOR HEX(__SUNPRO_C»8)
00156 #  define COMPILER_VERSION_MINOR HEX(__SUNPRO_C»4 & 0xF)
00157 #  define COMPILER_VERSION_PATCH HEX(__SUNPRO_C    & 0xF)
00158 # endif
00159
00160 #elif defined(__HP_cc)
00161 # define COMPILER_ID "HP"
00162   /* __HP_cc = VVRRPP */
00163 # define COMPILER_VERSION_MAJOR DEC(__HP_cc/10000)
00164 # define COMPILER_VERSION_MINOR DEC(__HP_cc/100 % 100)
00165 # define COMPILER_VERSION_PATCH DEC(__HP_cc     % 100)
00166
00167 #elif defined(__DECC)
00168 # define COMPILER_ID "Compaq"
00169   /* __DECC_VER = VVRRTPPPP */
00170 # define COMPILER_VERSION_MAJOR DEC(__DECC_VER/10000000)
00171 # define COMPILER_VERSION_MINOR DEC(__DECC_VER/100000  % 100)
00172 # define COMPILER_VERSION_PATCH DEC(__DECC_VER         % 10000)
00173
00174 #elif defined(__IBMC__) && defined(__COMPILER_VER__)
00175 # define COMPILER_ID "zOS"
00176   /* __IBMC__ = VRP */
00177 # define COMPILER_VERSION_MAJOR DEC(__IBMC__/100)
00178 # define COMPILER_VERSION_MINOR DEC(__IBMC__/10 % 10)
00179 # define COMPILER_VERSION_PATCH DEC(__IBMC__    % 10)
00180
00181 #elif defined(__open_xl__) && defined(__clang__)
00182 # define COMPILER_ID "IBMClang"
00183 # define COMPILER_VERSION_MAJOR DEC(__open_xl_version__)
00184 # define COMPILER_VERSION_MINOR DEC(__open_xl_release__)
00185 # define COMPILER_VERSION_PATCH DEC(__open_xl_modification__)
00186 # define COMPILER_VERSION_TWEAK DEC(__open_xl_ptf_fix_level__)
00187
00188
00189 #elif defined(__ibmxl__) && defined(__clang__)
00190 # define COMPILER_ID "XLClang"
00191 # define COMPILER_VERSION_MAJOR DEC(__ibmxl_version__)
00192 # define COMPILER_VERSION_MINOR DEC(__ibmxl_release__)
00193 # define COMPILER_VERSION_PATCH DEC(__ibmxl_modification__)
00194 # define COMPILER_VERSION_TWEAK DEC(__ibmxl_ptf_fix_level__)
00195
00196
00197 #elif defined(__IBMC__) && !defined(__COMPILER_VER__) && __IBMC__ >= 800
00198 # define COMPILER_ID "XL"
00199   /* __IBMC__ = VRP */
00200 # define COMPILER_VERSION_MAJOR DEC(__IBMC__/100)
00201 # define COMPILER_VERSION_MINOR DEC(__IBMC__/10 % 10)
00202 # define COMPILER_VERSION_PATCH DEC(__IBMC__    % 10)
00203
```

```
00204 #elif defined(__IBMC__) && !defined(__COMPILER_VER__) && __IBMC__ < 800
00205 # define COMPILER_ID "VisualAge"
00206   /* __IBMC__ = VRP */
00207 # define COMPILER_VERSION_MAJOR DEC(__IBMC__/100)
00208 # define COMPILER_VERSION_MINOR DEC(__IBMC__/10 % 10)
00209 # define COMPILER_VERSION_PATCH DEC(__IBMC__    % 10)
00210
00211 #elif defined(__NVCOMPILER)
00212 # define COMPILER_ID "NVHPC"
00213 # define COMPILER_VERSION_MAJOR DEC(__NVCOMPILER_MAJOR__)
00214 # define COMPILER_VERSION_MINOR DEC(__NVCOMPILER_MINOR__)
00215 # if defined(__NVCOMPILER_PATCHLEVEL__)
00216 #  define COMPILER_VERSION_PATCH DEC(__NVCOMPILER_PATCHLEVEL__)
00217 # endif
00218
00219 #elif defined(__PGI)
00220 # define COMPILER_ID "PGI"
00221 # define COMPILER_VERSION_MAJOR DEC(__PGIC__)
00222 # define COMPILER_VERSION_MINOR DEC(__PGIC_MINOR__)
00223 # if defined(__PGIC_PATCHLEVEL__)
00224 #  define COMPILER_VERSION_PATCH DEC(__PGIC_PATCHLEVEL__)
00225 # endif
00226
00227 #elif defined(__clang__) && defined(__cray__)
00228 # define COMPILER_ID "CrayClang"
00229 # define COMPILER_VERSION_MAJOR DEC(__cray_major__)
00230 # define COMPILER_VERSION_MINOR DEC(__cray_minor__)
00231 # define COMPILER_VERSION_PATCH DEC(__cray_patchlevel__)
00232 # define COMPILER_VERSION_INTERNAL_STR __clang_version__
00233
00234
00235 #elif defined(_CRAYC)
00236 # define COMPILER_ID "Cray"
00237 # define COMPILER_VERSION_MAJOR DEC(_RELEASE_MAJOR)
00238 # define COMPILER_VERSION_MINOR DEC(_RELEASE_MINOR)
00239
00240 #elif defined(__TI_COMPILER_VERSION__)
00241 # define COMPILER_ID "TI"
00242   /* __TI_COMPILER_VERSION__ = VVVRRRPPP */
00243 # define COMPILER_VERSION_MAJOR DEC(__TI_COMPILER_VERSION__/1000000)
00244 # define COMPILER_VERSION_MINOR DEC(__TI_COMPILER_VERSION__/1000   % 1000)
00245 # define COMPILER_VERSION_PATCH DEC(__TI_COMPILER_VERSION__        % 1000)
00246
00247 #elif defined(__CLANG_FUJITSU)
00248 # define COMPILER_ID "FujitsuClang"
00249 # define COMPILER_VERSION_MAJOR DEC(__FCC_major__)
00250 # define COMPILER_VERSION_MINOR DEC(__FCC_minor__)
00251 # define COMPILER_VERSION_PATCH DEC(__FCC_patchlevel__)
00252 # define COMPILER_VERSION_INTERNAL_STR __clang_version__
00253
00254
00255 #elif defined(__FUJITSU)
00256 # define COMPILER_ID "Fujitsu"
00257 # if defined(__FCC_version__)
00258 #   define COMPILER_VERSION __FCC_version__
00259 # elif defined(__FCC_major__)
00260 #   define COMPILER_VERSION_MAJOR DEC(__FCC_major__)
00261 #   define COMPILER_VERSION_MINOR DEC(__FCC_minor__)
00262 #   define COMPILER_VERSION_PATCH DEC(__FCC_patchlevel__)
00263 # endif
00264 # if defined(__fcc_version)
00265 #   define COMPILER_VERSION_INTERNAL DEC(__fcc_version)
00266 # elif defined(__FCC_VERSION)
00267 #   define COMPILER_VERSION_INTERNAL DEC(__FCC_VERSION)
00268 # endif
00269
00270
00271 #elif defined(__ghs__)
00272 # define COMPILER_ID "GHS"
00273 /* __GHS_VERSION_NUMBER = VVVVRP */
00274 # ifdef __GHS_VERSION_NUMBER
00275 # define COMPILER_VERSION_MAJOR DEC(__GHS_VERSION_NUMBER / 100)
00276 # define COMPILER_VERSION_MINOR DEC(__GHS_VERSION_NUMBER / 10 % 10)
00277 # define COMPILER_VERSION_PATCH DEC(__GHS_VERSION_NUMBER      % 10)
00278 # endif
00279
00280 #elif defined(__TASKING__)
00281 # define COMPILER_ID "Tasking"
00282   # define COMPILER_VERSION_MAJOR DEC(__VERSION__/1000)
00283   # define COMPILER_VERSION_MINOR DEC(__VERSION__ % 100)
00284 # define COMPILER_VERSION_INTERNAL DEC(__VERSION__)
00285
00286 #elif defined(__ORANGEC__)
00287 # define COMPILER_ID "OrangeC"
00288 # define COMPILER_VERSION_MAJOR DEC(__ORANGEC_MAJOR__)
00289 # define COMPILER_VERSION_MINOR DEC(__ORANGEC_MINOR__)
00290 # define COMPILER_VERSION_PATCH DEC(__ORANGEC_PATCHLEVEL__)
```

```
00291
00292 #elif defined(__TINYC__)
00293 # define COMPILER_ID "TinyCC"
00294
00295 #elif defined(__BCC__)
00296 # define COMPILER_ID "Bruce"
00297
00298 #elif defined(__SCO_VERSION__)
00299 # define COMPILER_ID "SCO"
00300
00301 #elif defined(__ARMCC_VERSION) && !defined(__clang__)
00302 # define COMPILER_ID "ARMCC"
00303 #if __ARMCC_VERSION >= 1000000
00304   /* __ARMCC_VERSION = VRRPPPP */
00305   # define COMPILER_VERSION_MAJOR DEC(__ARMCC_VERSION/1000000)
00306   # define COMPILER_VERSION_MINOR DEC(__ARMCC_VERSION/10000 % 100)
00307   # define COMPILER_VERSION_PATCH DEC(__ARMCC_VERSION     % 10000)
00308 #else
00309   /* __ARMCC_VERSION = VRPPPP */
00310   # define COMPILER_VERSION_MAJOR DEC(__ARMCC_VERSION/100000)
00311   # define COMPILER_VERSION_MINOR DEC(__ARMCC_VERSION/10000 % 10)
00312   # define COMPILER_VERSION_PATCH DEC(__ARMCC_VERSION    % 10000)
00313 #endif
00314
00315
00316 #elif defined(__clang__) && defined(__apple_build_version__)
00317 # define COMPILER_ID "AppleClang"
00318 # if defined(_MSC_VER)
00319 #  define SIMULATE_ID "MSVC"
00320 # endif
00321 # define COMPILER_VERSION_MAJOR DEC(__clang_major__)
00322 # define COMPILER_VERSION_MINOR DEC(__clang_minor__)
00323 # define COMPILER_VERSION_PATCH DEC(__clang_patchlevel__)
00324 # if defined(_MSC_VER)
00325   /* _MSC_VER = VVRR */
00326 #  define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
00327 #  define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
00328 # endif
00329 # define COMPILER_VERSION_TWEAK DEC(__apple_build_version__)
00330
00331 #elif defined(__clang__) && defined(__ARMCOMPILER_VERSION)
00332 # define COMPILER_ID "ARMClang"
00333   # define COMPILER_VERSION_MAJOR DEC(__ARMCOMPILER_VERSION/1000000)
00334   # define COMPILER_VERSION_MINOR DEC(__ARMCOMPILER_VERSION/10000 % 100)
00335   # define COMPILER_VERSION_PATCH DEC(__ARMCOMPILER_VERSION/100   % 100)
00336 # define COMPILER_VERSION_INTERNAL DEC(__ARMCOMPILER_VERSION)
00337
00338 #elif defined(__clang__) && defined(__ti__)
00339 # define COMPILER_ID "TIClang"
00340   # define COMPILER_VERSION_MAJOR DEC(__ti_major__)
00341   # define COMPILER_VERSION_MINOR DEC(__ti_minor__)
00342   # define COMPILER_VERSION_PATCH DEC(__ti_patchlevel__)
00343 # define COMPILER_VERSION_INTERNAL DEC(__ti_version__)
00344
00345 #elif defined(__clang__)
00346 # define COMPILER_ID "Clang"
00347 # if defined(_MSC_VER)
00348 #  define SIMULATE_ID "MSVC"
00349 # endif
00350 # define COMPILER_VERSION_MAJOR DEC(__clang_major__)
00351 # define COMPILER_VERSION_MINOR DEC(__clang_minor__)
00352 # define COMPILER_VERSION_PATCH DEC(__clang_patchlevel__)
00353 # if defined(_MSC_VER)
00354   /* _MSC_VER = VVRR */
00355 #  define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
00356 #  define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
00357 # endif
00358
00359 #elif defined(__LCC__) && (defined(__GNUC__) || defined(__GNUG__) || defined(__MCST__))
00360 # define COMPILER_ID "LCC"
00361 # define COMPILER_VERSION_MAJOR DEC(__LCC__ / 100)
00362 # define COMPILER_VERSION_MINOR DEC(__LCC__ % 100)
00363 # if defined(__LCC_MINOR__)
00364 #  define COMPILER_VERSION_PATCH DEC(__LCC_MINOR__)
00365 # endif
00366 # if defined(__GNUC__) && defined(__GNUC_MINOR__)
00367 #  define SIMULATE_ID "GNU"
00368 #  define SIMULATE_VERSION_MAJOR DEC(__GNUC__)
00369 #  define SIMULATE_VERSION_MINOR DEC(__GNUC_MINOR__)
00370 #  if defined(__GNUC_PATCHLEVEL__)
00371 #   define SIMULATE_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
00372 #  endif
00373 # endif
00374
00375 #elif defined(__GNUC__)
00376 # define COMPILER_ID "GNU"
00377 # define COMPILER_VERSION_MAJOR DEC(__GNUC__)
```

```
00378 # if defined(__GNUC_MINOR__)
00379 #  define COMPILER_VERSION_MINOR DEC(__GNUC_MINOR__)
00380 # endif
00381 # if defined(__GNUC_PATCHLEVEL__)
00382 #  define COMPILER_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
00383 # endif
00384
00385 #elif defined(_MSC_VER)
00386 # define COMPILER_ID "MSVC"
00387   /* _MSC_VER = VVRR */
00388 # define COMPILER_VERSION_MAJOR DEC(_MSC_VER / 100)
00389 # define COMPILER_VERSION_MINOR DEC(_MSC_VER % 100)
00390 # if defined(_MSC_FULL_VER)
00391 #  if _MSC_VER >= 1400
00392     /* _MSC_FULL_VER = VVRRPPPPP */
00393 #   define COMPILER_VERSION_PATCH DEC(_MSC_FULL_VER % 100000)
00394 #  else
00395     /* _MSC_FULL_VER = VVRRPPPP */
00396 #   define COMPILER_VERSION_PATCH DEC(_MSC_FULL_VER % 10000)
00397 #  endif
00398 # endif
00399 # if defined(_MSC_BUILD)
00400 #  define COMPILER_VERSION_TWEAK DEC(_MSC_BUILD)
00401 # endif
00402
00403 #elif defined(_ADI_COMPILER)
00404 # define COMPILER_ID "ADSP"
00405 #if defined(__VERSIONNUM__)
00406   /* __VERSIONNUM__ = 0xVVRRPPTT */
00407 #  define COMPILER_VERSION_MAJOR DEC(__VERSIONNUM__ >> 24 & 0xFF)
00408 #  define COMPILER_VERSION_MINOR DEC(__VERSIONNUM__ >> 16 & 0xFF)
00409 #  define COMPILER_VERSION_PATCH DEC(__VERSIONNUM__ >> 8 & 0xFF)
00410 #  define COMPILER_VERSION_TWEAK DEC(__VERSIONNUM__ & 0xFF)
00411 #endif
00412
00413 #elif defined(__IAR_SYSTEMS_ICC__) || defined(__IAR_SYSTEMS_ICC)
00414 # define COMPILER_ID "IAR"
00415 # if defined(__VER__) && defined(__ICCARM__)
00416 #  define COMPILER_VERSION_MAJOR DEC((__VER__) / 1000000)
00417 #  define COMPILER_VERSION_MINOR DEC(((__VER__) / 1000) % 1000)
00418 #  define COMPILER_VERSION_PATCH DEC((__VER__) % 1000)
00419 #  define COMPILER_VERSION_INTERNAL DEC(__IAR_SYSTEMS_ICC__)
00420 # elif defined(__VER__) && (defined(__ICCAVR__) || defined(__ICCRX__) || defined(__ICCRH850__) ||
      defined(__ICCRL78__) || defined(__ICC430__) || defined(__ICCRISCV__) || defined(__ICCV850__) ||
      defined(__ICC8051__) || defined(__ICCSTM8__))
00421 #  define COMPILER_VERSION_MAJOR DEC((__VER__) / 100)
00422 #  define COMPILER_VERSION_MINOR DEC((__VER__) - (((__VER__) / 100)*100))
00423 #  define COMPILER_VERSION_PATCH DEC(__SUBVERSION__)
00424 #  define COMPILER_VERSION_INTERNAL DEC(__IAR_SYSTEMS_ICC__)
00425 # endif
00426
00427 #elif defined(__SDCC_VERSION_MAJOR) || defined(SDCC)
00428 # define COMPILER_ID "SDCC"
00429 # if defined(__SDCC_VERSION_MAJOR)
00430 #  define COMPILER_VERSION_MAJOR DEC(__SDCC_VERSION_MAJOR)
00431 #  define COMPILER_VERSION_MINOR DEC(__SDCC_VERSION_MINOR)
00432 #  define COMPILER_VERSION_PATCH DEC(__SDCC_VERSION_PATCH)
00433 # else
00434   /* SDCC = VRP */
00435 #  define COMPILER_VERSION_MAJOR DEC(SDCC/100)
00436 #  define COMPILER_VERSION_MINOR DEC(SDCC/10 % 10)
00437 #  define COMPILER_VERSION_PATCH DEC(SDCC    % 10)
00438 # endif
00439
00440
00441 /* These compilers are either not known or too old to define an
00442   identification macro.  Try to identify the platform and guess that
00443   it is the native compiler.  */
00444 #elif defined(__hpux) || defined(__hpua)
00445 # define COMPILER_ID "HP"
00446
00447 #else /* unknown compiler */
00448 # define COMPILER_ID ""
00449 #endif
00450
00451 /* Construct the string literal in pieces to prevent the source from
00452    getting matched.  Store it in a pointer rather than an array
00453    because some compilers will just produce instructions to fill the
00454    array rather than assigning a pointer to a static array.  */
00455 char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]";
00456 #ifdef SIMULATE_ID
00457 char const* info_simulate = "INFO" ":" "simulate[" SIMULATE_ID "]";
00458 #endif
00459
00460 #ifdef __QNXNTO__
00461 char const* qnxnto = "INFO" ":" "qnxnto[]";
00462 #endif
```

```
00463
00464 #if defined(__CRAYXT_COMPUTE_LINUX_TARGET)
00465 char const *info_cray = "INFO" ":" "compiler_wrapper[CrayPrgEnv]";
00466 #endif
00467
00468 #define STRINGIFY_HELPER(X) #X
00469 #define STRINGIFY(X) STRINGIFY_HELPER(X)
00470
00471 /* Identify known platforms by name.  */
00472 #if defined(__linux) || defined(__linux__) || defined(linux)
00473 # define PLATFORM_ID "Linux"
00474
00475 #elif defined(__MSYS__)
00476 # define PLATFORM_ID "MSYS"
00477
00478 #elif defined(__CYGWIN__)
00479 # define PLATFORM_ID "Cygwin"
00480
00481 #elif defined(__MINGW32__)
00482 # define PLATFORM_ID "MinGW"
00483
00484 #elif defined(__APPLE__)
00485 # define PLATFORM_ID "Darwin"
00486
00487 #elif defined(_WIN32) || defined(__WIN32__) || defined(WIN32)
00488 # define PLATFORM_ID "Windows"
00489
00490 #elif defined(__FreeBSD__) || defined(__FreeBSD)
00491 # define PLATFORM_ID "FreeBSD"
00492
00493 #elif defined(__NetBSD__) || defined(__NetBSD)
00494 # define PLATFORM_ID "NetBSD"
00495
00496 #elif defined(__OpenBSD__) || defined(__OPENBSD)
00497 # define PLATFORM_ID "OpenBSD"
00498
00499 #elif defined(__sun) || defined(sun)
00500 # define PLATFORM_ID "SunOS"
00501
00502 #elif defined(_AIX) || defined(__AIX) || defined(__AIX__) || defined(__aix) || defined(__aix__)
00503 # define PLATFORM_ID "AIX"
00504
00505 #elif defined(__hpux) || defined(__hpux__)
00506 # define PLATFORM_ID "HP-UX"
00507
00508 #elif defined(__HAIKU__)
00509 # define PLATFORM_ID "Haiku"
00510
00511 #elif defined(__BeOS) || defined(__BEOS__) || defined(_BEOS)
00512 # define PLATFORM_ID "BeOS"
00513
00514 #elif defined(__QNX__) || defined(__QNXNTO__)
00515 # define PLATFORM_ID "QNX"
00516
00517 #elif defined(__tru64) || defined(_tru64) || defined(__TRU64__)
00518 # define PLATFORM_ID "Tru64"
00519
00520 #elif defined(__riscos) || defined(__riscos__)
00521 # define PLATFORM_ID "RISCos"
00522
00523 #elif defined(__sinix) || defined(__sinix__) || defined(__SINIX__)
00524 # define PLATFORM_ID "SINIX"
00525
00526 #elif defined(__UNIX_SV__)
00527 # define PLATFORM_ID "UNIX_SV"
00528
00529 #elif defined(__bsdos__)
00530 # define PLATFORM_ID "BSDOS"
00531
00532 #elif defined(_MPRAS) || defined(MPRAS)
00533 # define PLATFORM_ID "MP-RAS"
00534
00535 #elif defined(__osf) || defined(__osf__)
00536 # define PLATFORM_ID "OSF1"
00537
00538 #elif defined(_SCO_SV) || defined(SCO_SV) || defined(sco_sv)
00539 # define PLATFORM_ID "SCO_SV"
00540
00541 #elif defined(__ultrix) || defined(__ultrix__) || defined(_ULTRIX)
00542 # define PLATFORM_ID "ULTRIX"
00543
00544 #elif defined(__XENIX__) || defined(_XENIX) || defined(XENIX)
00545 # define PLATFORM_ID "Xenix"
00546
00547 #elif defined(__WATCOMC__)
00548 # if defined(__LINUX__)
00549 #  define PLATFORM_ID "Linux"
```

```
00550
00551 # elif defined(__DOS__)
00552 #  define PLATFORM_ID "DOS"
00553
00554 # elif defined(__OS2__)
00555 #  define PLATFORM_ID "OS2"
00556
00557 # elif defined(__WINDOWS__)
00558 #  define PLATFORM_ID "Windows3x"
00559
00560 # elif defined(__VXWORKS__)
00561 #  define PLATFORM_ID "VxWorks"
00562
00563 # else /* unknown platform */
00564 #  define PLATFORM_ID
00565 # endif
00566
00567 #elif defined(__INTEGRITY)
00568 # if defined(INT_178B)
00569 #  define PLATFORM_ID "Integrity178"
00570
00571 # else /* regular Integrity */
00572 #  define PLATFORM_ID "Integrity"
00573 # endif
00574
00575 # elif defined(_ADI_COMPILER)
00576 #  define PLATFORM_ID "ADSP"
00577
00578 #else /* unknown platform */
00579 # define PLATFORM_ID
00580
00581 #endif
00582
00583 /* For windows compilers MSVC and Intel we can determine
00584    the architecture of the compiler being used.  This is because
00585    the compilers do not have flags that can change the architecture,
00586    but rather depend on which compiler is being used
00587 */
00588 #if defined(_WIN32) && defined(_MSC_VER)
00589 # if defined(_M_IA64)
00590 #  define ARCHITECTURE_ID "IA64"
00591
00592 # elif defined(_M_ARM64EC)
00593 #  define ARCHITECTURE_ID "ARM64EC"
00594
00595 # elif defined(_M_X64) || defined(_M_AMD64)
00596 #  define ARCHITECTURE_ID "x64"
00597
00598 # elif defined(_M_IX86)
00599 #  define ARCHITECTURE_ID "X86"
00600
00601 # elif defined(_M_ARM64)
00602 #  define ARCHITECTURE_ID "ARM64"
00603
00604 # elif defined(_M_ARM)
00605 #  if _M_ARM == 4
00606 #   define ARCHITECTURE_ID "ARMV4I"
00607 #  elif _M_ARM == 5
00608 #   define ARCHITECTURE_ID "ARMV5I"
00609 #  else
00610 #   define ARCHITECTURE_ID "ARMV" STRINGIFY(_M_ARM)
00611 #  endif
00612
00613 # elif defined(_M_MIPS)
00614 #  define ARCHITECTURE_ID "MIPS"
00615
00616 # elif defined(_M_SH)
00617 #  define ARCHITECTURE_ID "SHx"
00618
00619 # else /* unknown architecture */
00620 #  define ARCHITECTURE_ID ""
00621 # endif
00622
00623 #elif defined(__WATCOMC__)
00624 # if defined(_M_I86)
00625 #  define ARCHITECTURE_ID "I86"
00626
00627 # elif defined(_M_IX86)
00628 #  define ARCHITECTURE_ID "X86"
00629
00630 # else /* unknown architecture */
00631 #  define ARCHITECTURE_ID ""
00632 # endif
00633
00634 #elif defined(__IAR_SYSTEMS_ICC__) || defined(__IAR_SYSTEMS_ICC)
00635 # if defined(__ICCARM__)
00636 #  define ARCHITECTURE_ID "ARM"
```

```
00637
00638 # elif defined(__ICCRX__)
00639 #  define ARCHITECTURE_ID "RX"
00640
00641 # elif defined(__ICCRH850__)
00642 #  define ARCHITECTURE_ID "RH850"
00643
00644 # elif defined(__ICCRL78__)
00645 #  define ARCHITECTURE_ID "RL78"
00646
00647 # elif defined(__ICCRISCV__)
00648 #  define ARCHITECTURE_ID "RISCV"
00649
00650 # elif defined(__ICCAVR__)
00651 #  define ARCHITECTURE_ID "AVR"
00652
00653 # elif defined(__ICC430__)
00654 #  define ARCHITECTURE_ID "MSP430"
00655
00656 # elif defined(__ICCV850__)
00657 #  define ARCHITECTURE_ID "V850"
00658
00659 # elif defined(__ICC8051__)
00660 #  define ARCHITECTURE_ID "8051"
00661
00662 # elif defined(__ICCSTM8__)
00663 #  define ARCHITECTURE_ID "STM8"
00664
00665 # else /* unknown architecture */
00666 #  define ARCHITECTURE_ID ""
00667 # endif
00668
00669 #elif defined(__ghs__)
00670 # if defined(__PPC64__)
00671 #  define ARCHITECTURE_ID "PPC64"
00672
00673 # elif defined(__ppc__)
00674 #  define ARCHITECTURE_ID "PPC"
00675
00676 # elif defined(__ARM__)
00677 #  define ARCHITECTURE_ID "ARM"
00678
00679 # elif defined(__x86_64__)
00680 #  define ARCHITECTURE_ID "x64"
00681
00682 # elif defined(__i386__)
00683 #  define ARCHITECTURE_ID "X86"
00684
00685 # else /* unknown architecture */
00686 #  define ARCHITECTURE_ID ""
00687 # endif
00688
00689 #elif defined(__clang__) && defined(__ti__)
00690 # if defined(__ARM_ARCH)
00691 #  define ARCHITECTURE_ID "Arm"
00692
00693 # else /* unknown architecture */
00694 #  define ARCHITECTURE_ID ""
00695 # endif
00696
00697 #elif defined(__TI_COMPILER_VERSION__)
00698 # if defined(__TI_ARM__)
00699 #  define ARCHITECTURE_ID "ARM"
00700
00701 # elif defined(__MSP430__)
00702 #  define ARCHITECTURE_ID "MSP430"
00703
00704 # elif defined(__TMS320C28XX__)
00705 #  define ARCHITECTURE_ID "TMS320C28x"
00706
00707 # elif defined(__TMS320C6X__) || defined(_TMS320C6X)
00708 #  define ARCHITECTURE_ID "TMS320C6x"
00709
00710 # else /* unknown architecture */
00711 #  define ARCHITECTURE_ID ""
00712 # endif
00713
00714 # elif defined(__ADSPSHARC__)
00715 #  define ARCHITECTURE_ID "SHARC"
00716
00717 # elif defined(__ADSPBLACKFIN__)
00718 #  define ARCHITECTURE_ID "Blackfin"
00719
00720 #elif defined(__TASKING__)
00721
00722 # if defined(__CTC__) || defined(__CPTC__)
00723 #  define ARCHITECTURE_ID "TriCore"
```

```
00724
00725 # elif defined(__CMCS__)
00726 #  define ARCHITECTURE_ID "MCS"
00727
00728 # elif defined(__CARM__)
00729 #  define ARCHITECTURE_ID "ARM"
00730
00731 # elif defined(__CARC__)
00732 #  define ARCHITECTURE_ID "ARC"
00733
00734 # elif defined(__C51__)
00735 #  define ARCHITECTURE_ID "8051"
00736
00737 # elif defined(__CPCP__)
00738 #  define ARCHITECTURE_ID "PCP"
00739
00740 # else
00741 #  define ARCHITECTURE_ID ""
00742 # endif
00743
00744 #else
00745 #  define ARCHITECTURE_ID
00746 #endif
00747
00748 /* Convert integer to decimal digit literals.  */
00749 #define DEC(n)                   \
00750   ('0' + (((n) / 10000000)%10)), \
00751   ('0' + (((n) / 1000000)%10)),  \
00752   ('0' + (((n) / 100000)%10)),   \
00753   ('0' + (((n) / 10000)%10)),    \
00754   ('0' + (((n) / 1000)%10)),     \
00755   ('0' + (((n) / 100)%10)),      \
00756   ('0' + (((n) / 10)%10)),       \
00757   ('0' +  ((n) % 10))
00758
00759 /* Convert integer to hex digit literals.  */
00760 #define HEX(n)             \
00761   ('0' + ((n)>>28 & 0xF)), \
00762   ('0' + ((n)>>24 & 0xF)), \
00763   ('0' + ((n)>>20 & 0xF)), \
00764   ('0' + ((n)>>16 & 0xF)), \
00765   ('0' + ((n)>>12 & 0xF)), \
00766   ('0' + ((n)>>8  & 0xF)), \
00767   ('0' + ((n)>>4  & 0xF)), \
00768   ('0' + ((n)     & 0xF))
00769
00770 /* Construct a string literal encoding the version number. */
00771 #ifdef COMPILER_VERSION
00772 char const* info_version = "INFO" ":" "compiler_version[" COMPILER_VERSION "]";
00773
00774 /* Construct a string literal encoding the version number components. */
00775 #elif defined(COMPILER_VERSION_MAJOR)
00776 char const info_version[] = {
00777   'I', 'N', 'F', 'O', ':',
00778   'c','o','m','p','i','l','e','r','_','v','e','r','s','i','o','n','[',
00779   COMPILER_VERSION_MAJOR,
00780 # ifdef COMPILER_VERSION_MINOR
00781   '.', COMPILER_VERSION_MINOR,
00782 #  ifdef COMPILER_VERSION_PATCH
00783    '.', COMPILER_VERSION_PATCH,
00784 #   ifdef COMPILER_VERSION_TWEAK
00785     '.', COMPILER_VERSION_TWEAK,
00786 #   endif
00787 #  endif
00788 # endif
00789   ']','\0'};
00790 #endif
00791
00792 /* Construct a string literal encoding the internal version number. */
00793 #ifdef COMPILER_VERSION_INTERNAL
00794 char const info_version_internal[] = {
00795   'I', 'N', 'F', 'O', ':',
00796   'c','o','m','p','i','l','e','r','_','v','e','r','s','i','o','n','_',
00797   'i','n','t','e','r','n','a','l','[',
00798   COMPILER_VERSION_INTERNAL,']','\0'};
00799 #elif defined(COMPILER_VERSION_INTERNAL_STR)
00800 char const* info_version_internal = "INFO" ":" "compiler_version_internal["
      COMPILER_VERSION_INTERNAL_STR "]";
00801 #endif
00802
00803 /* Construct a string literal encoding the version number components. */
00804 #ifdef SIMULATE_VERSION_MAJOR
00805 char const info_simulate_version[] = {
00806   'I', 'N', 'F', 'O', ':',
00807   's','i','m','u','l','a','t','e','_','v','e','r','s','i','o','n','[',
00808   SIMULATE_VERSION_MAJOR,
00809 # ifdef SIMULATE_VERSION_MINOR
```

```
00810   '.', SIMULATE_VERSION_MINOR,
00811 #  ifdef SIMULATE_VERSION_PATCH
00812    '.', SIMULATE_VERSION_PATCH,
00813 #   ifdef SIMULATE_VERSION_TWEAK
00814     '.', SIMULATE_VERSION_TWEAK,
00815 #   endif
00816 #  endif
00817 # endif
00818   ']','\0'};
00819 #endif
00820
00821 /* Construct the string literal in pieces to prevent the source from
00822    getting matched.  Store it in a pointer rather than an array
00823    because some compilers will just produce instructions to fill the
00824    array rather than assigning a pointer to a static array.  */
00825 char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]";
00826 char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]";
00827
00828
00829
00830 #if !defined(__STDC__) && !defined(__clang__)
00831 # if defined(_MSC_VER) || defined(__ibmxl__) || defined(__IBMC__)
00832 #  define C_VERSION "90"
00833 # else
00834 #  define C_VERSION
00835 # endif
00836 #elif __STDC_VERSION__ > 201710L
00837 # define C_VERSION "23"
00838 #elif __STDC_VERSION__ >= 201710L
00839 # define C_VERSION "17"
00840 #elif __STDC_VERSION__ >= 201000L
00841 # define C_VERSION "11"
00842 #elif __STDC_VERSION__ >= 199901L
00843 # define C_VERSION "99"
00844 #else
00845 # define C_VERSION "90"
00846 #endif
00847 const char* info_language_standard_default =
00848   "INFO" ":" "standard_default[" C_VERSION "]";
00849
00850 const char* info_language_extensions_default = "INFO" ":" "extensions_default["
00851 #if (defined(__clang__) || defined(__GNUC__) || defined(__xlC__) ||           \
00852      defined(__TI_COMPILER_VERSION__)) &&                                     \
00853   !defined(__STRICT_ANSI__)
00854   "ON"
00855 #else
00856   "OFF"
00857 #endif
00858 "]";
00859
00860 /*--------------------------------------------------------------------------*/
00861
00862 #ifdef ID_VOID_MAIN
00863 void main() {}
00864 #else
00865 # if defined(__CLASSIC_C__)
00866 int main(argc, argv) int argc; char *argv[];
00867 # else
00868 int main(int argc, char* argv[])
00869 # endif
00870 {
00871   int require = 0;
00872   require += info_compiler[argc];
00873   require += info_platform[argc];
00874   require += info_arch[argc];
00875 #ifdef COMPILER_VERSION_MAJOR
00876   require += info_version[argc];
00877 #endif
00878 #ifdef COMPILER_VERSION_INTERNAL
00879   require += info_version_internal[argc];
00880 #endif
00881 #ifdef SIMULATE_ID
00882   require += info_simulate[argc];
00883 #endif
00884 #ifdef SIMULATE_VERSION_MAJOR
00885   require += info_simulate_version[argc];
00886 #endif
00887 #if defined(__CRAYXT_COMPUTE_LINUX_TARGET)
00888   require += info_cray[argc];
00889 #endif
00890   require += info_language_standard_default[argc];
00891   require += info_language_extensions_default[argc];
00892   (void)argv;
00893   return require;
00894 }
00895 #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.

```
00001 /* This source file must have a .cpp extension so that all C++ compilers
00002    recognize the extension without flags.  Borland does not know .cxx for
00003    example.  */
00004 #ifndef __cplusplus
00005 # error "A C compiler has been selected for C++."
00006 #endif
00007
00008 #if !defined(__has_include)
00009 /* If the compiler does not have __has_include, pretend the answer is
00010    always no.  */
00011 #  define __has_include(x) 0
00012 #endif
00013
00014
00015 /* Version number components: V=Version, R=Revision, P=Patch
00016    Version date components:   YYYY=Year, MM=Month,   DD=Day  */
00017
00018 #if defined(__INTEL_COMPILER) || defined(__ICC)
00019 # define COMPILER_ID "Intel"
00020 # if defined(_MSC_VER)
00021 #  define SIMULATE_ID "MSVC"
00022 # endif
00023 # if defined(__GNUC__)
00024 #  define SIMULATE_ID "GNU"
00025 # endif
00026   /* __INTEL_COMPILER = VRP prior to 2021, and then VVVV for 2021 and later,
00027      except that a few beta releases use the old format with V=2021.  */
00028 # if __INTEL_COMPILER < 2021 || __INTEL_COMPILER == 202110 || __INTEL_COMPILER == 202111
00029 #  define COMPILER_VERSION_MAJOR DEC(__INTEL_COMPILER/100)
00030 #  define COMPILER_VERSION_MINOR DEC(__INTEL_COMPILER/10 % 10)
00031 #  if defined(__INTEL_COMPILER_UPDATE)
00032 #   define COMPILER_VERSION_PATCH DEC(__INTEL_COMPILER_UPDATE)
00033 #  else
00034 #   define COMPILER_VERSION_PATCH DEC(__INTEL_COMPILER   % 10)
00035 #  endif
00036 # else
00037 #  define COMPILER_VERSION_MAJOR DEC(__INTEL_COMPILER)
00038 #  define COMPILER_VERSION_MINOR DEC(__INTEL_COMPILER_UPDATE)
00039    /* The third version component from --version is an update index,
00040       but no macro is provided for it.  */
```

```
00041 #  define COMPILER_VERSION_PATCH DEC(0)
00042 # endif
00043 # if defined(__INTEL_COMPILER_BUILD_DATE)
00044    /* __INTEL_COMPILER_BUILD_DATE = YYYYMMDD */
00045 #  define COMPILER_VERSION_TWEAK DEC(__INTEL_COMPILER_BUILD_DATE)
00046 # endif
00047 # if defined(_MSC_VER)
00048    /* _MSC_VER = VVRR */
00049 #  define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
00050 #  define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
00051 # endif
00052 # if defined(__GNUC__)
00053 #  define SIMULATE_VERSION_MAJOR DEC(__GNUC__)
00054 # elif defined(__GNUG__)
00055 #  define SIMULATE_VERSION_MAJOR DEC(__GNUG__)
00056 # endif
00057 # if defined(__GNUC_MINOR__)
00058 #  define SIMULATE_VERSION_MINOR DEC(__GNUC_MINOR__)
00059 # endif
00060 # if defined(__GNUC_PATCHLEVEL__)
00061 #  define SIMULATE_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
00062 # endif
00063
00064 #elif (defined(__clang__) && defined(__INTEL_CLANG_COMPILER)) || defined(__INTEL_LLVM_COMPILER)
00065 # define COMPILER_ID "IntelLLVM"
00066 #if defined(_MSC_VER)
00067 # define SIMULATE_ID "MSVC"
00068 #endif
00069 #if defined(__GNUC__)
00070 # define SIMULATE_ID "GNU"
00071 #endif
00072 /* __INTEL_LLVM_COMPILER = VVVVRP prior to 2021.2.0, VVVVRRPP for 2021.2.0 and
00073  * later.  Look for 6 digit vs. 8 digit version number to decide encoding.
00074  * VVVV is no smaller than the current year when a version is released.
00075  */
00076 #if __INTEL_LLVM_COMPILER < 1000000L
00077 # define COMPILER_VERSION_MAJOR DEC(__INTEL_LLVM_COMPILER/100)
00078 # define COMPILER_VERSION_MINOR DEC(__INTEL_LLVM_COMPILER/10 % 10)
00079 # define COMPILER_VERSION_PATCH DEC(__INTEL_LLVM_COMPILER    % 10)
00080 #else
00081 # define COMPILER_VERSION_MAJOR DEC(__INTEL_LLVM_COMPILER/10000)
00082 # define COMPILER_VERSION_MINOR DEC(__INTEL_LLVM_COMPILER/100 % 100)
00083 # define COMPILER_VERSION_PATCH DEC(__INTEL_LLVM_COMPILER     % 100)
00084 #endif
00085 #if defined(_MSC_VER)
00086   /* _MSC_VER = VVRR */
00087 # define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
00088 # define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
00089 #endif
00090 #if defined(__GNUC__)
00091 # define SIMULATE_VERSION_MAJOR DEC(__GNUC__)
00092 #elif defined(__GNUG__)
00093 # define SIMULATE_VERSION_MAJOR DEC(__GNUG__)
00094 #endif
00095 #if defined(__GNUC_MINOR__)
00096 # define SIMULATE_VERSION_MINOR DEC(__GNUC_MINOR__)
00097 #endif
00098 #if defined(__GNUC_PATCHLEVEL__)
00099 # define SIMULATE_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
00100 #endif
00101
00102 #elif defined(__PATHCC__)
00103 # define COMPILER_ID "PathScale"
00104 # define COMPILER_VERSION_MAJOR DEC(__PATHCC__)
00105 # define COMPILER_VERSION_MINOR DEC(__PATHCC_MINOR__)
00106 # if defined(__PATHCC_PATCHLEVEL__)
00107 #  define COMPILER_VERSION_PATCH DEC(__PATHCC_PATCHLEVEL__)
00108 # endif
00109
00110 #elif defined(__BORLANDC__) && defined(__CODEGEARC_VERSION__)
00111 # define COMPILER_ID "Embarcadero"
00112 # define COMPILER_VERSION_MAJOR HEX(__CODEGEARC_VERSION__»24 & 0x00FF)
00113 # define COMPILER_VERSION_MINOR HEX(__CODEGEARC_VERSION__»16 & 0x00FF)
00114 # define COMPILER_VERSION_PATCH DEC(__CODEGEARC_VERSION__     & 0xFFFF)
00115
00116 #elif defined(__BORLANDC__)
00117 # define COMPILER_ID "Borland"
00118   /* __BORLANDC__ = 0xVRR */
00119 # define COMPILER_VERSION_MAJOR HEX(__BORLANDC__»8)
00120 # define COMPILER_VERSION_MINOR HEX(__BORLANDC__ & 0xFF)
00121
00122 #elif defined(__WATCOMC__) && __WATCOMC__ < 1200
00123 # define COMPILER_ID "Watcom"
00124    /* __WATCOMC__ = VVRR */
00125 # define COMPILER_VERSION_MAJOR DEC(__WATCOMC__ / 100)
00126 # define COMPILER_VERSION_MINOR DEC((__WATCOMC__ / 10) % 10)
00127 # if (__WATCOMC__ % 10) > 0
```

```
00128 #  define COMPILER_VERSION_PATCH DEC(__WATCOMC__ % 10)
00129 # endif
00130
00131 #elif defined(__WATCOMC__)
00132 # define COMPILER_ID "OpenWatcom"
00133   /* __WATCOMC__ = VVRP + 1100 */
00134 # define COMPILER_VERSION_MAJOR DEC((__WATCOMC__ - 1100) / 100)
00135 # define COMPILER_VERSION_MINOR DEC((__WATCOMC__ / 10) % 10)
00136 # if (__WATCOMC__ % 10) > 0
00137 #  define COMPILER_VERSION_PATCH DEC(__WATCOMC__ % 10)
00138 # endif
00139
00140 #elif defined(__SUNPRO_CC)
00141 # define COMPILER_ID "SunPro"
00142 # if __SUNPRO_CC >= 0x5100
00143   /* __SUNPRO_CC = 0xVRRP */
00144 #  define COMPILER_VERSION_MAJOR HEX(__SUNPRO_CC»12)
00145 #  define COMPILER_VERSION_MINOR HEX(__SUNPRO_CC»4 & 0xFF)
00146 #  define COMPILER_VERSION_PATCH HEX(__SUNPRO_CC    & 0xF)
00147 # else
00148   /* __SUNPRO_CC = 0xVRP */
00149 #  define COMPILER_VERSION_MAJOR HEX(__SUNPRO_CC»8)
00150 #  define COMPILER_VERSION_MINOR HEX(__SUNPRO_CC»4 & 0xF)
00151 #  define COMPILER_VERSION_PATCH HEX(__SUNPRO_CC    & 0xF)
00152 # endif
00153
00154 #elif defined(__HP_aCC)
00155 # define COMPILER_ID "HP"
00156   /* __HP_aCC = VVRRPP */
00157 # define COMPILER_VERSION_MAJOR DEC(__HP_aCC/10000)
00158 # define COMPILER_VERSION_MINOR DEC(__HP_aCC/100 % 100)
00159 # define COMPILER_VERSION_PATCH DEC(__HP_aCC     % 100)
00160
00161 #elif defined(__DECCXX)
00162 # define COMPILER_ID "Compaq"
00163   /* __DECCXX_VER = VVRRTPPPP */
00164 # define COMPILER_VERSION_MAJOR DEC(__DECCXX_VER/10000000)
00165 # define COMPILER_VERSION_MINOR DEC(__DECCXX_VER/100000  % 100)
00166 # define COMPILER_VERSION_PATCH DEC(__DECCXX_VER         % 10000)
00167
00168 #elif defined(__IBMCPP__) && defined(__COMPILER_VER__)
00169 # define COMPILER_ID "zOS"
00170   /* __IBMCPP__ = VRP */
00171 # define COMPILER_VERSION_MAJOR DEC(__IBMCPP__/100)
00172 # define COMPILER_VERSION_MINOR DEC(__IBMCPP__/10 % 10)
00173 # define COMPILER_VERSION_PATCH DEC(__IBMCPP__    % 10)
00174
00175 #elif defined(__open_xl__) && defined(__clang__)
00176 # define COMPILER_ID "IBMClang"
00177 # define COMPILER_VERSION_MAJOR DEC(__open_xl_version__)
00178 # define COMPILER_VERSION_MINOR DEC(__open_xl_release__)
00179 # define COMPILER_VERSION_PATCH DEC(__open_xl_modification__)
00180 # define COMPILER_VERSION_TWEAK DEC(__open_xl_ptf_fix_level__)
00181
00182
00183 #elif defined(__ibmxl__) && defined(__clang__)
00184 # define COMPILER_ID "XLClang"
00185 # define COMPILER_VERSION_MAJOR DEC(__ibmxl_version__)
00186 # define COMPILER_VERSION_MINOR DEC(__ibmxl_release__)
00187 # define COMPILER_VERSION_PATCH DEC(__ibmxl_modification__)
00188 # define COMPILER_VERSION_TWEAK DEC(__ibmxl_ptf_fix_level__)
00189
00190
00191 #elif defined(__IBMCPP__) && !defined(__COMPILER_VER__) && __IBMCPP__ >= 800
00192 # define COMPILER_ID "XL"
00193   /* __IBMCPP__ = VRP */
00194 # define COMPILER_VERSION_MAJOR DEC(__IBMCPP__/100)
00195 # define COMPILER_VERSION_MINOR DEC(__IBMCPP__/10 % 10)
00196 # define COMPILER_VERSION_PATCH DEC(__IBMCPP__    % 10)
00197
00198 #elif defined(__IBMCPP__) && !defined(__COMPILER_VER__) && __IBMCPP__ < 800
00199 # define COMPILER_ID "VisualAge"
00200   /* __IBMCPP__ = VRP */
00201 # define COMPILER_VERSION_MAJOR DEC(__IBMCPP__/100)
00202 # define COMPILER_VERSION_MINOR DEC(__IBMCPP__/10 % 10)
00203 # define COMPILER_VERSION_PATCH DEC(__IBMCPP__    % 10)
00204
00205 #elif defined(__NVCOMPILER)
00206 # define COMPILER_ID "NVHPC"
00207 # define COMPILER_VERSION_MAJOR DEC(__NVCOMPILER_MAJOR__)
00208 # define COMPILER_VERSION_MINOR DEC(__NVCOMPILER_MINOR__)
00209 # if defined(__NVCOMPILER_PATCHLEVEL__)
00210 #  define COMPILER_VERSION_PATCH DEC(__NVCOMPILER_PATCHLEVEL__)
00211 # endif
00212
00213 #elif defined(__PGI)
00214 # define COMPILER_ID "PGI"
```

```
00215 # define COMPILER_VERSION_MAJOR DEC(__PGIC__)
00216 # define COMPILER_VERSION_MINOR DEC(__PGIC_MINOR__)
00217 # if defined(__PGIC_PATCHLEVEL__)
00218 #  define COMPILER_VERSION_PATCH DEC(__PGIC_PATCHLEVEL__)
00219 # endif
00220
00221 #elif defined(__clang__) && defined(__cray__)
00222 # define COMPILER_ID "CrayClang"
00223 # define COMPILER_VERSION_MAJOR DEC(__cray_major__)
00224 # define COMPILER_VERSION_MINOR DEC(__cray_minor__)
00225 # define COMPILER_VERSION_PATCH DEC(__cray_patchlevel__)
00226 # define COMPILER_VERSION_INTERNAL_STR __clang_version__
00227
00228
00229 #elif defined(_CRAYC)
00230 # define COMPILER_ID "Cray"
00231 # define COMPILER_VERSION_MAJOR DEC(_RELEASE_MAJOR)
00232 # define COMPILER_VERSION_MINOR DEC(_RELEASE_MINOR)
00233
00234 #elif defined(__TI_COMPILER_VERSION__)
00235 # define COMPILER_ID "TI"
00236   /* __TI_COMPILER_VERSION__ = VVVRRRPPP */
00237 # define COMPILER_VERSION_MAJOR DEC(__TI_COMPILER_VERSION__/1000000)
00238 # define COMPILER_VERSION_MINOR DEC(__TI_COMPILER_VERSION__/1000   % 1000)
00239 # define COMPILER_VERSION_PATCH DEC(__TI_COMPILER_VERSION__        % 1000)
00240
00241 #elif defined(__CLANG_FUJITSU)
00242 # define COMPILER_ID "FujitsuClang"
00243 # define COMPILER_VERSION_MAJOR DEC(__FCC_major__)
00244 # define COMPILER_VERSION_MINOR DEC(__FCC_minor__)
00245 # define COMPILER_VERSION_PATCH DEC(__FCC_patchlevel__)
00246 # define COMPILER_VERSION_INTERNAL_STR __clang_version__
00247
00248
00249 #elif defined(__FUJITSU)
00250 # define COMPILER_ID "Fujitsu"
00251 # if defined(__FCC_version__)
00252 #   define COMPILER_VERSION __FCC_version__
00253 # elif defined(__FCC_major__)
00254 #   define COMPILER_VERSION_MAJOR DEC(__FCC_major__)
00255 #   define COMPILER_VERSION_MINOR DEC(__FCC_minor__)
00256 #   define COMPILER_VERSION_PATCH DEC(__FCC_patchlevel__)
00257 # endif
00258 # if defined(__fcc_version)
00259 #   define COMPILER_VERSION_INTERNAL DEC(__fcc_version)
00260 # elif defined(__FCC_VERSION)
00261 #   define COMPILER_VERSION_INTERNAL DEC(__FCC_VERSION)
00262 # endif
00263
00264
00265 #elif defined(__ghs__)
00266 # define COMPILER_ID "GHS"
00267 /* __GHS_VERSION_NUMBER = VVVVRP */
00268 # ifdef __GHS_VERSION_NUMBER
00269 # define COMPILER_VERSION_MAJOR DEC(__GHS_VERSION_NUMBER / 100)
00270 # define COMPILER_VERSION_MINOR DEC(__GHS_VERSION_NUMBER / 10 % 10)
00271 # define COMPILER_VERSION_PATCH DEC(__GHS_VERSION_NUMBER      % 10)
00272 # endif
00273
00274 #elif defined(__TASKING__)
00275 # define COMPILER_ID "Tasking"
00276   # define COMPILER_VERSION_MAJOR DEC(__VERSION__/1000)
00277   # define COMPILER_VERSION_MINOR DEC(__VERSION__ % 100)
00278 # define COMPILER_VERSION_INTERNAL DEC(__VERSION__)
00279
00280 #elif defined(__ORANGEC__)
00281 # define COMPILER_ID "OrangeC"
00282 # define COMPILER_VERSION_MAJOR DEC(__ORANGEC_MAJOR__)
00283 # define COMPILER_VERSION_MINOR DEC(__ORANGEC_MINOR__)
00284 # define COMPILER_VERSION_PATCH DEC(__ORANGEC_PATCHLEVEL__)
00285
00286 #elif defined(__SCO_VERSION__)
00287 # define COMPILER_ID "SCO"
00288
00289 #elif defined(__ARMCC_VERSION) && !defined(__clang__)
00290 # define COMPILER_ID "ARMCC"
00291 #if __ARMCC_VERSION >= 1000000
00292   /* __ARMCC_VERSION = VRRPPPP */
00293   # define COMPILER_VERSION_MAJOR DEC(__ARMCC_VERSION/1000000)
00294   # define COMPILER_VERSION_MINOR DEC(__ARMCC_VERSION/10000 % 100)
00295   # define COMPILER_VERSION_PATCH DEC(__ARMCC_VERSION     % 10000)
00296 #else
00297   /* __ARMCC_VERSION = VRPPPP */
00298   # define COMPILER_VERSION_MAJOR DEC(__ARMCC_VERSION/100000)
00299   # define COMPILER_VERSION_MINOR DEC(__ARMCC_VERSION/10000 % 10)
00300   # define COMPILER_VERSION_PATCH DEC(__ARMCC_VERSION    % 10000)
00301 #endif
```

```
00302
00303
00304 #elif defined(__clang__) && defined(__apple_build_version__)
00305 # define COMPILER_ID "AppleClang"
00306 # if defined(_MSC_VER)
00307 #  define SIMULATE_ID "MSVC"
00308 # endif
00309 # define COMPILER_VERSION_MAJOR DEC(__clang_major__)
00310 # define COMPILER_VERSION_MINOR DEC(__clang_minor__)
00311 # define COMPILER_VERSION_PATCH DEC(__clang_patchlevel__)
00312 # if defined(_MSC_VER)
00313    /* _MSC_VER = VVRR */
00314 #  define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
00315 #  define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
00316 # endif
00317 # define COMPILER_VERSION_TWEAK DEC(__apple_build_version__)
00318
00319 #elif defined(__clang__) && defined(__ARMCOMPILER_VERSION)
00320 # define COMPILER_ID "ARMClang"
00321   # define COMPILER_VERSION_MAJOR DEC(__ARMCOMPILER_VERSION/1000000)
00322   # define COMPILER_VERSION_MINOR DEC(__ARMCOMPILER_VERSION/10000 % 100)
00323   # define COMPILER_VERSION_PATCH DEC(__ARMCOMPILER_VERSION/100   % 100)
00324 # define COMPILER_VERSION_INTERNAL DEC(__ARMCOMPILER_VERSION)
00325
00326 #elif defined(__clang__) && defined(__ti__)
00327 # define COMPILER_ID "TIClang"
00328   # define COMPILER_VERSION_MAJOR DEC(__ti_major__)
00329   # define COMPILER_VERSION_MINOR DEC(__ti_minor__)
00330   # define COMPILER_VERSION_PATCH DEC(__ti_patchlevel__)
00331 # define COMPILER_VERSION_INTERNAL DEC(__ti_version__)
00332
00333 #elif defined(__clang__)
00334 # define COMPILER_ID "Clang"
00335 # if defined(_MSC_VER)
00336 #  define SIMULATE_ID "MSVC"
00337 # endif
00338 # define COMPILER_VERSION_MAJOR DEC(__clang_major__)
00339 # define COMPILER_VERSION_MINOR DEC(__clang_minor__)
00340 # define COMPILER_VERSION_PATCH DEC(__clang_patchlevel__)
00341 # if defined(_MSC_VER)
00342    /* _MSC_VER = VVRR */
00343 #  define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
00344 #  define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
00345 # endif
00346
00347 #elif defined(__LCC__) && (defined(__GNUC__) || defined(__GNUG__) || defined(__MCST__))
00348 # define COMPILER_ID "LCC"
00349 # define COMPILER_VERSION_MAJOR DEC(__LCC__ / 100)
00350 # define COMPILER_VERSION_MINOR DEC(__LCC__ % 100)
00351 # if defined(__LCC_MINOR__)
00352 #  define COMPILER_VERSION_PATCH DEC(__LCC_MINOR__)
00353 # endif
00354 # if defined(__GNUC__) && defined(__GNUC_MINOR__)
00355 #  define SIMULATE_ID "GNU"
00356 #  define SIMULATE_VERSION_MAJOR DEC(__GNUC__)
00357 #  define SIMULATE_VERSION_MINOR DEC(__GNUC_MINOR__)
00358 #  if defined(__GNUC_PATCHLEVEL__)
00359 #   define SIMULATE_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
00360 #  endif
00361 # endif
00362
00363 #elif defined(__GNUC__) || defined(__GNUG__)
00364 # define COMPILER_ID "GNU"
00365 # if defined(__GNUC__)
00366 #  define COMPILER_VERSION_MAJOR DEC(__GNUC__)
00367 # else
00368 #  define COMPILER_VERSION_MAJOR DEC(__GNUG__)
00369 # endif
00370 # if defined(__GNUC_MINOR__)
00371 #  define COMPILER_VERSION_MINOR DEC(__GNUC_MINOR__)
00372 # endif
00373 # if defined(__GNUC_PATCHLEVEL__)
00374 #  define COMPILER_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
00375 # endif
00376
00377 #elif defined(_MSC_VER)
00378 # define COMPILER_ID "MSVC"
00379   /* _MSC_VER = VVRR */
00380 # define COMPILER_VERSION_MAJOR DEC(_MSC_VER / 100)
00381 # define COMPILER_VERSION_MINOR DEC(_MSC_VER % 100)
00382 # if defined(_MSC_FULL_VER)
00383 #  if _MSC_VER >= 1400
00384     /* _MSC_FULL_VER = VVRRPPPPP */
00385 #   define COMPILER_VERSION_PATCH DEC(_MSC_FULL_VER % 100000)
00386 #  else
00387     /* _MSC_FULL_VER = VVRRPPPP */
00388 #   define COMPILER_VERSION_PATCH DEC(_MSC_FULL_VER % 10000)
```

```
00389 #  endif
00390 # endif
00391 # if defined(_MSC_BUILD)
00392 #  define COMPILER_VERSION_TWEAK DEC(_MSC_BUILD)
00393 # endif
00394
00395 #elif defined(_ADI_COMPILER)
00396 # define COMPILER_ID "ADSP"
00397 #if defined(__VERSIONNUM__)
00398   /* __VERSIONNUM__ = 0xVVRRPPTT */
00399 #  define COMPILER_VERSION_MAJOR DEC(__VERSIONNUM__ » 24 & 0xFF)
00400 #  define COMPILER_VERSION_MINOR DEC(__VERSIONNUM__ » 16 & 0xFF)
00401 #  define COMPILER_VERSION_PATCH DEC(__VERSIONNUM__ » 8 & 0xFF)
00402 #  define COMPILER_VERSION_TWEAK DEC(__VERSIONNUM__ & 0xFF)
00403 #endif
00404
00405 #elif defined(__IAR_SYSTEMS_ICC__) || defined(__IAR_SYSTEMS_ICC)
00406 # define COMPILER_ID "IAR"
00407 # if defined(__VER__) && defined(__ICCARM__)
00408 #  define COMPILER_VERSION_MAJOR DEC((__VER__) / 1000000)
00409 #  define COMPILER_VERSION_MINOR DEC(((__VER__) / 1000) % 1000)
00410 #  define COMPILER_VERSION_PATCH DEC((__VER__) % 1000)
00411 #  define COMPILER_VERSION_INTERNAL DEC(__IAR_SYSTEMS_ICC__)
00412 # elif defined(__VER__) && (defined(__ICCAVR__) || defined(__ICCRX__) || defined(__ICCRH850__) ||
      defined(__ICCRL78__) || defined(__ICC430__) || defined(__ICCRISCV__) || defined(__ICCV850__) ||
      defined(__ICC8051__) || defined(__ICCSTM8__))
00413 #  define COMPILER_VERSION_MAJOR DEC((__VER__) / 100)
00414 #  define COMPILER_VERSION_MINOR DEC((__VER__) - (((__VER__) / 100)*100))
00415 #  define COMPILER_VERSION_PATCH DEC(__SUBVERSION__)
00416 #  define COMPILER_VERSION_INTERNAL DEC(__IAR_SYSTEMS_ICC__)
00417 # endif
00418
00419
00420 /* These compilers are either not known or too old to define an
00421   identification macro.  Try to identify the platform and guess that
00422   it is the native compiler.  */
00423 #elif defined(__hpux) || defined(__hpua)
00424 # define COMPILER_ID "HP"
00425
00426 #else /* unknown compiler */
00427 # define COMPILER_ID ""
00428 #endif
00429
00430 /* Construct the string literal in pieces to prevent the source from
00431    getting matched.  Store it in a pointer rather than an array
00432    because some compilers will just produce instructions to fill the
00433    array rather than assigning a pointer to a static array.  */
00434 char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]";
00435 #ifdef SIMULATE_ID
00436 char const* info_simulate = "INFO" ":" "simulate[" SIMULATE_ID "]";
00437 #endif
00438
00439 #ifdef __QNXNTO__
00440 char const* qnxnto = "INFO" ":" "qnxnto[]";
00441 #endif
00442
00443 #if defined(__CRAYXT_COMPUTE_LINUX_TARGET)
00444 char const *info_cray = "INFO" ":" "compiler_wrapper[CrayPrgEnv]";
00445 #endif
00446
00447 #define STRINGIFY_HELPER(X) #X
00448 #define STRINGIFY(X) STRINGIFY_HELPER(X)
00449
00450 /* Identify known platforms by name.  */
00451 #if defined(__linux) || defined(__linux__) || defined(linux)
00452 # define PLATFORM_ID "Linux"
00453
00454 #elif defined(__MSYS__)
00455 # define PLATFORM_ID "MSYS"
00456
00457 #elif defined(__CYGWIN__)
00458 # define PLATFORM_ID "Cygwin"
00459
00460 #elif defined(__MINGW32__)
00461 # define PLATFORM_ID "MinGW"
00462
00463 #elif defined(__APPLE__)
00464 # define PLATFORM_ID "Darwin"
00465
00466 #elif defined(_WIN32) || defined(__WIN32__) || defined(WIN32)
00467 # define PLATFORM_ID "Windows"
00468
00469 #elif defined(__FreeBSD__) || defined(__FreeBSD)
00470 # define PLATFORM_ID "FreeBSD"
00471
00472 #elif defined(__NetBSD__) || defined(__NetBSD)
00473 # define PLATFORM_ID "NetBSD"
```

```
00474
00475 #elif defined(__OpenBSD__) || defined(__OPENBSD)
00476 # define PLATFORM_ID "OpenBSD"
00477
00478 #elif defined(__sun) || defined(sun)
00479 # define PLATFORM_ID "SunOS"
00480
00481 #elif defined(_AIX) || defined(__AIX) || defined(__AIX__) || defined(__aix) || defined(__aix__)
00482 # define PLATFORM_ID "AIX"
00483
00484 #elif defined(__hpux) || defined(__hpux__)
00485 # define PLATFORM_ID "HP-UX"
00486
00487 #elif defined(__HAIKU__)
00488 # define PLATFORM_ID "Haiku"
00489
00490 #elif defined(__BeOS) || defined(__BEOS__) || defined(_BEOS)
00491 # define PLATFORM_ID "BeOS"
00492
00493 #elif defined(__QNX__) || defined(__QNXNTO__)
00494 # define PLATFORM_ID "QNX"
00495
00496 #elif defined(__tru64) || defined(_tru64) || defined(__TRU64__)
00497 # define PLATFORM_ID "Tru64"
00498
00499 #elif defined(__riscos) || defined(__riscos__)
00500 # define PLATFORM_ID "RISCos"
00501
00502 #elif defined(__sinix) || defined(__sinix__) || defined(__SINIX__)
00503 # define PLATFORM_ID "SINIX"
00504
00505 #elif defined(__UNIX_SV__)
00506 # define PLATFORM_ID "UNIX_SV"
00507
00508 #elif defined(__bsdos__)
00509 # define PLATFORM_ID "BSDOS"
00510
00511 #elif defined(_MPRAS) || defined(MPRAS)
00512 # define PLATFORM_ID "MP-RAS"
00513
00514 #elif defined(__osf) || defined(__osf__)
00515 # define PLATFORM_ID "OSF1"
00516
00517 #elif defined(_SCO_SV) || defined(SCO_SV) || defined(sco_sv)
00518 # define PLATFORM_ID "SCO_SV"
00519
00520 #elif defined(__ultrix) || defined(__ultrix__) || defined(_ULTRIX)
00521 # define PLATFORM_ID "ULTRIX"
00522
00523 #elif defined(__XENIX__) || defined(_XENIX) || defined(XENIX)
00524 # define PLATFORM_ID "Xenix"
00525
00526 #elif defined(__WATCOMC__)
00527 # if defined(__LINUX__)
00528 #  define PLATFORM_ID "Linux"
00529
00530 # elif defined(__DOS__)
00531 #  define PLATFORM_ID "DOS"
00532
00533 # elif defined(__OS2__)
00534 #  define PLATFORM_ID "OS2"
00535
00536 # elif defined(__WINDOWS__)
00537 #  define PLATFORM_ID "Windows3x"
00538
00539 # elif defined(__VXWORKS__)
00540 #  define PLATFORM_ID "VxWorks"
00541
00542 # else /* unknown platform */
00543 #  define PLATFORM_ID
00544 # endif
00545
00546 #elif defined(__INTEGRITY)
00547 # if defined(INT_178B)
00548 #  define PLATFORM_ID "Integrity178"
00549
00550 # else /* regular Integrity */
00551 #  define PLATFORM_ID "Integrity"
00552 # endif
00553
00554 # elif defined(_ADI_COMPILER)
00555 #  define PLATFORM_ID "ADSP"
00556
00557 #else /* unknown platform */
00558 # define PLATFORM_ID
00559
00560 #endif
```

```
00561
00562 /* For windows compilers MSVC and Intel we can determine
00563    the architecture of the compiler being used.  This is because
00564    the compilers do not have flags that can change the architecture,
00565    but rather depend on which compiler is being used
00566 */
00567 #if defined(_WIN32) && defined(_MSC_VER)
00568 # if defined(_M_IA64)
00569 #  define ARCHITECTURE_ID "IA64"
00570
00571 # elif defined(_M_ARM64EC)
00572 #  define ARCHITECTURE_ID "ARM64EC"
00573
00574 # elif defined(_M_X64) || defined(_M_AMD64)
00575 #  define ARCHITECTURE_ID "x64"
00576
00577 # elif defined(_M_IX86)
00578 #  define ARCHITECTURE_ID "X86"
00579
00580 # elif defined(_M_ARM64)
00581 #  define ARCHITECTURE_ID "ARM64"
00582
00583 # elif defined(_M_ARM)
00584 #  if _M_ARM == 4
00585 #   define ARCHITECTURE_ID "ARMV4I"
00586 #  elif _M_ARM == 5
00587 #   define ARCHITECTURE_ID "ARMV5I"
00588 #  else
00589 #   define ARCHITECTURE_ID "ARMV" STRINGIFY(_M_ARM)
00590 #  endif
00591
00592 # elif defined(_M_MIPS)
00593 #  define ARCHITECTURE_ID "MIPS"
00594
00595 # elif defined(_M_SH)
00596 #  define ARCHITECTURE_ID "SHx"
00597
00598 # else /* unknown architecture */
00599 #  define ARCHITECTURE_ID ""
00600 # endif
00601
00602 #elif defined(__WATCOMC__)
00603 # if defined(_M_I86)
00604 #  define ARCHITECTURE_ID "I86"
00605
00606 # elif defined(_M_IX86)
00607 #  define ARCHITECTURE_ID "X86"
00608
00609 # else /* unknown architecture */
00610 #  define ARCHITECTURE_ID ""
00611 # endif
00612
00613 #elif defined(__IAR_SYSTEMS_ICC__) || defined(__IAR_SYSTEMS_ICC)
00614 # if defined(__ICCARM__)
00615 #  define ARCHITECTURE_ID "ARM"
00616
00617 # elif defined(__ICCRX__)
00618 #  define ARCHITECTURE_ID "RX"
00619
00620 # elif defined(__ICCRH850__)
00621 #  define ARCHITECTURE_ID "RH850"
00622
00623 # elif defined(__ICCRL78__)
00624 #  define ARCHITECTURE_ID "RL78"
00625
00626 # elif defined(__ICCRISCV__)
00627 #  define ARCHITECTURE_ID "RISCV"
00628
00629 # elif defined(__ICCAVR__)
00630 #  define ARCHITECTURE_ID "AVR"
00631
00632 # elif defined(__ICC430__)
00633 #  define ARCHITECTURE_ID "MSP430"
00634
00635 # elif defined(__ICCV850__)
00636 #  define ARCHITECTURE_ID "V850"
00637
00638 # elif defined(__ICC8051__)
00639 #  define ARCHITECTURE_ID "8051"
00640
00641 # elif defined(__ICCSTM8__)
00642 #  define ARCHITECTURE_ID "STM8"
00643
00644 # else /* unknown architecture */
00645 #  define ARCHITECTURE_ID ""
00646 # endif
00647
```

```
00648 #elif defined(__ghs__)
00649 # if defined(__PPC64__)
00650 #  define ARCHITECTURE_ID "PPC64"
00651
00652 # elif defined(__ppc__)
00653 #  define ARCHITECTURE_ID "PPC"
00654
00655 # elif defined(__ARM__)
00656 #  define ARCHITECTURE_ID "ARM"
00657
00658 # elif defined(__x86_64__)
00659 #  define ARCHITECTURE_ID "x64"
00660
00661 # elif defined(__i386__)
00662 #  define ARCHITECTURE_ID "X86"
00663
00664 # else /* unknown architecture */
00665 #  define ARCHITECTURE_ID ""
00666 # endif
00667
00668 #elif defined(__clang__) && defined(__ti__)
00669 # if defined(__ARM_ARCH)
00670 #  define ARCHITECTURE_ID "Arm"
00671
00672 # else /* unknown architecture */
00673 #  define ARCHITECTURE_ID ""
00674 # endif
00675
00676 #elif defined(__TI_COMPILER_VERSION__)
00677 # if defined(__TI_ARM__)
00678 #  define ARCHITECTURE_ID "ARM"
00679
00680 # elif defined(__MSP430__)
00681 #  define ARCHITECTURE_ID "MSP430"
00682
00683 # elif defined(__TMS320C28XX__)
00684 #  define ARCHITECTURE_ID "TMS320C28x"
00685
00686 # elif defined(__TMS320C6X__) || defined(_TMS320C6X)
00687 #  define ARCHITECTURE_ID "TMS320C6x"
00688
00689 # else /* unknown architecture */
00690 #  define ARCHITECTURE_ID ""
00691 # endif
00692
00693 # elif defined(__ADSPSHARC__)
00694 #  define ARCHITECTURE_ID "SHARC"
00695
00696 # elif defined(__ADSPBLACKFIN__)
00697 #  define ARCHITECTURE_ID "Blackfin"
00698
00699 #elif defined(__TASKING__)
00700
00701 # if defined(__CTC__) || defined(__CPTC__)
00702 #  define ARCHITECTURE_ID "TriCore"
00703
00704 # elif defined(__CMCS__)
00705 #  define ARCHITECTURE_ID "MCS"
00706
00707 # elif defined(__CARM__)
00708 #  define ARCHITECTURE_ID "ARM"
00709
00710 # elif defined(__CARC__)
00711 #  define ARCHITECTURE_ID "ARC"
00712
00713 # elif defined(__C51__)
00714 #  define ARCHITECTURE_ID "8051"
00715
00716 # elif defined(__CPCP__)
00717 #  define ARCHITECTURE_ID "PCP"
00718
00719 # else
00720 #  define ARCHITECTURE_ID ""
00721 # endif
00722
00723 #else
00724 #  define ARCHITECTURE_ID
00725 #endif
00726
00727 /* Convert integer to decimal digit literals.  */
00728 #define DEC(n)                   \
00729   ('0' + (((n) / 10000000)%10)), \
00730   ('0' + (((n) / 1000000)%10)),  \
00731   ('0' + (((n) / 100000)%10)),   \
00732   ('0' + (((n) / 10000)%10)),    \
00733   ('0' + (((n) / 1000)%10)),     \
00734   ('0' + (((n) / 100)%10)),      \
```

```
00735   ('0' + (((n) / 10)%10)),       \
00736   ('0' +  ((n) % 10))
00737
00738 /* Convert integer to hex digit literals.  */
00739 #define HEX(n)                    \
00740   ('0' + ((n)»28 & 0xF)), \
00741   ('0' + ((n)»24 & 0xF)), \
00742   ('0' + ((n)»20 & 0xF)), \
00743   ('0' + ((n)»16 & 0xF)), \
00744   ('0' + ((n)»12 & 0xF)), \
00745   ('0' + ((n)»8  & 0xF)), \
00746   ('0' + ((n)»4  & 0xF)), \
00747   ('0' + ((n)     & 0xF))
00748
00749 /* Construct a string literal encoding the version number. */
00750 #ifdef COMPILER_VERSION
00751 char const* info_version = "INFO" ":" "compiler_version[" COMPILER_VERSION "]";
00752
00753 /* Construct a string literal encoding the version number components. */
00754 #elif defined(COMPILER_VERSION_MAJOR)
00755 char const info_version[] = {
00756   'I', 'N', 'F', 'O', ':',
00757   'c','o','m','p','i','l','e','r','_','v','e','r','s','i','o','n','[',
00758   COMPILER_VERSION_MAJOR,
00759 # ifdef COMPILER_VERSION_MINOR
00760   '.', COMPILER_VERSION_MINOR,
00761 #  ifdef COMPILER_VERSION_PATCH
00762    '.', COMPILER_VERSION_PATCH,
00763 #   ifdef COMPILER_VERSION_TWEAK
00764     '.', COMPILER_VERSION_TWEAK,
00765 #   endif
00766 #  endif
00767 # endif
00768   ']','\0'};
00769 #endif
00770
00771 /* Construct a string literal encoding the internal version number. */
00772 #ifdef COMPILER_VERSION_INTERNAL
00773 char const info_version_internal[] = {
00774   'I', 'N', 'F', 'O', ':',
00775   'c','o','m','p','i','l','e','r','_','v','e','r','s','i','o','n','_',
00776   'i','n','t','e','r','n','a','l','[',
00777   COMPILER_VERSION_INTERNAL,']','\0'};
00778 #elif defined(COMPILER_VERSION_INTERNAL_STR)
00779 char const* info_version_internal = "INFO" ":" "compiler_version_internal["
      COMPILER_VERSION_INTERNAL_STR "]";
00780 #endif
00781
00782 /* Construct a string literal encoding the version number components. */
00783 #ifdef SIMULATE_VERSION_MAJOR
00784 char const info_simulate_version[] = {
00785   'I', 'N', 'F', 'O', ':',
00786   's','i','m','u','l','a','t','e','_','v','e','r','s','i','o','n','[',
00787   SIMULATE_VERSION_MAJOR,
00788 # ifdef SIMULATE_VERSION_MINOR
00789   '.', SIMULATE_VERSION_MINOR,
00790 #  ifdef SIMULATE_VERSION_PATCH
00791    '.', SIMULATE_VERSION_PATCH,
00792 #   ifdef SIMULATE_VERSION_TWEAK
00793     '.', SIMULATE_VERSION_TWEAK,
00794 #   endif
00795 #  endif
00796 # endif
00797   ']','\0'};
00798 #endif
00799
00800 /* Construct the string literal in pieces to prevent the source from
00801    getting matched.  Store it in a pointer rather than an array
00802    because some compilers will just produce instructions to fill the
00803    array rather than assigning a pointer to a static array.  */
00804 char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]";
00805 char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]";
00806
00807
00808
00809 #if defined(__INTEL_COMPILER) && defined(_MSVC_LANG) && _MSVC_LANG < 201403L
00810 #  if defined(__INTEL_CXX11_MODE__)
00811 #    if defined(__cpp_aggregate_nsdmi)
00812 #      define CXX_STD 201402L
00813 #    else
00814 #      define CXX_STD 201103L
00815 #    endif
00816 #  else
00817 #    define CXX_STD 199711L
00818 #  endif
00819 #elif defined(_MSC_VER) && defined(_MSVC_LANG)
00820 #  define CXX_STD _MSVC_LANG
```

```
00821 #else
00822 #  define CXX_STD __cplusplus
00823 #endif
00824
00825 const char* info_language_standard_default = "INFO" ":" "standard_default["
00826 #if CXX_STD > 202002L
00827   "23"
00828 #elif CXX_STD > 201703L
00829   "20"
00830 #elif CXX_STD >= 201703L
00831   "17"
00832 #elif CXX_STD >= 201402L
00833   "14"
00834 #elif CXX_STD >= 201103L
00835   "11"
00836 #else
00837   "98"
00838 #endif
00839 "]";
00840
00841 const char* info_language_extensions_default = "INFO" ":" "extensions_default["
00842 #if (defined(__clang__) || defined(__GNUC__) || defined(__xlC__) ||           \
00843      defined(__TI_COMPILER_VERSION__)) &&                                     \
00844   !defined(__STRICT_ANSI__)
00845   "ON"
00846 #else
00847   "OFF"
00848 #endif
00849 "]";
00850
00851 /*--------------------------------------------------------------------------*/
00852
00853 int main(int argc, char* argv[])
00854 {
00855   int require = 0;
00856   require += info_compiler[argc];
00857   require += info_platform[argc];
00858   require += info_arch[argc];
00859 #ifdef COMPILER_VERSION_MAJOR
00860   require += info_version[argc];
00861 #endif
00862 #ifdef COMPILER_VERSION_INTERNAL
00863   require += info_version_internal[argc];
00864 #endif
00865 #ifdef SIMULATE_ID
00866   require += info_simulate[argc];
00867 #endif
00868 #ifdef SIMULATE_VERSION_MAJOR
00869   require += info_simulate_version[argc];
00870 #endif
00871 #if defined(__CRAYXT_COMPUTE_LINUX_TARGET)
00872   require += info_cray[argc];
00873 #endif
00874   require += info_language_standard_default[argc];
00875   require += info_language_extensions_default[argc];
00876   (void)argv;
00877   return require;
00878 }
```

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.

```
00001 #include "class_studentai.h"
00002 #include "class_funkcijos.h"
00003
00004
00006 void Netinkamas_Ivestis(std::string Problema)
00007 {
```

```
00008     std::cin.clear();
00009     std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
00010     std::cerr « Problema;
00011 }
00012
00014
00015 std::random_device rd;
00016 std::mt19937 generuoti(rd());
00017 std::uniform_int_distribution<int> nd_kiekis(5, 20);
00018 std::uniform_int_distribution<int> dis(1, 10);
00019 std::uniform_int_distribution<int> dis_lytis(0, 1);
00020
00021 std::vector<std::string> vardaiV = { "Jonas", "Petras", "Antanas", "Juozas", "Kazys", "Darius",
      "Linas", "Tomas", "Giedrius", "Marius" };
00022 std::vector<std::string> vardaiM = { "Ona", "Maryte", "Aldona", "Gabija", "Dalia", "Danute", "Asta",
      "Rasa", "Nijole", "Aiste", "Gabriele" };
00023 std::vector<std::string> pavardesV = { "Jonaitis", "Petraitis", "Antanaitis", "Juozaitis",
      "Kaziukaitis", "Dariukaitis", "Linaitis", "Tomaitis", "Giedraitis", "Mariukaitis" };
00024 std::vector<std::string> pavardesM = { "Jonaite", "Petraityte", "Antanaite", "Juozaite", "Kaziukaite",
      "Dariukaite", "Linaite", "Tomaite", "Giedraite", "Mariukaite", "Antaniene", "Jonaitiene", "Antaniene"
      };
00025
00026 int lytis = dis_lytis(generuoti);
00027
00029 void GeneruotiNDPazymius(studentas& S, int ND_kiekis)
00030 {
00031     std::vector<int> pazymiai;
00032     for (int i = 0; i < ND_kiekis; ++i) {
00033         pazymiai.push_back(dis(generuoti));
00034     }
00035     S.setND(pazymiai);
00036     S.setEGZ(dis(generuoti));
00037 }
00038
00040 void GeneruotiVardus(studentas& S)
00041 {
00042     int lytis = dis_lytis(generuoti);
00043
00044     if (lytis == 0)
00045     {
00046         S.setVardas(vardaiV[dis(generuoti) % 10]);
00047         S.setPavarde(pavardesV[dis(generuoti) % 10]);
00048     }
00049
00050     else
00051     {
00052         S.setVardas(vardaiM[dis(generuoti) % 10]);
00053         S.setPavarde(pavardesM[dis(generuoti) % 10]);
00054     }
00055
00056 }
00057
00059 void GeneruotiFailus(int reserveDydis, std::string& G_Failo_Vieta)
00060 {
00061     int nd_kiekis_gen = nd_kiekis(generuoti);
00062
00063     std::ofstream GFailas(G_Failo_Vieta);
00064
00065     if (!GFailas.is_open())
00066     {
00067         std::cout « "Nepavyko atidaryti failo " « G_Failo_Vieta « std::endl;
00068         return;
00069     }
00070
00071     //headline spausdinimas
00072     GFailas « std::left « std::setw(20) « "Pavarde" « std::setw(20) « "Vardas";
00073
00074     for (int i = 0; i < nd_kiekis_gen; i++)
00075     {
00076         GFailas « std::left « std::setw(7) « "ND" + std::to_string(i + 1);
00077     }
00078
00079     GFailas « std::setw(5) « "Egz." « std::endl;
00080
00081     //studentu duomenu spasudiniams
00082     for (int i = 0; i < reserveDydis; i++)
00083     {
00084         GFailas « std::left « std::setw(20) « "Pavarde" + std::to_string(i + 1) « std::left «
      std::setw(20) « "Vardas" + std::to_string(i + 1);
00085         for (int j = 0; j < nd_kiekis_gen; j++)
00086         {
00087             GFailas « std::setw(7) « dis(generuoti);
00088         }
00089         GFailas « std::setw(7) « dis(generuoti);
00090         GFailas « "\n";
00091     }
00092
```

```
00093     GFailas.close();
00094
00095 }
00096
00098
00099 void Ivesti_Pazymius(studentas& S)
00100 {
00101     std::vector<int> pazymiai;
00102     char TaipNePaz;
00103     std::cout « "\nIveskite namu darbu pazymi: ";
00104     int pazymys;
00105     do
00106     {
00107         while (true)
00108         {
00109             try
00110             {
00111                 std::cin » pazymys;
00112                 if (std::cin.fail() || std::cin.peek() != '\n' || pazymys < 1 || pazymys > 10)
00113                 {
00114                     throw std::invalid_argument("Netinkama ivestis. Iveskite sveikaji skaiciu nuo 1
      iki 10. ");
00115                 }
00116
00117                 if (pazymys >= 1 && pazymys <= 10)
00118                 {
00119                     pazymiai.push_back(pazymys);// pridedamas pazymis i vektoriu
00120                 }
00121                 break;
00122             }
00123
00124             catch (const std::invalid_argument& paz)
00125             {
00126                 Netinkamas_Ivestis(paz.what());
00127             }
00128         }
00129
00130         std::cout « "Ar norite ivesti dar viena pazymi? (iveskite T, jei taip , N, jei ne): ";
00131         while (true)
00132         {
00133             try
00134             {
00135                 std::cin » TaipNePaz;
00136                 if ((std::cin.fail() || std::cin.peek() != '\n') || (TaipNePaz != 'T' && TaipNePaz !=
      'N'))
00137                 {
00138                     throw std::invalid_argument("Netinkama ivestis. Iveskite T arba N.  ");
00139                 }
00140
00141                 break;
00142             }
00143             catch (const std::invalid_argument& tpp)
00144             {
00145                 Netinkamas_Ivestis(tpp.what());
00146             }
00147         }
00148
00149         if (TaipNePaz == 'T')
00150             std::cout « std::endl « "Iveskite namu darbu pazymi: ";
00151
00152     } while (TaipNePaz == 'T');
00153
00154     S.setND(pazymiai);
00155
00156     int egz;
00157     std::cout « std::endl « "Iveskite egzamino pazymi: ";
00158     while (true)
00159     {
00160         try
00161         {
00162             std::cin » egz;
00163             if (std::cin.fail() || std::cin.peek() != '\n' || egz < 1 || egz > 10)
00164             {
00165                 throw std::invalid_argument("Netinkama ivestis. Iveskite sveikaji skaiciu nuo 1 iki
      10. ");
00166             }
00167             break;
00168         }
00169         catch (const std::invalid_argument& e)
00170         {
00171             Netinkamas_Ivestis(e.what());
00172         }
00173     }
00174     S.setEGZ(egz);
00175 }
00176
00177 void Ivesti_Varda(studentas& S)
```

```
00178 {
00179     std::string vardas, pavarde;
00180     std::cout « std::endl « "Iveskite studento varda: ";
00181     while (true)
00182     {
00183         try
00184         {
00185             std::cin » vardas;
00186             if (std::cin.fail() || std::cin.peek() != '\n' || !all_of(vardas.begin(), vardas.end(),
      ::isalpha))
00187             {
00188                 throw std::invalid_argument("Netinkama ivestis. Iveskite varda, sudaryta tik is
      raidziu. ");
00189             }
00190
00191             break;
00192         }
00193         catch (const std::invalid_argument& v)
00194         {
00195             Netinkamas_Ivestis(v.what());
00196         }
00197     }
00198
00199     std::cout « std::endl « "Iveskite studento pavarde: ";
00200     while (true)
00201     {
00202         try
00203         {
00204             std::cin » pavarde;
00205             if (std::cin.fail() || std::cin.peek() != '\n' || !all_of(pavarde.begin(), pavarde.end(),
      ::isalpha))
00206             {
00207                 throw std::invalid_argument("Netinkama ivestis. Iveskite pavarde, sudaryta tik is
      raidziu. ");
00208             }
00209
00210             break;
00211         }
00212         catch (const std::invalid_argument& pv)
00213         {
00214             Netinkamas_Ivestis(pv.what());
00215         }
00216     }
00217     S.setVardas(vardas);
00218     S.setPavarde(pavarde);
00219 }
00220
00222
00223 std::vector<studentas> Nuskaityti_Is_Failo(const std::string& Failo_Pavadinimas, int reserveDydis)
00224 {
00225     // Pradedamas skaiciuti laikas
00226     auto start = std::chrono::high_resolution_clock::now();
00227
00228     std::ifstream file(Failo_Pavadinimas);
00229     std::vector<studentas> studentai;
00230     studentai.reserve(reserveDydis);  // Iš anksto rezervuojama atmintis
00231
00232     if (!file.is_open())
00233     {
00234         std::cerr « "Klaida atidarant faila " « Failo_Pavadinimas « std::endl;
00235         return studentai;
00236     }
00237     // Praleidziama pirma header eilute
00238     std::string header;
00239     std::getline(file, header);
00240
00241     std::string eilute;
00242     while (std::getline(file, eilute))
00243     {
00244         std::istringstream iss(eilute);
00245         studentas s;
00246         if (iss » s)
00247         {
00248             studentai.push_back(s);
00249         }
00250         else
00251         {
00252             std::cerr « "Klaida nuskaitant duomenis is eilutes: " « eilute « std::endl;
00253         }
00254     }
00255
00256     // Baigia skaiciuoti laika
00257     auto end = std::chrono::high_resolution_clock::now();
00258
00259     //Apskaiciuoja laika
00260     auto duration = std::chrono::duration_cast<std::chrono::duration<double»(end - start);
00261
```

```
00262     file.close();
00263     std::cout « "\nFailo nuskaitymas uztruko " « duration.count() « " sek." « std::endl;
00264     return studentai;
00265 }
00266
00268
00269 void Rikiuoti_Duomenis(std::vector<studentas>& S)
00270 {
00271
00272     // Rusiavimo pasirinkimai
00273     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: ";
00274     int Rusiavimo_Pasirinkimas;
00275     while (true)
00276     {
00277         try
00278         {
00279             std::cin » Rusiavimo_Pasirinkimas;
00280
00281             if (std::cin.fail() || std::cin.peek() != '\n' || Rusiavimo_Pasirinkimas < 1 ||
      Rusiavimo_Pasirinkimas > 4)
00282             {
00283                 throw std::invalid_argument("Netinkama ivestis. Iveskite sveikaji skaiciu nuo 1 iki 4.
      ");
00284             }
00285             break;
00286         }
00287         catch (const std::invalid_argument& rp)
00288         {
00289             Netinkamas_Ivestis(rp.what());
00290
00291
00292         }
00293
00294     }
00295     // Pradedamas skaiciuoti laikas
00296     auto RikiavimoPradzia = std::chrono::high_resolution_clock::now();
00297     switch (Rusiavimo_Pasirinkimas)
00298     {
00299     case 1:
00300         std::sort(S.begin(), S.end(), [](const studentas& a, const studentas& b)
00301             {
00302                 return a.getVardas() < b.getVardas(); // Rûðiuojama pagal vardà
00303             });
00304
00305         break;
00306     case 2:
00307         std::sort(S.begin(), S.end(), [](const studentas& a, const studentas& b)
00308             {
00309                 return a.getPavarde() < b.getPavarde(); // Rûðiuojama pagal pavarde
00310             });
00311         break;
00312     case 3:
00313         std::sort(S.begin(), S.end(), [](const studentas& a, const studentas& b)
00314             {
00315                 return a.getGalutinisM() < b.getGalutinisM(); // Rûðiuojama pagal GalutiniM
00316             });
00317         break;
00318     case 4:
00319         std::sort(S.begin(), S.end(), [](const studentas& a, const studentas& b)
00320             {
00321                 return a.getGalutinisV() < b.getGalutinisV(); // Rûðiuojama pagal GalutiniV
00322             });
00323         break;
00324     }
00325     // Baigia skaiciuoti laika
00326     auto RikiavimoPabaiga = std::chrono::high_resolution_clock::now();
00327
00328     //Apskaiciuoja laika
00329     auto Rikiavimotrukme = std::chrono::duration_cast<std::chrono::duration<double»(RikiavimoPabaiga -
      RikiavimoPradzia);
00330
00331     std::cout « "\nRikiavimas didejancia tvarka pagal pasirinkta kriteriju uztruko " «
      Rikiavimotrukme.count() « " sek." « std::endl;
00332 }
00333
00335 void Skirstyti_Studentus(std::vector<studentas>& S, std::vector<studentas>& N, std::vector<studentas>&
      G, int Strategija)
00336 {
00337
00338     std::cout « "\nAr norite studentus surusiuoti pagal mediana ar vidurki? M jei mediana, V jei
      vidurki: ";
00339     char RusiavimoPasirinkimas;
00340     while (true)
00341     {
00342         try
00343         {
```

```
00344             std::cin » RusiavimoPasirinkimas;
00345             if (std::cin.fail() || std::cin.peek() != '\n' || (RusiavimoPasirinkimas != 'V' &&
    RusiavimoPasirinkimas != 'M'))
00346             {
00347                 throw std::invalid_argument("Netinkama ivestis. Iveskite M arba V: ");
00348             }
00349             break;
00350         }
00351         catch (const std::invalid_argument& rp)
00352         {
00353             Netinkamas_Ivestis(rp.what());
00354         }
00355     }
00356
00357     // Pradedamas skaiciuti laikas
00358     auto RusavimoPradzia = std::chrono::high_resolution_clock::now();
00359
00360
00361     if (Strategija == 1)
00362     {
00363
00364         for (auto& studentas : S)
00365         {
00366             if (RusiavimoPasirinkimas == 'V')
00367             {
00368                 if (studentas.getGalutinisV() < 5)
00369                     N.push_back(studentas);
00370                 else
00371                     G.push_back(studentas);
00372             }
00373             else if (RusiavimoPasirinkimas == 'M')
00374             {
00375                 if (studentas.getGalutinisM() < 5)
00376                     N.push_back(studentas);
00377                 else
00378                     G.push_back(studentas);
00379             }
00380         }
00381     }
00382     if (Strategija == 2)
00383     {
00384
00385         auto i = S.begin();
00386         while (i != S.end())
00387         {
00388             if (RusiavimoPasirinkimas == 'V')
00389             {
00390                 if (i->getGalutinisV() < 5)
00391                 {
00392                     N.push_back(*i);
00393                     i = S.erase(i);
00394                     continue;
00395                 }
00396             }
00397             else if (RusiavimoPasirinkimas == 'M')
00398             {
00399                 if (i->getGalutinisM() < 5)
00400                 {
00401                     N.push_back(*i);
00402                     i = S.erase(i);
00403                     continue;
00404                 }
00405             }
00406             ++i;
00407         }
00408     }
00409
00410     if (Strategija == 3)
00411     {
00412         auto i = std::remove_if(S.begin(), S.end(), [&](const auto& studentas)
00413             {
00414                 bool istrinti = false;
00415                 if (RusiavimoPasirinkimas == 'V')
00416                 {
00417                     if (studentas.getGalutinisV() < 5)
00418                     {
00419                         istrinti = true;
00420                     }
00421                 }
00422                 else if (RusiavimoPasirinkimas == 'M')
00423                 {
00424                     if (studentas.getGalutinisM() < 5)
00425                     {
00426                         istrinti = true;
00427                     }
00428                 }
00429                 if (istrinti)
```

```
00430                 {
00431                     N.push_back(studentas);
00432                 }
00433                 return istrinti;
00434             });
00435
00436         S.erase(i, S.end());
00437
00438     }
00439
00440     // Baigia skaiciuoti laika
00441     auto RusaivimoPabaiga = std::chrono::high_resolution_clock::now();
00442
00443     //Apskaiciuoja laika
00444     auto Rusiavimotrukme = std::chrono::duration_cast<std::chrono::duration<double»(RusaivimoPabaiga -
      RusavimoPradzia);
00445
00446     std::cout « "\nRusiavimas i galvocius ir nepazangius uztruko " « Rusiavimotrukme.count() « " sek."
      « std::endl;
00447 }
00448
00450 void Spausdinti_Rezultatus(const std::vector<studentas>& N, const std::vector<studentas>& G)
00451 {
00452     std::ofstream Galvociu_failas("Galvociai.txt");
00453     if (!Galvociu_failas.is_open())
00454     {
00455         std::cerr « "Klaida atidarant rezultatu faila" « std::endl;
00456         return;
00457     }
00458     int i = 0;
00459     for (auto& studentas : G)
00460     {
00461         if (i == 0)
00462             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;
00463
00464         Galvociu_failas « std::setw(7) « i + 1 « studentas;
00465         i++;
00466
00467     }
00468     Galvociu_failas.close();
00469
00470     std::ofstream Nepazangiuju_failas("Nepazangus.txt");
00471     if (!Nepazangiuju_failas.is_open())
00472     {
00473         std::cerr « "Klaida atidarant rezultatu faila" « std::endl;
00474         return;
00475     }
00476     i = 0;
00477     for (auto& studentas : N)
00478     {
00479         if (i == 0)
00480             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;
00481
00482         Nepazangiuju_failas « std::setw(7) « i + 1 « studentas;
00483         i++;
00484
00485     }
00486     Nepazangiuju_failas.close();
00487
00488     std::cout « std::endl « "Rezultatai atspausdinti" « std::endl;
00489 }
00490
00491 void Testavimas()
00492 {
00493     // Testuojamas default konstruktorius
00494     {
00495         std::cout « "\n1. Testuojamas default konstruktorius\n\n";
00496         studentas s;
00497         std::cout « std::endl;
00498     }
00499
00500     // Testuojamas parametrizuotas konstruktorius
00501     {
00502         std::cout « "\n2.  Testuojamas parametrizuotas konstruktorius\n\n";
00503         std::string vardas = "Jonas";
00504         std::string pavarde = "Jonaitis";
00505         std::vector<int> nd = { 5, 7, 8 };
00506         int egz = 9;
00507         studentas s(vardas, pavarde, nd, egz);
00508         std::cout « std::endl;
00509     }
00510
00511     // Testuojamas copy  konstruktorius
```

```
00512     {
00513         std::cout « "\n3. Testuojamas copy konstruktorius\n\n";
00514         studentas s1("Petras", "Petraitis", { 10, 9, 8 }, 10);
00515         studentas s2 = s1;
00516         std::cout « std::endl;
00517     }
00518
00519     // Testuojamas move konstruktorius
00520     {
00521         std::cout « "\n4. Testuojamas move konstruktorius\n\n";
00522         studentas s1("Kazys", "Kazlauskas", { 6, 5, 7 }, 8);
00523         studentas s2 = std::move(s1);
00524         std::cout « std::endl;
00525     }
00526
00527     // Testuojamas kopijavimo priskyrimo operatorius
00528     {
00529         std::cout « "\n5. Testuojamas copy priskyrimo operatorius\n\n";
00530         studentas s1, s2;
00531         s2 = s1;
00532         std::cout « std::endl;
00533     }
00534
00535     // Testuojamas move priskyrimo operatorius
00536     {
00537         std::cout « "\n6. Testuojamas move priskyrimo operatorius\n\n";
00538         studentas s3, s2;
00539         s3 = std::move(s2);
00540         std::cout « std::endl;
00541     }
00542
00543     // Destruktoriaus patikrinimas
00544     {
00545         std::cout « "\n7. Destruktoriaus patikrinimas\n\n";
00546         // Sukuriamas dynamic studentas
00547         studentas* s1 = new studentas();
00548
00549         // I ji pridedamepazymius
00550         s1->setND({ 10, 9, 8 });
00551
00552         // Istriname
00553         delete s1;
00554
00555         // Sukuriam nauja
00556         studentas s2;
00557
00558         // Patikrinam ar jame nebeliko s1 pazymiu
00559         assert(s2.getND().empty());
00560         std::cout « std::endl;
00561
00562     }
00563     // Testuojamas ivesties operatorius
00564     {
00565         std::cout « "\n8. Testuojamas ivesties operatorius\n\n";
00566         std::vector<int> I = { 5, 6, 7, 8 };//toki ND vector turi gauti
00567         std::istringstream iss("Mindaugas Mindaugaitis 5 6 7 8 9");
00568         studentas s;
00569         iss » s;
00570         assert(s.getVardas() == "Mindaugas");
00571         assert(s.getPavarde() == "Mindaugaitis");
00572         assert(s.getND() == I);
00573         assert(s.getEGZ() == 9);
00574         std::cout « std::endl;
00575     }
00576     // Testuojamas ivesties operatorius
00577     {
00578         std::cout « "\n9. Testuojamas ivesties operatorius\n\n";
00579         std::istringstream iss("Lina Linaityte 4 5 9 9");
00580         studentas s;
00581         iss » s;
00582         std::ostringstream oss;
00583         oss « s;
00584         std::string tikimasi = "           Linaityte                Lina                 7.8
      7.4\n";
00585         assert(oss.str() == tikimasi);
00586         std::cout « std::endl;
00587     }
00588
00589     //realizuota abstrakti klasë zmogus, jos objektø kûrimas negalimas (pademonstruota).
00590     {
00591         //zmogus z;
00592
00593     }
00594 }
```

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.

```
00001 #ifndef CLASS_FUNKCIJOS_H
00002 #define CLASS_FUNKCIJOS_H
00003 #include "class_studentai.h"
00004
00006 void Netinkamas_Ivestis(std::string Problema);
00007
00009 void GeneruotiNDPazymius(studentas& S, int ND_kiekis);
00010 void GeneruotiVardus(studentas& S);
00011 void GeneruotiFailus(int reserveDydis, std::string& failoPav);
00012
00014 void Ivesti_Pazymius(studentas& S);
00015 void Ivesti_Varda(studentas& S);
00016
00018 std::vector<studentas> Nuskaityti_Is_Failo(const std::string& Failo_Pavadinimas, int reserveDydis);
00019
00021 bool VarduRikiavimas(const studentas& a, const studentas& b);
00022 bool PavardziuRikiavimas(const studentas& a, const studentas& b);
00023 bool MedianuRikiavimas(const studentas& a, const studentas& b);
00024 bool VidurkiuRikiavimas(const studentas& a, const studentas& b);
00025 void Rikiuoti_Duomenis(std::vector<studentas>& S);
00026
00028 void Skirstyti_Studentus(std::vector<studentas>& S, std::vector<studentas>& N, std::vector<studentas>&
      G, int Strategija);
00029
00031 void Spausdinti_Rezultatus(const std::vector<studentas>& N, const std::vector<studentas>& G);
00032
00034 void Testavimas();
00035
00036 #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
```

```
00064             S.push_back(naujas);// pridedamas studentas i vektoriu
00065
00066             std::cout « std::endl « "Ar norite ivesti " « S.size() + 1 « " studenta ? (T jei taip, N -
      ne) : ";
00067             while (true)
00068             {
00069                 try
00070                 {
00071                     std::cin » TaipNe;
00072                     if ((std::cin.fail() || std::cin.peek() != '\n') || (TaipNe != 'T' && TaipNe !=
      'N'))
00073                     {
00074                         throw std::invalid_argument("Netinkama ivestis. Iveskite T arba N.  ");
00075                     }
00076                     break;
00077                 }
00078                 catch (const std::invalid_argument& tp)
00079                 {
00080                     Netinkamas_Ivestis(tp.what());
00081                 }
00082             }
00083
00084         } while (TaipNe == 'T');
00085
00086         Rikiuoti_Duomenis(S);
00087         Skirstyti_Studentus(S, N, G, Strategija);
00088
00089         if (Strategija == 1)
00090             Spausdinti_Rezultatus(N, G);
00091
00092         if (Strategija == 2)
00093             Spausdinti_Rezultatus(N, S);
00094
00095         if (Strategija == 3)
00096             Spausdinti_Rezultatus(N, S);
00097     }
00098
00099     if (Pasirinkimas == 2)
00100     {
00101
00102         studentas naujas;
00103         do
00104         {
00105             Ivesti_Varda(naujas);
00106
00107             std::cout « std::endl « "Kiek namu darbu pazymiu norite sugeneruoti: ";
00108             int ND_kiekis;
00109             while (true)
00110             {
00111                 try
00112                 {
00113                     std::cin » ND_kiekis;
00114                     if (std::cin.fail() || std::cin.peek() != '\n')
00115                     {
00116                         throw std::invalid_argument("Netinkama ivestis. Iveskite sveikaji skaiciu. ");
00117                     }
00118                     break;
00119                 }
00120                 catch (const std::invalid_argument& ndk)
00121                 {
00122                     Netinkamas_Ivestis(ndk.what());
00123                 }
00124             }
00125
00126
00127             GeneruotiNDPazymius(naujas, ND_kiekis);
00128
00129             S.push_back(naujas); // pridedamas studentas i vektoriu
00130             std::cout « std::endl « "Ar norite ivesti " « S.size() + 1 « " studenta ? (T jei taip, N -
      ne) : ";
00131             while (true)
00132             {
00133                 try
00134                 {
00135                     std::cin » TaipNe;
00136                     if ((std::cin.fail() || std::cin.peek() != '\n') || (TaipNe != 'T' && TaipNe !=
      'N'))
00137                     {
00138                         throw std::invalid_argument("Netinkama ivestis. Iveskite T arba N.  ");
00139                     }
00140                     break;
00141                 }
00142                 catch (const std::invalid_argument& tp)
00143                 {
00144                     Netinkamas_Ivestis(tp.what());
00145                 }
00146             }
```

```
00147
00148
00149         } while (TaipNe == 'T');
00150
00151         Rikiuoti_Duomenis(S);
00152         Skirstyti_Studentus(S, N, G, Strategija);
00153
00154         if (Strategija == 1)
00155             Spausdinti_Rezultatus(N, G);
00156
00157         if (Strategija == 2)
00158             Spausdinti_Rezultatus(N, S);
00159
00160         if (Strategija == 3)
00161             Spausdinti_Rezultatus(N, S);
00162     }
00163
00164     if (Pasirinkimas == 3)
00165     {
00166
00167         studentas naujas;
00168
00169         std::cout « std::endl « "Kiek studentu norite sugeneruoti: ";
00170         int Studentu_kiekis;
00171         while (true)
00172         {
00173             try
00174             {
00175                 std::cin » Studentu_kiekis;
00176                 if (std::cin.fail() || std::cin.peek() != '\n')
00177                 {
00178                     throw std::invalid_argument("Netinkama ivestis. Iveskite sveikaji skaiciu. ");
00179                 }
00180                 break;
00181             }
00182             catch (const std::invalid_argument& sk)
00183             {
00184                 Netinkamas_Ivestis(sk.what());
00185
00186
00187             }
00188
00189         }
00190
00191         std::cout « std::endl « "Kiek namu darbu pazymiu norite sugeneruoti: ";
00192         int ND_kiekis;
00193         while (true)
00194         {
00195             try
00196             {
00197                 std::cin » ND_kiekis;
00198                 if (std::cin.fail() || std::cin.peek() != '\n')
00199                 {
00200                     throw std::invalid_argument("Netinkama ivestis. Iveskite sveikaji skaiciu. ");
00201                 }
00202                 break;
00203             }
00204             catch (const std::invalid_argument& ndk)
00205             {
00206                 Netinkamas_Ivestis(ndk.what());
00207
00208
00209             }
00210         }
00211
00212         for (int i = 0; i < Studentu_kiekis; ++i)
00213         {
00214
00215             GeneruotiVardus(naujas);
00216
00217             GeneruotiNDPazymius(naujas, ND_kiekis);
00218
00219
00220             S.push_back(naujas); // pridedamas studentas i vektoriu
00221         }
00222
00223         Rikiuoti_Duomenis(S);
00224
00225         Skirstyti_Studentus(S, N, G, Strategija);
00226
00227         if (Strategija == 1)
00228             Spausdinti_Rezultatus(N, G);
00229
00230         if (Strategija == 2)
00231             Spausdinti_Rezultatus(N, S);
00232
00233         if (Strategija == 3)
```

```
00234             Spausdinti_Rezultatus(N, S);
00235     }
00236
00237     if (Pasirinkimas == 4)
00238     {
00239         int Failo_Pasirinkimas;
00240         std::string Failas;
00241
00242         while (true)
00243         {
00244             // Parinkimo meniu ir failo pasirinkimas
00245             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: ";
00246
00247             std::cin » Failo_Pasirinkimas;
00248             // Tikrinimas ar įvestis yra tinkama
00249             if (std::cin.fail() || std::cin.peek() != '\n' || Failo_Pasirinkimas < 1 ||
      Failo_Pasirinkimas > 5)
00250             {
00251                 throw std::invalid_argument("Netinkama ivestis. Iveskite sveikaji skaiciu nuo 1 iki 5.
      ");
00252             }
00253             int reserveDydis = 0;
00254             // Nuskaitomas pasirinkto failo kelias
00255             switch (Failo_Pasirinkimas)
00256             {
00257             case 1:
00258                 Failas = "Studentai1000.txt";
00259                 reserveDydis = 1000;
00260                 break;
00261             case 2:
00262                 Failas = "Studentai10000.txt";
00263                 reserveDydis = 10000;
00264                 break;
00265             case 3:
00266                 Failas = "Studentai100000.txt";
00267                 reserveDydis = 100000;
00268                 break;
00269             case 4:
00270                 Failas = "Studentai1000000.txt";
00271                 reserveDydis = 1000000;
00272                 break;
00273
00274             case 5:
00275                 Failas = "Studentai10000000.txt";
00276                 reserveDydis = 10000000;
00277                 break;
00278
00279             }
00280             // Tikrinimas ar pasirinktas failas egzistuoja
00281             if (!fs::exists(Failas))
00282             {
00283                 Netinkamas_Ivestis("Pasirinktas failas neegzistuoja. Pasirinkite kita faila. ");
00284                 std::cout « "\n";
00285                 continue;
00286
00287             }
00288             // Nuskaitymas duomenų iš pasirinkto failo
00289             S = Nuskaityti_Is_Failo(Failas, reserveDydis);
00290             Rikiuoti_Duomenis(S);
00291             Skirstyti_Studentus(S, N, G, Strategija);
00292
00293             if (Strategija == 1)
00294                 Spausdinti_Rezultatus(N, G);
00295
00296             if (Strategija == 2)
00297                 Spausdinti_Rezultatus(N, S);
00298
00299             if (Strategija == 3)
00300                 Spausdinti_Rezultatus(N, S);
00301             break; // Išeiti iš ciklo, kai buna pasirinktas tinkamas failas
00302         }
00303
00304     }
00305
00306     if (Pasirinkimas == 5)
00307     {
00308         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: ";
00309
00310         std::string G_Failas;
00311         int G_Failo_Pasirinkimas;
00312         int reserveDydis = 0;
00313         while (true)
00314         {
00315             try
00316             {
```

```
00317                 std::cin » G_Failo_Pasirinkimas;
00318                 if (std::cin.fail() || std::cin.peek() != '\n' || G_Failo_Pasirinkimas < 1 ||
      G_Failo_Pasirinkimas > 5)
00319                 {
00320                     throw std::invalid_argument("Netinkama ivestis. Iveskite sveikaji skaiciu nuo 1
      iki 5. ");
00321                 }
00322                 break;
00323             }
00324             catch (const std::invalid_argument& pfg)
00325             {
00326                 Netinkamas_Ivestis(pfg.what());
00327             }
00328         }
00329
00330
00331         switch (G_Failo_Pasirinkimas)
00332         {
00333         case 1:
00334             G_Failas = "Studentai1000.txt";
00335             reserveDydis = 1000;
00336             break;
00337         case 2:
00338             G_Failas = "Studentai10000.txt";
00339             reserveDydis = 10000;
00340             break;
00341         case 3:
00342             G_Failas = "Studentai100000.txt";
00343             reserveDydis = 100000;
00344             break;
00345         case 4:
00346             G_Failas = "Studentai1000000.txt";
00347             reserveDydis = 1000000;
00348             break;
00349
00350         case 5:
00351             G_Failas = "Studentai10000000.txt";
00352             reserveDydis = 10000000;
00353             break;
00354
00355         }
00356
00357         //jei failas jau egzistuoja, tiesiog nuskaitoma nuo jo
00358         if (fs::exists(G_Failas))
00359         {
00360             std::cout « "Pasirinktas failas jau egzistuoja, dabar nuo jo bus nuskaitoma";
00361             S = Nuskaityti_Is_Failo(G_Failas, reserveDydis);
00362         }
00363         else //jei failas neegzistuoja, tai ji sugeneruoja ir tada nuskaito
00364         {
00365             std::cout « "Pasirinktas failas neegzistuoja, jis generuojamas";
00366             GeneruotiFailus(reserveDydis, G_Failas);
00367             S = Nuskaityti_Is_Failo(G_Failas, reserveDydis);
00368         }
00369         Rikiuoti_Duomenis(S);
00370         Skirstyti_Studentus(S, N, G, Strategija);
00371
00372         if (Strategija == 1)
00373             Spausdinti_Rezultatus(N, G);
00374
00375         if (Strategija == 2)
00376             Spausdinti_Rezultatus(N, S);
00377
00378         if (Strategija == 3)
00379             Spausdinti_Rezultatus(N, S);
00380     }
00381
00382     if (Pasirinkimas == 6)
00383         std::cout « "\n" « "Darbas baigtas";
00384
00385     if (Pasirinkimas == 7)
00386         Testavimas();
00387
00388
00389
00390     std::cout « "\n";
00391     return 0;
00392 }
```

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.

```
00001 #ifndef CLASS_STUDENTAI_H
00002 #define CLASS_STUDENTAI_H
00003
00004 #include <chrono>
00005 #include <string>
00006 #include <vector>
00007 #include <iostream>
00008 #include <sstream>
00009 #include <fstream>
00010 #include <stdexcept>
00011 #include <limits>
00012 #include <filesystem>
00013 #include <algorithm>
00014 #include <numeric>
00015 #include <random>
00016 #include <cassert>
00017
00018 //ZMOGUS
00019 class zmogus {
00020 protected:
00021     std::string vardas;
00022     std::string pavarde;
00023 public:
00024     //Konstruktorius
00025     zmogus() : vardas("Bevardis"), pavarde("Bepavardis") { /* std::cout « "Suveike zmogus default
      konstruktorius\n"; */ }
00026     zmogus(const std::string& vardas, const std::string& pavarde)
00027         : vardas(vardas), pavarde(pavarde) {}
00028
00029     ~zmogus() {}
00030
00031     virtual std::string getVardas() const = 0;
00032     virtual std::string getPavarde() const = 0;
00033
00034     // Setter'iai
00035     virtual void setVardas(const std::string& newName) { vardas = newName; }
00036     virtual void setPavarde(const std::string& newSurname) { pavarde = newSurname; }
00037
00038 };
00039
00040 //STUDENTAS
00041 class  studentas : public zmogus {
00042 private:
```

```
00043
00044     std::vector<int> ND;
00045     int EGZ;
00046     double GalutinisV;
00047     double GalutinisM;
00048     void ApskaiciuotiGalutinius()
00049     {
00050         if (!ND.empty())
00051         {
00052             GalutinisV = 0.4 * std::accumulate(ND.begin(), ND.end(), 0.0) / ND.size() + 0.6 * EGZ;
00053             if (ND.size() > 1)
00054             {
00055                 std::vector<int> sortedND = ND;
00056                 std::sort(sortedND.begin(), sortedND.end());
00057                 size_t mid = sortedND.size() / 2;
00058                 GalutinisM = 0.4 * (sortedND.size() % 2 == 0 ? (sortedND[mid - 1] + sortedND[mid]) /
      2.0 : sortedND[mid]) + 0.6 * EGZ;
00059             }
00060             else
00061                 GalutinisM = 0.4 * ND[0] + 0.6 * EGZ;
00062         }
00063         else
00064         {
00065             // Jei ND tuščias
00066             GalutinisV = GalutinisM = 0.6 * EGZ;
00067         }
00068     }
00069
00070 public:
00071     studentas() : EGZ(0), ND(), GalutinisV(0), GalutinisM(0) {
00072         //std::cout << "Suveike studentas default konstruktorius\n";
00073     }
00074
00075     studentas(const std::string& vardas, const std::string& pavarde, const std::vector<int>& ND, int
      EGZ)
00076         : zmogus(vardas, pavarde), ND(ND), EGZ(EGZ) {
00077         ApskaiciuotiGalutinius();
00078         //std::cout << "Suveike parametrizuotas konstruktorius\n";
00079     }
00080
00081
00082     // Implementuojame abstrakčius metodus
00083     virtual std::string getVardas() const override {
00084         return vardas;
00085     }
00086
00087     virtual std::string getPavarde() const override {
00088         return pavarde;
00089     }
00090     // Destruktorius
00091     ~studentas() { ND.clear();  /*std::cout << "Suveike destruktorius\n";*/ }
00092
00093     // Copy konstruktorius
00094     studentas(const studentas& other)
00095     {
00096         vardas = other.vardas;
00097         pavarde = other.pavarde;
00098         ND = other.ND;
00099         EGZ = other.EGZ;
00100         GalutinisV = other.GalutinisV;
00101         GalutinisM = other.GalutinisM;
00102         //std::cout << "Suveike copy konstruktorius\n";
00103     }
00104     // Move konstruktorius
00105     studentas(studentas&& other) noexcept
00106     {
00107         vardas = std::move(other.vardas);
00108         pavarde = std::move(other.pavarde);
00109         ND = std::move(other.ND);
00110         EGZ = std::move(other.EGZ);
00111         GalutinisV = std::move(other.GalutinisV);
00112         GalutinisM = std::move(other.GalutinisM);
00113         other.clearEverything();
00114         //std::cout << "Suveike move konstruktorius\n";
00115     }
00116     // Copy priskyrimo operatorius
00117     studentas& operator=(const studentas& other)
00118     {
00119
00120         if (this != &other)
00121         {
00122             vardas = other.vardas;
00123             pavarde = other.pavarde;
00124             ND = other.ND;
00125             EGZ = other.EGZ;
00126             GalutinisV = other.GalutinisV;
00127             GalutinisM = other.GalutinisM;
```

```
00128             //std::cout « "Suveike copy priskyrimo operatorius\n";
00129         }
00130         return *this;
00131     }
00132     // Move priskyrimo operatorius
00133     studentas& operator=(studentas&& other) noexcept
00134     {
00135
00136         if (this != &other)
00137         {
00138             vardas = std::move(other.vardas);
00139             pavarde = std::move(other.pavarde);
00140             ND = std::move(other.ND);
00141             EGZ = std::move(other.EGZ);
00142             GalutinisV = std::move(other.GalutinisV);
00143             GalutinisM = std::move(other.GalutinisM);
00144             other.clearEverything();
00145             //std::cout « "Suveike move priskyrimo operatorius\n";
00146         }
00147         return *this;
00148     }
00149
00150
00151     // Getter'iai
00152     std::vector<int> getND() const { return ND; }
00153     int getEGZ() const { return EGZ; }
00154     double getGalutinisV() const { return GalutinisV; }
00155     double getGalutinisM() const { return GalutinisM; }
00156
00157     // Setter'iai
00158     void setVardas(const std::string& newName) { vardas = newName; }
00159     void setPavarde(const std::string& newSurname) { pavarde = newSurname; }
00160     void setND(const std::vector<int>& newND) { ND = newND; ApskaiciuotiGalutinius(); }
00161     void setEGZ(int newEGZ) {
00162         EGZ = newEGZ;
00163         ApskaiciuotiGalutinius();
00164     }
00165
00166
00167     friend std::istream& operator»(std::istream& is, studentas& s)
00168     {
00169         s.vardas.clear();
00170         s.pavarde.clear();
00171         s.ND.clear();
00172         s.EGZ = 0;
00173
00174
00175         if (!(is » s.vardas » s.pavarde))
00176         {
00177             return is;
00178         }
00179
00180         int pazymys;
00181         std::vector<int> NDpazymiai;
00182         while (is » pazymys)
00183         {
00184             NDpazymiai.push_back(pazymys);
00185         }
00186
00187         // Patikrina, ar pasiekė failo pabaigą
00188         if (is.eof()) {
00189             is.clear();
00190         }
00191         // Jei įvedimo operacija nepavyko
00192         else if (is.fail()) {
00193
00194             is.clear();
00195             std::string unused;
00196             std::getline(is, unused);
00197             return is;
00198         }
00199
00200         if (!NDpazymiai.empty())
00201         {
00202             s.EGZ = NDpazymiai.back();
00203             NDpazymiai.pop_back();
00204             s.ND = NDpazymiai;
00205         }
00206
00207         s.ApskaiciuotiGalutinius();
00208         //std::cout « "Suveike ivesties operatorius\n";
00209         return is;
00210     }
00211
00212     friend std::ostream& operator«(std::ostream& os, const studentas& s)
00213     {
00214         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;
00215         //std::cout « "Suveike isvesties operatorius\n";
00216         return os;
00217 
00218     }
00219 
00220 
00221     void clearEverything()
00222     {
00223         this->vardas.clear();
00224         this->pavarde.clear();
00225         this->ND.clear();
00226         this->EGZ = 0;
00227         this->GalutinisV = 0;
00228         this->GalutinisM = 0;
00229 
00230     }
00231 };
00232 
00233 #endif
```

Index