SOURCE CODE TUGAS BESAR I

IF2210/Pemrograman Berorientasi Objek

Kalkulator Sakti

Dipersiapkan oleh:

Kelompok /\* \*/

**Luqman A. Siswanto 13513024**

**Muhammad Aodyra Khaidir 13513063**

**Wiwit Rifa’i 13513073**

**Ahmad Darmawan 13513096**

Mentor :

**Sonny Lazuardi Hermawan 13511030**

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung

Jl. Ganesha 10, Bandung 40132

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Sekolah Teknik Elektro dan Informatika ITB** | Nomor Dokumen | | Halaman |
| *IF2210-TB-01* | | *11* |
| *Revisi* | *01* | *19 Maret 2015* |

**DAFTAR ISI**

1. **Package Command**
   1. ClassController.h

|  |
| --- |
| #ifndef \_\_ClassController\_H\_  #define \_\_ClassController\_H\_  #include <string>  #include <cstdio>  #include <cassert>  #include <cstdlib>  #include <conio.h>  #include "Reader.h"  #include "Saver.h"  #include "Logger.h"  #include "../Equation/Equation.h"  #include "../Equation/Expression.h"  #include "../Extension/Extension.h"  /\*\*  \* @class Saver  \* @author Luqman A. Siswanto (13513024)  \* @version 1.0  \*  \* @section Description  \* @brief Kelas ClassController bertugas untuk mengatur kehidupan dan kematian kelas-kelas lain.  \*\*/  class ClassController **{**  public**:**  /\*\*  \* @brief Konstruktor kelas ClassController.  \*  \*\*/  ClassController**();**  /\*\*  \* @brief Copy Constructor kelas ClassController dengan parameter.  \* @param ClassController yang akan di-copy.  \*\*/  ClassController**(**const ClassController**&);**  /\*\*  \* @brief Operator assignment kelas ClassController.  \*\*/  ClassController**&** **operator=(**const ClassController**&);**  /\*\*  \* @brief Destruktor kelas ClassController.  \*\*/  **~**ClassController**();**  /\*\*  \* @brief Mengeksekusi ekspresi yang dimasukkan oleh user.  \* @param string - masukan string ekspresi yang akan dieksekusi.  \*\*/  void ExecuteExpression**(**string**&);**  /\*\*  \* @brief Mengeksekusi command yang dimasukkan oleh user.  \* @param string - masukan string command yang akan dieksekusi.  \*\*/  void ExecuteCommand**(**string**);**  /\*\*  \* @brief Mengembalikan perintah yang telah dimasukkan user.  \* @param int - banyak perintah.  \*\*/  void Undo**(**int**);**  /\*\*  \* @brief Mengembalikan perintah yang telah di-undo user.  \* @param int - banyak perintah.  \*\*/  void Redo**(**int**);**    /\*\*  \* @brief Mengubah mode ekspresi (prefiks - infiks - postfiks)  \* @param int - nomor setting (ada di kelas Extension).  \*\*/  void SetExpressionMode**(**int**);**  /\*\*  \* @brief Mengubah mode equation (bilangan - logika)  \* @param int - nomor setting (ada di kelas Extension).  \*\*/  void SetEquationMode**(**int**);**  /\*\*  \* @brief Mengubah mode number (romawi - arab)  \* @param int - nomor setting (ada di kelas Extension).  \*\*/  void SetNumberMode**(**int**);**    /\*\*  \* @brief Mengembalikan setting ke mode default.  \*\*/  void ResetSetting**();**  /\*\*  \* @brief Menampilkan n perintah terakhir yang dimasukkan user.  \* @param n - int. Banyak perintah.  \*\*/  void ShowMem**(**int**);**  /\*\*  \* @brief Menampilkan semua perintah yang dimasukkan user.  \*\*/  void ShowMemAll**();**  /\*\*  \* @brief Menampilkan 'help' yang berisi daftar command yang berlaku dalam program.  \*\*/  void Help**();**  /\*\*  \* @brief Menampilkan mode setting terkini  \*\*/  void ViewSetting**();**    private**:**  int \_expressionMode**;** // mode ekspresi saat ini  int \_equationMode**;** // mode equation saat ini  int \_numberMode**;** // mode bilangan saat ini  Reader**\*** \_reader**;** // pembaca  Saver**\*** \_saver**;** // penyimpan file  Logger**\*** \_logger**;** // penyimpan perintah yang pernah dimasukkan user  **};**  #endif // \_\_ClassController\_H\_ |

* 1. ClassController.cpp

|  |
| --- |
| #include "ClassController.h"  ClassController**::**ClassController**()** **{**  \_expressionMode **=** Extension**::**defaultExpressionMode**;**  \_equationMode **=** Extension**::**defaultEquationMode**;**  \_numberMode **=** Extension**::**defaultNumberMode**;**  \_reader **=** **new** Reader**();**  \_saver **=** **new** Saver**();**  \_logger **=** **new** Logger**();**    printf**(**"Type 'help' to show documentation.\n\n"**);**  string buffer**;**  int id **=** 0**,** n**;**  **do** **{**  id**++;**  printf**(**"> "**);**  buffer **=** \_reader**->**Read**();**  **if(**\_reader**->**IsEquation**())** **{**  **try** **{**  ExecuteExpression**(**buffer**);**  \_logger**->**AddEquation**(**Log**(**id**,** buffer**));**  **}** **catch** **(**EquationException**&** e**)** **{**  **if(**e**.**getID**()** **!=** EquationException**::**EmptyEquation**)**  printf**(**"\nException : %s\n"**,** e**.**getMessage**().**c\_str**());**  **}**  **}** **else** **{**  ExecuteCommand**(**buffer**);**  \_logger**->**AddCommand**(**Log**(**id**,** buffer**));**  **}**  **}** **while(**buffer **!=** "exit"**);**  printf**(**"Bye bye\n"**);**  **}**  ClassController**::**ClassController**(**const ClassController**&** man**)** **{**  **this->**\_expressionMode **=** man**.**\_expressionMode**;**  **this->**\_equationMode **=** man**.**\_equationMode**;**  **this->**\_numberMode **=** man**.**\_numberMode**;**  **this->**\_reader **=** **new** Reader**(\***man**.**\_reader**);**  **this->**\_saver **=** **new** Saver**(\***man**.**\_saver**);**  **this->**\_logger **=** **new** Logger**(\***man**.**\_logger**);**  **}**  ClassController**&** ClassController**::operator=(**const ClassController**&** man**)** **{**  **delete** \_reader**;**  **delete** \_saver**;**  **delete** \_logger**;**  **this->**\_expressionMode **=** man**.**\_expressionMode**;**  **this->**\_equationMode **=** man**.**\_equationMode**;**  **this->**\_numberMode **=** man**.**\_numberMode**;**  **this->**\_reader **=** **new** Reader**(\***man**.**\_reader**);**  **this->**\_saver **=** **new** Saver**(\***man**.**\_saver**);**  **this->**\_logger **=** **new** Logger**(\***man**.**\_logger**);**  **return** **\*this;**  **}**  ClassController**::~**ClassController**()** **{**  **delete** \_reader**;**  **delete** \_saver**;**  **delete** \_logger**;**  **}**  void ClassController**::**ExecuteExpression**(**string**&** buffer**)** **{**  Equation solver**(**buffer**,** \_expressionMode**,** \_equationMode**,** \_numberMode**);**  **if(**\_equationMode **==** Extension**::**NumberMode**)** **{**  solver**.**SolveMathematical**();**  **}** **else** **if(**\_equationMode **==** Extension**::**LogicMode**)** **{**  solver**.**SolveLogical**();**  **}** **else** **{**  assert**(false);**  **}**  buffer **+=** " = " **+** solver**.**GetResult**();**  printf**(**"%s\n"**,** solver**.**GetResult**().**c\_str**());**  **}**  void ClassController**::**ExecuteCommand**(**string buffer**)** **{**  stringstream ss**(**buffer**);**  string temp**,** word**;**  int n**;**  ss **>>** word**;**  **if(**word **==** "save"**)** **{**  printf**(**"Nama file : "**);**  getline**(**cin**,** word**);**  \_saver**->**SetFileName**(**word**);**  \_saver**->**SetLogMemory**(**Logger**(\***\_logger**));**  \_saver**->**ConvertToFile**();**  **}** **else** **if(**word **==** "redo"**)** **{**  ss **>>** n**;**  Redo**(**n**);**  **}** **else** **if(**word **==** "undo"**)** **{**  ss **>>** n**;**  Undo**(**n**);**  **}** **else** **if(**word **==** "set"**)** **{**  ss **>>** word**;**  **if(**word **==** "infiks"**)** SetExpressionMode**(**Extension**::**Infix**);**  **else** **if(**word **==** "prefiks"**)** SetExpressionMode**(**Extension**::**Prefix**);**  **else** **if(**word **==** "postfiks"**)** SetExpressionMode**(**Extension**::**Postfix**);**    **else** **if(**word **==** "bilangan"**)** SetEquationMode**(**Extension**::**NumberMode**);**  **else** **if(**word **==** "logika"**)** SetEquationMode**(**Extension**::**LogicMode**);**    **else** **if(**word **==** "romawi"**)** SetNumberMode**(**Extension**::**RomawiMode**);**  **else** **if(**word **==** "arab"**)** SetNumberMode**(**Extension**::**ArabMode**);**    **else** printf**(**"Setting gagal.\n"**);**  **}** **else** **if(**word **==** "reset"**)** **{**  ResetSetting**();**  **}** **else** **if(**word **==** "show"**)** **{**  ss **>>** temp **>>** word**;**  n **=** atoi**(**word**.**c\_str**());**  **if(**word **==** "all"**)** ShowMemAll**();**  **else** **if(**n **>=** 0**)** ShowMem**(**n**);**  **else** **{**  printf**(**"Pembacaan histori gagal.\n"**);**  **}**  **}** **else** **if(**word **==** "help"**)** **{**  Help**();**  **}** **else** **if(**word **==** "view"**)** **{**  ViewSetting**();**  **}** **else** **if(**word **==** "exit"**)** **{**  assert**(true);**  **}** **else** **{**  printf**(**"Command tidak ada\n"**);**  **}**  **}**  void ClassController**::**Redo**(**int n**)** **{**  int redo **=** \_logger**->**RedoEquation**(**n**);**  **if(**redo **==** 0**)** **{**  printf**(**"Redo tidak dapat dilakukan.\n"**);**  **}** **else** **{**  printf**(**"%d ekspresi berhasil di Redo.\n"**,** redo**);**  **}**  **}**  void ClassController**::**Undo**(**int n**)** **{**  int undo **=** \_logger**->**UndoEquation**(**n**);**  **if(**undo **==** 0**)** **{**  printf**(**"Undo tidak dapat dilakukan.\n"**);**  **}** **else** **{**  printf**(**"%d ekspresi berhasil di Undo.\n"**,** undo**);**  **}**  **}**  void ClassController**::**SetExpressionMode**(**int expressionMode**)** **{**  \_expressionMode **=** expressionMode**;**  printf**(**"Set expression mode success.\n"**);**  **}**  void ClassController**::**SetEquationMode**(**int equationMode**)** **{**  \_equationMode **=** equationMode**;**  printf**(**"Set equation mode success.\n"**);**  **}**  void ClassController**::**SetNumberMode**(**int numberMode**)** **{**  \_numberMode **=** numberMode**;**  printf**(**"Set number mode success.\n"**);**  **}**  void ClassController**::**ResetSetting**()** **{**  \_expressionMode **=** Extension**::**defaultExpressionMode**;**  \_equationMode **=** Extension**::**defaultEquationMode**;**  \_numberMode **=** Extension**::**defaultNumberMode**;**  printf**(**"Reset setting to default success.\n"**);**  **}**  void ClassController**::**ShowMem**(**int n**)** **{**  \_logger**->**ShowMem**(**n**);**  **}**  void ClassController**::**ShowMemAll**()** **{**  \_logger**->**ShowMemAll**();**  **}**  void ClassController**::**Help**()** **{**  // show what command we are offering here  // list command yang ditawarkan ada di doc RencanaKelasImplementasi, udah gw update  printf**(**"\n"**);**  printf**(**" Berikut adalah command yang disediakan di program ini.\n\n"**);**  printf**(**" save : Untuk menyimpan operasi yang pernah dilakukan dalam file"**);**  printf**(**" eksternal. File eksternal akan diminta selanjutnya\n\n"**);**  printf**(**" undo <n> : Menghapus n buah ekspresi terakhir, <n> adalah bilangan cacah\n\n"**);**  printf**(**" redo <n> : Mengulang n perintah terakhir\n\n"**);**  printf**(**" set \n"**);**  printf**(**" prefiks : Mengubah setting ekspresi menjadi prefiks\n"**);**  printf**(**" infiks : Mengubah setting ekspresi menjadi infiks\n"**);**  printf**(**" postfiks : Mengubah setting ekspresi menjadi postfiks\n"**);**  printf**(**" bilangan : Mengubah setting equation menjadi bilangan\n"**);**  printf**(**" logika : Mengubah setting equation menjadi logika\n"**);**  printf**(**" arab : Mengubah setting bilangan menjadi bilangan arab\n"**);**  printf**(**" romawi : Mengubah setting bilangan menjadi romawi\n\n"**);**  printf**(**" reset : Mengembalikan setting ke mode default\n\n"**);**    printf**(**" Klik apapun KECUALI [Enter] untuk melanjutkan "**);** **while(!**kbhit**());**  system**(**"cls"**);**  printf**(**"\n\n"**);**    printf**(**" show mem <n> : Menampilkan ekspresi atau perintah terakhir sebanyak n"**);** printf**(**" <n> adalah bilangan cacah\n\n"**);**  printf**(**" show mem all : Menampilkan seluruh ekspresi atau perintah yang"**);**  printf**(**" pernah dilakukan\n\n"**);**  printf**(**" view : Menampilkan mode setting saat ini\n\n"**);**  printf**(**" exit : Keluar dari command\n\n"**);**    printf**(**" DEFAULT MODE :\n"**);**  printf**(**" Expression mode : infiks\n"**);**  printf**(**" Equation mode : bilangan\n"**);**  printf**(**" Number mode : arab\n\n"**);**  printf**(**" Setting berada dalam default mode ketika awal run program\n\n"**);**  **}**  void ClassController**::**ViewSetting**()** **{**  // belum selesai  printf**(**"Setting expression : "**);**  **if(**\_expressionMode **==** Extension**::**Prefix**)** **{**  printf**(**"prefiks"**);**  **}** **else** **if(**\_expressionMode **==** Extension**::**Infix**)** **{**  printf**(**"infiks"**);**  **}** **else** **if(**\_expressionMode **==** Extension**::**Postfix**)** **{**  printf**(**"postfiks"**);**  **}** **else** **{**  assert**(false);**  **}**  printf**(**"\n"**);**    printf**(**"Setting equation : "**);**  **if(**\_equationMode **==** Extension**::**NumberMode**)** **{**  printf**(**"bilangan"**);**  **}** **else** **if(**\_equationMode **==** Extension**::**LogicMode**)** **{**  printf**(**"logika"**);**  **}** **else** **{**  assert**(false);**  **}**  printf**(**"\n"**);**    printf**(**"Setting number : "**);**  **if(**\_numberMode **==** Extension**::**ArabMode**)** **{**  printf**(**"arab"**);**  **}** **else** **if(**\_numberMode **==** Extension**::**RomawiMode**)** **{**  printf**(**"romawi"**);**  **}** **else** **{**  assert**(false);**  **}**  printf**(**"\n"**);**  **}** |

* 1. Log.h

|  |
| --- |
| #ifndef \_\_LOG\_H\_  #define \_\_LOG\_H\_  #include <string>  **using** **namespace** std**;**  /\*\*  \* @class Log  \* @author Luqman A. Siswanto (13513024)  \* @version 1.0  \*  \* @section Description  \* @brief Kelas Log adalah abstract data type untuk log command  \*\*/  class Log **{**  public**:**  /\*\*  \* @brief Konstruktor kelas Log.  \*\*/  Log**();**  /\*\*  \* @brief Konstruktor kelas Log dengan parameter.  \* @param int - ID.  \* @param string - command pada log yang akan disimpan  \*\*/  Log**(**int**,** string**);**  /\*\*  \* @brief Copy Constructor kelas Log.  \* @param Log yang akan di-copy.  \*\*/  Log**(**const Log**&);**  /\*\*  \* @brief Operator assignment kelas Log.  \* @param Log yang akan di-copy.  \*\*/  Log**&** **operator=(**const Log**&);**  /\*\*  \* @brief Destruktor kelas Log.  \*\*/  **~**Log**();**    /\*\*  \* @brief Getter untuk ID.  \* @return int - ID pada log.  \*\*/  int GetID**();**  /\*\*  \* @brief Getter untuk kalimat log.  \* @return string - sentence.  \*\*/  string GetSentence**();**  /\*\*  \* @brief Setter untuk ID.  \* @param int - ID log.  \*\*/  void SetID**(**int**);**  /\*\*  \* @brief Setter untuk sentence.  \* @param string - sentence.  \*\*/  void SetSentence**(**string**);**  private**:**  int \_id**;** // id log, dipastikan tiap log memiliki ID yang unik dan terurut menaik berdasarkan eksekusi  string \_sentence**;** // kalimat yang berisi perintah  **};**  #endif |

* 1. Log.cpp

|  |
| --- |
| #include "Log.h"  Log**::**Log**()** **{**  \_id **=** 0**;**  \_sentence **=** ""**;**  **}**  Log**::**Log**(**int id**,** string sentence**)** **{**  \_id **=** id**;**  \_sentence **=** sentence**;**  **}**  Log**::**Log**(**const Log**&** log**)** **{**  \_id **=** log**.**\_id**;**  \_sentence **=** log**.**\_sentence**;**  **}**  Log**&** Log**::operator=(**const Log**&** log**)** **{**  \_id **=** log**.**\_id**;**  \_sentence **=** log**.**\_sentence**;**  **return** **\*this;**  **}**  Log**::~**Log**()** **{**  \_sentence**.**clear**();**  **}**  int Log**::**GetID**()** **{**  **return** \_id**;**  **}**  string Log**::**GetSentence**()** **{**  **return** \_sentence**;**  **}**  void Log**::**SetID**(**int id**)** **{**  \_id **=** id**;**  **}**  void Log**::**SetSentence**(**string sentence**)** **{**  \_sentence **=** sentence**;**  **}** |

* 1. Logger.h

|  |
| --- |
| #ifndef \_\_LOGGER\_H\_  #define \_\_LOGGER\_H\_  #include "Log.h"  #include <vector>  #include <cstdio>  #include <cmath>  #include <cassert>  **using** **namespace** std**;**  /\*\*  \* @class Logger  \* @author Luqman A. Siswanto (13513024)  \* @version 1.0  \*  \* @section Description  \* @brief Kelas Log adalah abstract data type untuk log command  \*  \* @section Rule  \* @brief Semua dalam vector commands dijamin valid (tidak ada operasi undo/redo)  \* @brief Seluruh isi vector equations belum tentu valid karena bisa jadi hasil undo  \*\*/  class Logger **{**  public**:**  /\*\*  \* @brief Konstruktor kelas logger.  \*\*/  Logger**();**  /\*\*  \* @brief Konstruktor kelas logger dengan parameter.  \* @param vector<Log> - commands  \* @param vector<Log> - equations  \*\*/  Logger**(**vector**<**Log**>,** vector**<**Log**>);**  /\*\*  \* @brief Konstruktor kelas logger dengan parameter.  \* @param vector<Log> - commands  \* @param vector<Log> - equations  \* @param int - ukuran equations yang valid  \*\*/  Logger**(**vector**<**Log**>,** vector**<**Log**>,** int**);**  /\*\*  \* @brief Copy constructor kelas logger.  \* @param Logger - logger yang akan di-copy  \*\*/  Logger**(**const Logger**&);**  /\*\*  \* @brief Operator assignment kelas Logger.  \* @param Logger - logger yang akan di-copy  \*\*/  Logger**&** **operator=(**const Logger**&);**  /\*\*  \* @brief Destruktor kelas Logger.  \*\*/  **~**Logger**();**    /\*\*  \* @brief Getter Log Commands dengan indeks tertentu.  \* @param int - indeks  \* @return Log - Log Command dengan indeks tertentu  \*\*/  Log GetCommand**(**int**);**  /\*\*  \* @brief Getter Log Equations dengan indeks tertentu.  \* @param int - indeks  \* @return Log - Log Equation dengan indeks tertentu  \*\*/  Log GetEquation**(**int**);**  /\*\*  \* @brief Getter ukuran commands yang valid.  \* @return int - commands size  \*\*/  int GetSizeCommands**();**  /\*\*  \* @brief Getter ukuran equations yang valid.  \* @return int - equations size  \*\*/  int GetSizeEquations**();**    /\*\*  \* @brief Menghilangkan seluruh commands dan equations di memori.  \*\*/  void Clear**();**  /\*\*  \* @brief Menghilangkan commands di memori.  \*\*/  void ClearCommands**();**  /\*\*  \* @brief Menghilangkan commands di equations.  \*\*/  void ClearEquations**();**    /\*\*  \* @brief Menambahkan log pada list of commands.  \* @param Log - yang akan ditambah  \*\*/  void AddCommand**(**Log**);**  /\*\*  \* @brief Menambahkan log pada list of equations.  \* @param Log - yang akan ditambah  \*\*/  void AddEquation**(**Log**);**    /\*\*  \* @brief Membatalkan equation yang telah ditambah  \* @param int - berapa item undo yang diminta  \* @return int - berapa item undo yang berhasil dilakukan  \*\*/  int UndoEquation**(**int**);**  /\*\*  \* @brief Melakukan kembali equation yang telah di-undo  \* @param int - berapa item redo yang diminta  \* @return int - berapa item redo yang berhasil dilakukan  \*\*/  int RedoEquation**(**int**);**  /\*\*  \* @brief Menampilkan memori kembali ke layar  \* @param int - berapa item yang akan ditampilkan ke layar  \*\*/  void ShowMem**(**int**);**  /\*\*  \* @brief Menampilkan semua memori kembali ke layar  \*\*/  void ShowMemAll**();**    private**:**  vector**<**Log**>** \_commands**;** // ID command dan equation dijamin unique  vector**<**Log**>** \_equations**;**  int \_sizeEquations**;** // menandakan banyak Log equation yang valid  // setelah operasi redo dan undo  **};**  #endif |

* 1. Logger.cpp

|  |
| --- |
| #include "Logger.h"  Logger**::**Logger**()** **{**  \_commands**.**clear**();**  \_equations**.**clear**();**  \_sizeEquations **=** 0**;**  **}**  Logger**::**Logger**(**vector**<**Log**>** commands**,** vector**<**Log**>** equations**)** **{**  \_commands **=** commands**;**  \_equations **=** equations**;**  \_sizeEquations **=** equations**.**size**();**  **}**  Logger**::**Logger**(**vector**<**Log**>** commands**,** vector**<**Log**>** equations**,** int sizeEquations**)** **{**  \_commands **=** commands**;**  \_equations **=** equations**;**  \_sizeEquations **=** sizeEquations**;**  **}**  Logger**::**Logger**(**const Logger**&** logger**)** **{**  \_commands **=** logger**.**\_commands**;**  \_equations **=** logger**.**\_equations**;**  \_sizeEquations **=** logger**.**\_sizeEquations**;**  **}**  Logger**&** Logger**::operator=(**const Logger**&** logger**)** **{**  \_commands **=** logger**.**\_commands**;**  \_equations **=** logger**.**\_equations**;**  \_sizeEquations **=** logger**.**\_sizeEquations**;**  **return** **\*this;**  **}**  Logger**::~**Logger**()** **{**  \_commands**.**clear**();**  \_equations**.**clear**();**  **}**  Log Logger**::**GetCommand**(**int i**)** **{**  assert**(**0 **<=** i **&&** i **<** \_commands**.**size**());**  **return** \_commands**[**i**];**  **}**  Log Logger**::**GetEquation**(**int i**)** **{**  assert**(**0 **<=** i **&&** i **<** \_sizeEquations**);**  **return** \_equations**[**i**];**  **}**  int Logger**::**GetSizeCommands**()** **{**  **return** \_commands**.**size**();**  **}**  int Logger**::**GetSizeEquations**()** **{**  **return** \_sizeEquations**;**  **}**  void Logger**::**Clear**()** **{**  \_commands**.**clear**();**  \_equations**.**clear**();**  \_sizeEquations **=** 0**;**  **}**  void Logger**::**ClearCommands**()** **{**  \_commands**.**clear**();**  **}**  void Logger**::**ClearEquations**()** **{**  \_equations**.**clear**();**  \_sizeEquations **=** 0**;**  **}**  void Logger**::**AddCommand**(**Log log**)** **{**  \_commands**.**push\_back**(**log**);**  **}**  void Logger**::**AddEquation**(**Log log**)** **{**  assert**(**\_sizeEquations **<=** \_equations**.**size**());**  **if(**\_sizeEquations **==** \_equations**.**size**())** **{**  \_equations**.**push\_back**(**log**);**  **}** **else** **{**  \_equations**[**\_sizeEquations**]** **=** log**;**  **}**  \_sizeEquations**++;**  **}**  int Logger**::**UndoEquation**(**int n**)** **{**  int tobe **=** max**(**\_sizeEquations **-** n**,** 0**);**  int change **=** \_sizeEquations **-** tobe**;**  \_sizeEquations **=** tobe**;**  **return** change**;**  **}**  int Logger**::**RedoEquation**(**int n**)** **{**  int tobe **=** min**(**\_sizeEquations **+** n**,** **(**int**)** \_equations**.**size**());**  int change **=** tobe **-** \_sizeEquations**;**  \_sizeEquations **=** tobe**;**  **return** change**;**  **}**  void Logger**::**ShowMem**(**int n**)** **{**  int ptEq **=** \_sizeEquations **-** 1**;**  int ptCom **=** **(**int**)** \_commands**.**size**()** **-** 1**;**  int index **=** 0**;**  **while(**n **>** 0 **&&** **(**ptEq **>=** 0 **||** ptCom **>=** 0**))** **{**  **++**index**;**  **if(**ptEq **>=** 0 **&&** ptCom **>=** 0**)** **{**  **if(**\_equations**[**ptEq**].**GetID**()** **>** \_commands**[**ptCom**].**GetID**())** **{**  printf**(**"Perintah terakhir (equation) %2d : %s\n"**,** index**,**  \_equations**[**ptEq**].**GetSentence**().**c\_str**());**  ptEq**--;**  **}** **else** **{**  printf**(**"Perintah terakhir (command) %3d : %s\n"**,** index**,**  \_commands**[**ptCom**].**GetSentence**().**c\_str**());**  ptCom**--;**  **}**  **}** **else** **if(**ptEq **>=** 0**)** **{**  printf**(**"Perintah terakhir (equation) %2d : %s\n"**,** index**,**  \_equations**[**ptEq**].**GetSentence**().**c\_str**());**  ptEq**--;**  **}** **else** **{**  assert**(**ptCom **>=** 0**);**  printf**(**"Perintah terakhir (command) %3d : %s\n"**,** index**,**  \_commands**[**ptCom**].**GetSentence**().**c\_str**());**  ptCom**--;**  **}**  n**--;**  **}**  **}**  void Logger**::**ShowMemAll**()** **{**  int ptEq **=** \_sizeEquations **-** 1**;**  int ptCom **=** **(**int**)** \_commands**.**size**()** **-** 1**;**  int index **=** 0**;**  **while(**ptEq **>=** 0 **||** ptCom **>=** 0**)** **{**  **++**index**;**  **if(**ptEq **>=** 0 **&&** ptCom **>=** 0**)** **{**  **if(**\_equations**[**ptEq**].**GetID**()** **>** \_commands**[**ptCom**].**GetID**())** **{**  printf**(**"Perintah terakhir (equation) %2d : %s\n"**,** index**,**  \_equations**[**ptEq**].**GetSentence**().**c\_str**());**  ptEq**--;**  **}** **else** **{**  printf**(**"Perintah terakhir (command) %3d : %s\n"**,** index**,**  \_commands**[**ptCom**].**GetSentence**().**c\_str**());**  ptCom**--;**  **}**  **}** **else** **if(**ptEq **>=** 0**)** **{**  printf**(**"Perintah terakhir (equation) %2d : %s\n"**,** index**,**  \_equations**[**ptEq**].**GetSentence**().**c\_str**());**  ptEq**--;**  **}** **else** **{**  assert**(**ptCom **>=** 0**);**  printf**(**"Perintah terakhir (command) %3d : %s\n"**,** index**,**  \_commands**[**ptCom**].**GetSentence**().**c\_str**());**  ptCom**--;**  **}**  **}**  **if(**index **==** 0**)** **{**  printf**(**"Tidak ada memori yang dapat ditampilkan.\n"**);**  **}**  **}** |

* 1. Reader.h

|  |
| --- |
| #ifndef \_\_READER\_H\_  #define \_\_READER\_H\_  #include <iostream>  #include <sstream>  #include <string>  #include <vector>  **using** **namespace** std**;**  /\*\*  \* @class Reader  \* @author Luqman A. Siswanto (13513024)  \* @version 1.0  \*  \* @section Description  \* @brief Kelas Reader bertugas menerima input dari user kemudian mengkategorikan input tersebut termasuk command atau ekspresi  \*  \*\*/  class Reader **{**  public**:**  /\*\*  \* @brief Konstruktor kelas Reader.  \*\*/  Reader**();**  /\*\*  \* @brief Copy constructor kelas logger.  \* @param Reader : yang akan di-copy  \*\*/  Reader**(**const Reader**&);**  /\*\*  \* @brief Copy constructor kelas logger.  \* @param Reader : yang akan di-copy  \*\*/  Reader**&** **operator=(**const Reader**&);**  /\*\*  \* @brief Destruktor kelas logger.  \*\*/  **~**Reader**();**    /\*\*  \* @brief Membaca perintah dari user sekaligus meng-update data member isEquation.  \* @return string - string yang berhasil dibaca  \*\*/  string Read**();**  /\*\*  \* @brief Mengambil kalimat dari input yang telah dibaca.  \* @return string - string yang berhasil dibaca  \*\*/  string GetSentence**();**  /\*\*  \* @brief Mengembalikan predikat apakah sebuah string merupakan equation  \* @return bool - jika true, maka string adalah equation  \*\*/  bool IsEquation**();**    private**:**  /\*\*  \* @brief Melakukan identifikasi apakah string merupakan equation  \*\*/  void \_UpdateIsEquation**();**  string \_sentence**;** // kalimat yang tepat dibaca dari user  bool \_isEquation**;** // true jika dia equation, false jika perintah  static const string \_commands**[];** // list dari command yang diterima oleh program  **};**  #endif |

* 1. Reader.cpp

|  |
| --- |
| #include "Reader.h"  const string Reader**::**\_commands**[]** **=** **{**"save"**,**  "redo"**,**  "undo"**,**  "set"**,**  "reset"**,**  "show"**,**  "view"**,**  "help"**,**  "exit"**};**  Reader**::**Reader**()** **{**  \_sentence **=** "Default Sentence"**;**  \_isEquation **=** **false;**  **}**  Reader**::**Reader**(**const Reader**&** r**)** **{**  \_sentence **=** r**.**\_sentence**;**  \_isEquation **=** r**.**\_isEquation**;**  **}**  Reader**&** Reader**::operator=(**const Reader**&** r**)** **{**  \_sentence **=** r**.**\_sentence**;**  \_isEquation **=** r**.**\_isEquation**;**  **return** **\*this;**  **}**  Reader**::~**Reader**()** **{**  \_sentence**.**clear**();**  **}**  string Reader**::**Read**()** **{**  getline**(**cin**,** \_sentence**);**  \_UpdateIsEquation**();**  **return** \_sentence**;**  **}**  string Reader**::**GetSentence**()** **{**  **return** \_sentence**;**  **}**  bool Reader**::**IsEquation**()** **{**  **return** \_isEquation**;**  **}**  void Reader**::**\_UpdateIsEquation**()** **{**  stringstream ss**(**\_sentence**);**  string firstWord**;**  ss **>>** firstWord**;**  int size **=** **sizeof(**\_commands**)** **/** **sizeof(\***\_commands**);**  **for(**int i **=** 0**;** i **<** size**;** i**++)** **{**  **if(**firstWord **==** \_commands**[**i**])** **{**  \_isEquation **=** **false;**  **return;**  **}**  **}**  \_isEquation **=** **true;**  **}** |

* 1. Saver.h

|  |
| --- |
| /\*\*  \* @file Saver.h  \*\*/  #ifndef Saver\_H  #define Saver\_H  #include <ctime>  #include <cstdio>  #include <vector>  #include <cstdio>  #include <cmath>  #include <cassert>  #include "Logger.h"  **using** **namespace** std**;**  /\*\*  \* @class Saver  \* @author Ahmad Darmawan (13513096)  \* @version 1.0  \*  \* @brief Kelas Saver bertugas untuk mengelola penyimpanan instruksi ke dalam file eksternal.  \*\*/  class Saver  **{**  public**:**  /\*\*  \* @brief Konstruktor kelas saver.  \*  \*\*/  Saver**();**  /\*\*  \* @brief Konstruktor kelas saver dengan parameter.  \* [I.S] string S dan Logger L terdefinisi.  \* [F.S] Elemen Saver telah terassign dengan string S dan Logger L.  \* @param S - String yang akan menjadi nama file.  \* @param L - Logger berisi instruksi (ekspresi dan perintah) yang disimpan.  \*\*/  Saver**(**string S**,** Logger L**);**  /\*\*  \* @brief Copy Konstruktor kelas saver.  \* @param S - Objek Saver.  \*\*/  Saver**(**const Saver**&** S**);**  /\*\*  \* @brief Assignment kelas saver.  \* @param S - Objek Saver.  \* @return S' - Objek Saver yang telah diassign dari S.  \*\*/  Saver**&** **operator=(**const Saver**&** S**);**  /\*\*  \* @brief Destruktor kelas saver.  \*\*/  **~**Saver**();**  /\*\*  \* @brief Getter untuk mendapatkan nama file.  \* @return S - string nama file.  \*\*/  string GetFileName**();**  /\*\*  \* @brief Getter untuk mendapatkan variabel logger.  \* @return L - variabel logger.  \*\*/  Logger GetLogMemory**();**  /\*\*  \* @brief Setter untuk mengubah nama file.  \* @brief [I.S] String terdefinisi yang dapat dijadikan nama file.  \* @brief [F.S] \_filename terassign dengan parameter string.  \* @param S - nama file.  \*\*/  void SetFileName**(**string**);**  /\*\*  \* @brief Setter untuk mengubah Logger.  \* @brief [I.S] Logger terdefinisi.  \* @brief [F.S] \_logMemory terassign dengan parameter Logger.  \* @param L - Logger.  \*\*/  void SetLogMemory**(**Logger**);**  /\*\*  \* @brief Prosedur program membuat file eksternal.  \* @brief [I.S] String \_filename dan Logger \_logMemory terdefinisi.  \* @brief [F.S] Tercipta file eksternal bernama \_filename dengan masukan dari \_logMemory.  \*\*/  void ConvertToFile**();**    private**:**  string \_filename**;** // \_filename adalah nama file.  Logger \_logMemory**;** // \_logMemory adalah variabel logger.  FILE**\*** out**;** // out adalah variabel output file stream.  **};**  #endif |

* 1. Saver.cpp

|  |
| --- |
| #include "Saver.h"  **using** **namespace** std**;**  /\*\*  \* @brief Konstruktor kelas saver.  \*  \*\*/  Saver**::**Saver**()** **{**  \_filename **=** ""**;**  **}**  /\*\*  \* @brief Konstruktor kelas saver dengan parameter.  \* @param S - String yang akan menjadi nama file.  \* @param L - Logger berisi instruksi (ekspresi dan perintah) yang disimpan.  \*\*/  Saver**::**Saver**(**string S**,** Logger L**)** **:** \_logMemory**(**L**)** **{**  \_filename **=** S**;**  **}**  /\*\*  \* @brief Copy Konstruktor kelas saver.  \* @param S - Objek Saver.  \*\*/  Saver**::**Saver**(**const Saver**&** S**)** **{**  **this->**\_filename **=** S**.**\_filename**;**  **this->**\_logMemory **=** S**.**\_logMemory**;**  **}**  /\*\*  \* @brief Assignment kelas saver.  \* @param S - Objek Saver.  \* @return S' - Objek Saver yang telah diassign dari S.  \*\*/  Saver**&** Saver**::operator=(**const Saver**&** S**)** **{**  Saver temp**;**  temp**.**\_filename **=** S**.**\_filename**;**  temp**.**\_logMemory **=** S**.**\_logMemory**;**  **}**  /\*\*  \* @brief Destruktor kelas saver.  \*\*/  Saver**::~**Saver**()** **{**  **}**  /\*\*  \* @brief Getter untuk mendapatkan nama file.  \* @return S - string nama file.  \*\*/  string Saver**::**GetFileName**()** **{**  **return** \_filename**;**  **}**  /\*\*  \* @brief Getter untuk mendapatkan variabel logger.  \* @return L - variabel logger.  \*\*/  Logger Saver**::**GetLogMemory**()** **{**  **return** \_logMemory**;**  **}**  /\*\*  \* @brief Setter untuk mengubah nama file.  \* @param S - nama file.  \*\*/  void Saver**::**SetFileName**(**string filename**)** **{**  \_filename **=** filename**;**  **}**  /\*\*  \* @brief Setter untuk mengubah Logger.  \* @param L - Logger.  \*\*/  void Saver**::**SetLogMemory**(**Logger logMemory**)** **{**  \_logMemory **=** logMemory**;**  **}**  /\*\*  \* @brief Prosedur program membuat file eksternal.  \*\*/  void Saver**::**ConvertToFile**()** **{**  out **=** fopen**(**\_filename**.**c\_str**(),**"w"**);**  /\* Pencatat Waktu \*/  time\_t ltime**;** /\* calendar time \*/  ltime**=**time**(NULL);** /\* get current cal time \*/  fprintf**(**out**,** "%s"**,**asctime**(** localtime**(&**ltime**)** **)** **);**  /\* Memasukkan Ekspresi dan Command \*/  int ptEq **=** \_logMemory**.**GetSizeEquations**()** **-** 1**;**  int ptCom **=** **(**int**)** \_logMemory**.**GetSizeCommands**()** **-** 1**;**  int index **=** 0**;**  **while(**ptEq **>=** 0 **||** ptCom **>=** 0**)** **{**  **++**index**;**  **if(**ptEq **>=** 0 **&&** ptCom **>=** 0**)** **{**  **if(**\_logMemory**.**GetEquation**(**ptEq**).**GetID**()** **>** \_logMemory**.**GetCommand**(**ptCom**).**GetID**())** **{**  fprintf**(**out**,** "> Perintah terakhir (equation) %2d : %s\n"**,** index**,**  \_logMemory**.**GetEquation**(**ptEq**).**GetSentence**().**c\_str**());**  ptEq**--;**  **}** **else** **{**  fprintf**(**out**,** "> Perintah terakhir (command) %3d : %s\n"**,** index**,**  \_logMemory**.**GetCommand**(**ptCom**).**GetSentence**().**c\_str**());**  ptCom**--;**  **}**  **}** **else** **if(**ptEq **>=** 0**)** **{**  fprintf**(**out**,** "> Perintah terakhir (equation) %2d : %s\n"**,** index**,**  \_logMemory**.**GetEquation**(**ptEq**).**GetSentence**().**c\_str**());**  ptEq**--;**  **}** **else** **{**  assert**(**ptCom **>=** 0**);**  fprintf**(**out**,** "> Perintah terakhir (command) %3d : %s\n"**,** index**,**  \_logMemory**.**GetCommand**(**ptCom**).**GetSentence**().**c\_str**());**  ptCom**--;**  **}**  **}**  fclose **(**out**);**  printf**(**"Penulisan di %s berhasil.\n"**,** \_filename**.**c\_str**());**  **}** |

1. Package Equation
   1. Expression.h

|  |
| --- |
| /\*\*  \* @file Expression.h  \*\*/  #ifndef \_\_EXPRESSION\_H  #define \_\_EXPRESSION\_H  #include "EquationException.h"  #include <string>  #include <cassert>  #include "../Stack/stack.h"  #include "../Token/Logic.h"  #include "../Token/Token.h"  #include "../Token/NumberArab.h"  #include "../Token/NumberRomawi.h"  #include "../Extension/Extension.h"  /\*\*  \* @class Expression  \* @author Wiwit Rifa'i (13513073)  \* @version 1.0  \*  \* @section Description  \* @brief Kelas Expression bertanggung jawab dalam mengolah ekspresi matematika atau logika seperti mengubah string menjadi ekspresi matematika dan mengubah bentuk infix atau postfix menjadi prefix agar bisa dihitung di class Equation.  \*\*/  class Expression **{**  public**:**  /\*\*  \* @brief Konstruktor kelas Expression.  \*\*/  Expression**();**  /\*\*  \* @brief Konstruktor kelas Expression dengan parameter.  \* @param string strExp - String yang merepresentasikan ekspresi matematika atau logika yang akan dikelola.  \* @param int modeExpression - kode integer yang menentukan apakah ekspresi dalam string tersebut dalam bentuk prefix, infix, ataupun postfix.  \* @param int modeEquation - kode integer yang menentukan jenis operasi yang digunakan yaitu operasi pada bilangan ataukah logika.  \* @param int modeNumber - kode integer yang menentukan jenis operan bilangan yang digunakan yaitu bilangan arab ataupun bilangan romawi.  \*\*/  Expression**(**const string**&** strExp **,** int modeExpression**,** int modeEquation**,** int modeNumber**);**  /\*\*  \* @brief Copy Constructor kelas Expression.  \*\*/  Expression**(**const Expression**&);**  /\*\*  \* @brief Destruktor kelas Expression.  \*\*/  virtual **~**Expression**();**  /\*\*  \* @brief Operator assignment kelas Expression.  \*\*/  Expression**&** **operator=(**const Expression**&);**  private**:**  /\*\*  \* @brief Mengubah string ekspresi dan mengubahnya menjadi token-token yang disimpan dalam stack.  \* @param string strEq - string ekspresi yang akan diubah menjadi token-token yang addressnya disimpan dalam stack.  \* @param int modeEquation - kode integer yang menentukan jenis operasi yang digunakan yaitu operasi pada bilangan ataukah logika.  \* @param int modeNumber - kode integer yang menentukan jenis operan bilangan yang digunakan yaitu bilangan arab ataupun bilangan romawi.  \*\*/  void \_stringToTokens**(**const string**&** **,** int **,** int **);**  /\*\*  \* @brief Mengubah keterurutan token-token pada stack \_stackToken dari ekspresi infix menjadi ekspresi prefix.  \*\*/  void \_inToPostfix**();**  /\*\*  \* @brief Mengubah keterurutan token-token pada stack \_stackToken dari ekspresi postfix menjadi ekspresi prefix.  \*\*/  void \_postToPrefix**();**  protected**:**  stack**<**Token **\*>** \_stackToken**;** // \_stackToken menyimpan address dari token-token yang merepresentasikan suatu ekspresi matematika  **};**  #endif |

* 1. Expression.cpp

|  |
| --- |
| /\*\*  \* @file Expression.cpp  \*\*/  #include "Expression.h"  Expression**::**Expression**()** **{**  **}**  Expression**::**Expression**(**const string**&** strExp **,** int modeExpression**,** int modeEquation**,** int modeNumber**)** **{**  \_stringToTokens**(**strExp**,** modeEquation**,** modeNumber**);**  **if(**modeExpression **==** Extension**::**Prefix**)** **{**  assert**(true);**  **}** **else** **if(**modeExpression **==** Extension**::**Infix**)** **{**  \_inToPostfix**();**  **}** **else** **if(**modeExpression **==** Extension**::**Postfix**){**  \_postToPrefix**();**  **}** **else** **{**  assert**(false);**  **}**  **}**  Expression**::**Expression**(**const Expression**&** E**)** **{**  \_stackToken **=** E**.**\_stackToken**;**  **}**  Expression**::~**Expression**()** **{**  Token **\*** t**;**  **while(!**\_stackToken**.**empty**())** **{**  t **=** \_stackToken**.**top**();**  \_stackToken**.**pop**();**  **delete** t**;**  **}**  **}**  Expression**&** Expression**::operator=(**const Expression**&** E**)** **{**  \_stackToken **=** E**.**\_stackToken**;**  **return** **\*this;**  **}**  void Expression**::**\_stringToTokens**(**const string**&** \_strEq**,** int modeEquation**,** int modeNumber**)** **{**  /\* mengubah string menjadi stack of token \*/  int pos **=** 0**,** len **=** 0**,** i **=** 0**;**  bool op **=** **false;**  **while(**i **<** \_strEq**.**length**())** **{**  **if(**\_strEq**[**i**]** **==** ' ' **||** \_strEq**[**i**]** **==** '\t' **||** \_strEq**[**i**]** **==** '\n' **||** \_strEq**[**i**]** **==** '~'**)** **{**  **if(**op**)** **{**  **if(**modeEquation **==** Extension**::**NumberMode**)** **{**  **if(**modeNumber **==** Extension**::**ArabMode**)**  \_stackToken**.**push**(new** NumberArab**(**\_strEq**.**substr**(**pos**,** len**)));**  **else** **if(**modeNumber **==** Extension**::**RomawiMode**)**  \_stackToken**.**push**(new** NumberRomawi**(**\_strEq**.**substr**(**pos**,** len**)));**  **else**  assert**(false);**  **}**  **else** **{**  **if(**modeEquation **==** Extension**::**LogicMode**)**  **{**  bool isLogic **=** **false;**  char LogicCheck**[]** **=** **{**'T'**,** 't'**,** 'F'**,** 'f'**,** '&'**,** '^'**,** '|'**};**  **for(**int i **=** 0**;** i**<**7 **&&** **!**isLogic**;** i**++)**  isLogic **=** **(**\_strEq**[**pos**]** **==** LogicCheck**[**i**]);**  **if(**isLogic**)**  \_stackToken**.**push**(new** Logic**(**\_strEq**.**substr**(**pos**,** len**)));**  **else**  **if(**modeNumber **==** Extension**::**ArabMode**)**  \_stackToken**.**push**(new** NumberArab**(**\_strEq**.**substr**(**pos**,** len**)));**  **else** **if(**modeNumber **==** Extension**::**RomawiMode**)**  \_stackToken**.**push**(new** NumberRomawi**(**\_strEq**.**substr**(**pos**,** len**)));**  **else**  assert**(false);**  **}**  **else**  assert**(false);**  **}**  op **=** **false;**  **}**  **if(**\_strEq**[**i**]** **==** '~'**)**  \_stackToken**.**push**(new** Logic**(**"~"**));**  **}**  **else** **{**  **if(!**op**)** **{**  pos **=** i**;** len **=** 0**;**  op **=** **true;**  **}**  len**++;**  **}**  i**++;**  **}**  **if(**op**)** **{**  **if(**modeEquation **==** Extension**::**NumberMode**)** **{**  **if(**modeNumber **==** Extension**::**ArabMode**)**  \_stackToken**.**push**(new** NumberArab**(**\_strEq**.**substr**(**pos**,** len**)));**  **else** **if(**modeNumber **==** Extension**::**RomawiMode**)**  \_stackToken**.**push**(new** NumberRomawi**(**\_strEq**.**substr**(**pos**,** len**)));**  **else**  assert**(false);**  **}**  **else** **{**  **if(**modeEquation **==** Extension**::**LogicMode**)**  **{**  bool isLogic **=** **false;**  char LogicCheck**[]** **=** **{**'T'**,** 't'**,** 'F'**,** 'f'**,** '~'**,** '&'**,** '^'**,** '|'**};**  **for(**int i **=** 0**;** i**<**8 **&&** **!**isLogic**;** i**++)**  isLogic **=** **(**\_strEq**[**pos**]** **==** LogicCheck**[**i**]);**  **if(**isLogic**)**  \_stackToken**.**push**(new** Logic**(**\_strEq**.**substr**(**pos**,** len**)));**  **else**  **if(**modeNumber **==** Extension**::**ArabMode**)**  \_stackToken**.**push**(new** NumberArab**(**\_strEq**.**substr**(**pos**,** len**)));**  **else** **if(**modeNumber **==** Extension**::**RomawiMode**)**  \_stackToken**.**push**(new** NumberRomawi**(**\_strEq**.**substr**(**pos**,** len**)));**  **else**  assert**(false);**  **}**  **else**  assert**(false);**  **}**  **}**  **}**  void Expression**::**\_inToPostfix**()** **{**  /\* mengubah ekspresi infix menjadi prefix \*/  stack**<**Token **\*>** pre**,** opr**;**  string stemp **=** "#"**;**  **while(!**\_stackToken**.**empty**())** **{**  Token **\*** t **=** \_stackToken**.**top**();**  \_stackToken**.**pop**();**  **if(**t**->**GetIsOperator**())** **{**  **if(**t**->**GetSymToken**()[**0**]** **==** '~'**)** **{**  char c **=** stemp**[**0**];**  **if((**c **<** '0' **||** c **>** '9'**)** **&&** **(**c **!=** '-' **||** stemp**.**size**()<=**1**)** **&&** **(**c **!=** '('**)** **&&** **((**c **<** 'a'**)** **||** **(**c **>** 'z'**))** **&&** **((**c **<** 'A'**)** **||** **(**c **>** 'Z'**))** **&&** **(**c **!=** '~'**))**  **throw(**EquationException**(**EquationException**::**IllegalUsingOperator**));**  **}**  **if((**t**->**GetSymToken**())[**0**]** **==** ')'**)** **{**  opr**.**push**(**t**);**  **continue;**  **}**  **else** **if((**t**->**GetSymToken**())[**0**]** **==** '('**)** **{**  **while(!**opr**.**empty**())**  **if((**opr**.**top**()->**GetSymToken**())[**0**]** **==** ')'**)** **{**  opr**.**pop**();**  **break;**  **}**  **else** **{**  pre**.**push**(**opr**.**top**());**  opr**.**pop**();**  **}**  **continue;**  **}**  **while(!**opr**.**empty**())** **{**  **if(**t**->**GetPrior**()** **<** opr**.**top**()->**GetPrior**())** **{**  pre**.**push**(**opr**.**top**());**  opr**.**pop**();**  **}**  **else**  **break;**  **}**  opr**.**push**(**t**);**  **}**  **else**  pre**.**push**(**t**);**  stemp **=** t**->**GetSymToken**();**  **}**  **while(!**opr**.**empty**())** **{**  pre**.**push**(**opr**.**top**());**  opr**.**pop**();**  **}**  **while(!**pre**.**empty**())** **{**  \_stackToken**.**push**(**pre**.**top**());**  pre**.**pop**();**  **}**  **}**  void Expression**::**\_postToPrefix**()** **{**  /\* mengubah ekspresi postfix menjadi prefix \*/  stack**<**Token **\*>** pre**,** stk**;**  Token **\*** flag **=** **new** Token**(**"#"**);**  **while(!**\_stackToken**.**empty**())** **{**  **if(**\_stackToken**.**top**()->**GetIsOperator**())**  stk**.**push**(**\_stackToken**.**top**());**  **else**  **{**  pre**.**push**(**\_stackToken**.**top**());**  **while(!**stk**.**empty**()** **&&** stk**.**top**()** **==** flag**)**  **{**  stk**.**pop**();**  pre**.**push**(**stk**.**top**());**  stk**.**pop**();**  **}**  stk**.**push**(**flag**);**  **}**  \_stackToken**.**pop**();**  **}**  **while(!**pre**.**empty**())** **{**  \_stackToken**.**push**(**pre**.**top**());**  pre**.**pop**();**  **}**  **}** |

* 1. Equation.h

|  |
| --- |
| /\*\*  \* @file Equation.h  \*\*/  #ifndef \_\_EQUATION\_H\_  #define \_\_EQUATION\_H\_  #include "Expression.h"  #include "EquationException.h"  /\*\*  \* @class Equation  \* @author Wiwit Rifa'i (13513073)  \* @version 1.0  \*  \* @section Description  \* @brief Kelas Equation bertanggung jawab dalam perhutingan hasil ekspresi matematika ataupun logika dalam bentuk prefix. Kelas Equation merupakan inheritent dari kelas Expression.  \*\*/  class Equation **:** public Expression **{**  public**:**  /\*\*  \* @brief Konstruktor kelas Equation.  \*\*/  Equation**();**  /\*\*  \* @brief Konstruktor kelas Equation dengan parameter.  \* @param int strEqin - String yang merepresentasikan ekspresi matematika atau logika yang akan dikelola.  \* @param int modeExpress - kode integer yang menentukan apakah ekspresi dalam string tersebut dalam bentuk prefix, infix, ataupun postfix.  \* @param int modeEquation - kode integer yang menentukan jenis operasi yang digunakan yaitu operais pada bilangan ataukah logika.  \* @param int modeNumber - kode integer yang menentukan jenis operan bilangan yang digunakan yaitu bilangan arab ataupun bilangan romawi.  \*\*/  Equation**(**string strEqin**,** int modeExpress**,** int modeEquation**,** int modeNumber**);**  /\*\*  \* @brief Copy Constructor kelas Equation  \*\*/  Equation**(**const Equation**&);**  /\*\*  \* @brief Destructor kelas Equation  \*\*/  virtual **~**Equation**();**  /\*\*  \* @brief operator assignment kelas Equation  \*\*/  Equation**&** **operator=(**const Equation**&);**  /\*\*  \* @brief Getter hasil perhitungan dari ekspresi  \* @return string \_result - string yang merepresentasikan hasil dari perhitungan  \*\*/  string GetResult**();**  /\*\*  \* @brief Menghitung hasil ekspresi matematika yang ada dalam stack<token \* > \_stackToken.  \*\*/  void SolveMathematical**();**  /\*\*  \* @brief Menghitung hasil ekspresi logika yang ada dalam stack<token \* > \_stackToken.  \*\*/  void SolveLogical**();**  /\*\*  \* @brief Menghitung operasi bilangan yaitu untuk operator (+), (-), (\*), dan (/).  \* @param Number \* opn1 - pointer pada operan bilangan pertama  \* @param Token \* opr - pointer pada operator bilangan  \* @param Number \* opn2 - pointer pada operan bilangan kedua  \* @return Number \* ret - pointer pada bilangan hasil dari operasi  \*\*/  Number **\*** CalculateNumber**(**Number **\*** opn1**,** Token **\*** opr**,** Number **\*** opn2**);**  /\*\*  \* @brief Menghitung operasi logika yaitu untuk operator (&), (|), dan (^).  \* @param Logic \* opn1 - pointer pada operan logika pertama  \* @param Token \* opr - pointer pada operator logika  \* @param Logic \* opn2 - pointer pada operan logika kedua  \* @return Logic \* ret - pointer pada hasil logika dari operasi  \*\*/  Logic **\*** CalculateLogic**(**Logic **\*** opn1**,** Token **\*** opr**,** Logic **\*** opn2**);**  /\*\*  \* @brief Menghitung operasi relasional pada bilangan yaitu untuk operator (<), (<=), (>), (>=), (=) dan (!=).  \* @param Number \* opn1 - pointer pada operan bilangan pertama  \* @param Token \* opr - pointer pada operator relasional  \* @param Number \* opn2 - pointer pada operan bilangan kedua  \* @return Logic \* ret - pointer pada hasil logika dari operasi  \*\*/  Logic **\*** CalculateNumberToLogic**(**Number **\*** opn1**,** Token **\*** opr**,** Number **\*** opn2**);**    private**:**  string \_strEq**;** // \_strEq adalah string yang merepresentasikan ekspresi awal  string \_result**;** // \_result adalah string yang merepresentasikan hasil dari ekpresi  int \_modeEquation**;** // \_modeEquation adalah kode yang merepresentasikan mode operasi yang digunakan logika ataukah bilangan  int \_modeNumber**;** // \_modeNumber adalah kode yang merepresentasikan mode operan bilangan yang digunakan yaitu bilangan arab atau romawi  **};**  #endif |

* 1. Equation.cpp

|  |
| --- |
| /\*\*  \* @file Equation.cpp  \*\*/  #include "Equation.h"  **using** **namespace** std**;**  Equation**::**Equation**()** **{**  \_modeEquation **=** Extension**::**defaultEquationMode**;**  \_modeNumber **=** Extension**::**defaultNumberMode**;**  \_result **=** ""**;**  **}**  Equation**::**Equation**(**string strEqin**,** int modeExpress**,** int modeEquation**,** int modeNumber**)** **:** Expression**(**strEqin**,** modeExpress**,** modeEquation**,** modeNumber**)** **{**  \_strEq **=** strEqin**;**  \_modeEquation **=** modeEquation**;**  \_modeNumber **=** modeNumber**;**  **}**  Equation**::**Equation**(**const Equation**&** E**)** **:** Expression**(**E**){**  \_strEq **=** E**.**\_strEq**;**  \_result **=** E**.**\_result**;**  \_modeEquation **=** E**.**\_modeEquation**;**  \_modeNumber **=** E**.**\_modeNumber**;**  \_stackToken **=** E**.**\_stackToken**;**  **}**  Equation**::~**Equation**()** **{**  Token **\*** t**;**  **while(!**\_stackToken**.**empty**())** **{**  t **=** \_stackToken**.**top**();**  \_stackToken**.**pop**();**  **delete** t**;**  **}**  **}**  Equation**&** Equation**::operator=(**const Equation**&** E**)** **{**  \_strEq **=** E**.**\_strEq**;**  \_stackToken **=** E**.**\_stackToken**;**  **}**  string Equation**::**GetResult**()** **{**  **if(**\_result**.**length**()** **<** 1**)** **{**  **if(**\_modeEquation **==** Extension**::**NumberMode**)** **{**  SolveMathematical**();**  **}** **else** **if(**\_modeEquation **==** Extension**::**LogicMode**)** **{**  SolveLogical**();**  **}** **else** **{**  assert**(false);**  **}**  **}**  **return** \_result**;**  **}**  void Equation**::**SolveMathematical**()** **{**  Number **\*** opn1**,** **\*** opn2**;**  Token **\*** opr**;**  stack**<**Number **\*>** res**;**  **if(**\_stackToken**.**empty**())** **{**  **throw(**EquationException**(**EquationException**::**EmptyEquation**));**  **}**  **while(!**\_stackToken**.**empty**())** **{**  opr **=** **(**Token **\*)**\_stackToken**.**top**();** \_stackToken**.**pop**();**  **if(**opr**->**GetIsOperator**())** **{**  **if(**res**.**empty**())**  **throw(**EquationException**(**EquationException**::**IllegalUsingOperator**));**  opn1 **=** res**.**top**();** res**.**pop**();**  **if(**res**.**empty**())**  **throw(**EquationException**(**EquationException**::**IllegalUsingOperator**));**  opn2 **=** res**.**top**();** res**.**pop**();**  // cout << opn1-> GetSymToken() << " " << opr->GetSymToken() << " " << opn2->GetSymToken() << " = ";  opn1 **=** CalculateNumber**(**opn1**,** opr**,** opn2**);**  // cout << opn1->GetSymToken() << endl;  res**.**push**(**opn1**);**  **}**  **else**  res**.**push**((**Number **\*)**opr**);**  **}**    **if(**res**.**size**()** **!=** 1**)**  **throw(**EquationException**(**EquationException**::**IllegalUsingOperator**));**  \_stackToken**.**push**((**Token **\*)**res**.**top**());**  \_result **=** res**.**top**()->**GetSymToken**();**  **}**  void Equation**::**SolveLogical**()** **{**  Logic **\*** opl1**,** **\*** opl2**;**  Number **\*** opn1**,** **\*** opn2**;**  Token **\*** opr**;**  stack**<**Token **\*>** res**;**  **if(**\_stackToken**.**empty**())** **{**  **throw(**EquationException**(**EquationException**::**EmptyEquation**));**  **}**  **while(!**\_stackToken**.**empty**())** **{**  opr **=** **(**Token **\*)**\_stackToken**.**top**();** \_stackToken**.**pop**();**  **if(**opr**->**GetIsOperator**())** **{**  **if(**res**.**empty**())**  **throw(**EquationException**(**EquationException**::**IllegalUsingOperator**));**  **if(**opr**->**GetPrior**()** **<=** 6**)** **{** // hitung logika  opl1 **=** **(**Logic **\*)**res**.**top**();** res**.**pop**();**    **if(**opl1**->**GetPrior**()** **!=** 0**)**  **throw(**EquationException**(**EquationException**::**IllegalUsingOperator**));**    **if(**opr**->**GetSymToken**()[**0**]** **==** '~'**)** **{**// kasus operator unary negasi (~)  // cout << opr-> GetSymToken() << " " << opl1->GetSymToken() << " = " << (~(\*opl1)).GetSymToken() << endl;  **(\***opl1**)** **=** **~(\***opl1**);**  res**.**push**(**opl1**);**  **continue;**  **}**    **if(**res**.**empty**())**  **throw(**EquationException**(**EquationException**::**IllegalUsingOperator**));**  opl2 **=** **(**Logic **\*)**res**.**top**();** res**.**pop**();**  **if(**opl2**->**GetPrior**()** **!=** 0**)**  **throw(**EquationException**(**EquationException**::**IllegalUsingOperator**));**    // cout << opl1-> GetSymToken() << " " << opr->GetSymToken() << " " << opl2->GetSymToken() << " = ";  opl1 **=** CalculateLogic**(**opl1**,** opr**,** opl2**);**  // cout << opl1->GetSymToken() << endl;  res**.**push**((**Token **\*)**opl1**);**  **}**  **else** **{** // hitung number  opn1 **=** **(**Number **\*)**res**.**top**();** res**.**pop**();**  **if(**opn1**->**GetPrior**()** **!=** 1**)**  **throw(**EquationException**(**EquationException**::**IllegalUsingOperator**));**    **if(**res**.**empty**())**  **throw(**EquationException**(**EquationException**::**IllegalUsingOperator**));**  opn2 **=** **(**Number **\*)**res**.**top**();** res**.**pop**();**  **if(**opn2**->**GetPrior**()** **!=** 1**)**  **throw(**EquationException**(**EquationException**::**IllegalUsingOperator**));**    // cout << opn1-> GetSymToken() << " " << opr->GetSymToken() << " " << opn2->GetSymToken() << " = ";  **if(**opr**->**GetPrior**()>**8**)** **{**  opn1 **=** CalculateNumber**(**opn1**,** opr**,** opn2**);**  // cout << opn1->GetSymToken() << endl;  res**.**push**((**Token **\*)**opn1**);**  **}**  **else** **{**  opl1 **=** CalculateNumberToLogic**(**opn1**,** opr**,** opn2**);**  // cout << opl1->GetSymToken() << endl;  res**.**push**((**Token **\*)**opl1**);**  **}**  **}**  **}**  **else**  res**.**push**(**opr**);**  **}**    **if(**res**.**size**()** **!=** 1**)**  **throw(**EquationException**(**EquationException**::**IllegalUsingOperator**));**  \_stackToken**.**push**((**Token **\*)**res**.**top**());**  \_result **=** res**.**top**()->**GetSymToken**();**  **}**  Number**\*** Equation**::**CalculateNumber**(**Number **\*** opn1**,** Token **\*** opr**,** Number **\*** opn2**)** **{**  Number **\*** ret**;**  **if(**\_modeNumber **==** Extension**::**ArabMode**)** **{**  ret **=** **new** NumberArab**();**  **}** **else** **if(**\_modeNumber **==** Extension**::**RomawiMode**)** **{**  ret **=** **new** NumberRomawi**();**  **}** **else** **{**  assert**(false);**  **}**    **if(**opr**->**GetSymToken**()** **==** "\*"**)**  **(\***ret**)** **=** **(\***opn1**)** **\*** **(\***opn2**);**  **else** **if(**opr**->**GetSymToken**()** **==** "/"**)** **{**  **if(**opn2**->**getNilai**()** **==** 0**)**  **throw(**EquationException**(**EquationException**::**DivideByZero**));**  **(\***ret**)** **=** **(\***opn1**)** **/** **(\***opn2**);**  **}**  **else** **if(**opr**->**GetSymToken**()** **==** "+"**)**  **(\***ret**)** **=** **(\***opn1**)** **+** **(\***opn2**);**  **else** **if(**opr**->**GetSymToken**()** **==** "-"**)**  **(\***ret**)** **=** **(\***opn1**)** **-** **(\***opn2**);**  **else** **if(**opr**->**GetSymToken**()** **==** "%"**)** **{**  **if(**opn2**->**getNilai**()** **<=** 0**)**  **throw(**EquationException**(**EquationException**::**ModuloByNonPositif**));**  **(\***ret**)** **=** **(\***opn1**)** **%** **(\***opn2**);**  **}**  **else**  **throw(**EquationException**(**EquationException**::**UndefinedOperator**));**  **if(**\_modeNumber **==** Extension**::**RomawiMode **&&** **(**ret**->**getNilai**()** **<** 1 **||** ret**->**getNilai**()** **>** 3999**))**  **throw(**EquationException**(**EquationException**::**OutOfBoundRomawi**));**  **delete** opn1**;**  **delete** opn2**;**  **return** ret**;**  **}**  Logic **\*** Equation**::**CalculateLogic**(**Logic **\*** opn1**,** Token **\*** opr**,** Logic **\*** opn2**)** **{**  Logic **\*** ret **=** **new** Logic**();**  **if(**opr**->**GetSymToken**()** **==** "&"**)**  **(\***ret**)** **=** **(\***opn1**)** **&** **(\***opn2**);**  **else** **if(**opr**->**GetSymToken**()** **==** "|"**)**  **(\***ret**)** **=** **(\***opn1**)** **|** **(\***opn2**);**  **else** **if(**opr**->**GetSymToken**()** **==** "^"**)**  **(\***ret**)** **=** **(\***opn1**)** **^** **(\***opn2**);**  **else**  **throw(**EquationException**(**EquationException**::**UndefinedOperator**));**  **delete** opn1**;**  **delete** opn2**;**  **return** ret**;**  **}**  Logic **\*** Equation**::**CalculateNumberToLogic**(**Number **\*** opn1**,** Token **\*** opr**,** Number **\*** opn2**)** **{**  Logic **\*** ret **=** **new** Logic**();**  **if(**opr**->**GetSymToken**()** **==** "<"**)**  **(\***ret**)** **=** **(\***opn1**)** **<** **(\***opn2**);**  **else** **if(**opr**->**GetSymToken**()** **==** "<="**)**  **(\***ret**)** **=** **(\***opn1**)** **<=** **(\***opn2**);**  **else** **if(**opr**->**GetSymToken**()** **==** ">"**)**  **(\***ret**)** **=** **(\***opn1**)** **>** **(\***opn2**);**  **else** **if(**opr**->**GetSymToken**()** **==** ">="**)**  **(\***ret**)** **=** **(\***opn1**)** **>=** **(\***opn2**);**  **else** **if(**opr**->**GetSymToken**()** **==** "="**)**  **(\***ret**)** **=** **(\***opn1**)** **==** **(\***opn2**);**  **else** **if(**opr**->**GetSymToken**()** **==** "!="**)**  **(\***ret**)** **=** **(\***opn1**)** **!=** **(\***opn2**);**  **else**  **throw(**EquationException**(**EquationException**::**UndefinedOperator**));**  **delete** opn1**;**  **delete** opn2**;**  **return** ret**;**  **}** |

* 1. EquationException.h

|  |
| --- |
| /\*\*  \* @file EquationException.h  \*\*/  #ifndef \_\_EQUATION\_EXCEPTION\_H\_  #define \_\_EQUATION\_EXCEPTION\_H\_  #include <string>  **using** **namespace** std**;**  /\*\*  \* @class EquationException  \* @author Wiwit Rifa'i (13513073)  \* @version 1.0  \*  \* @section Description  \* @brief kelas EquationException merupakan kelas mengurusi exception pada equation.  \*\*/  class EquationException **{**  public**:**  //ctor  /\*\*  \* @brief Konstruktor kelas EquationException.  \* @param int - menyatakan jenis error yang terjadi.  \*\*/  EquationException**(**int**);**  /\*\*  \* @brief Copy Constructor kelas EquationException.  \* @param const EquationException& - reference objek yang akan di copy;  \*\*/  EquationException**(**const EquationException**&);**  /\*\*  \* @brief Destruktor kelas EquationException.  \*\*/  **~**EquationException**();**  /\*\*  \* @brief Operator assignment kelas EquationException.  \* @param EquationException& - reference objek yang akan di-copy.  \*\*/  EquationException**&** **operator=(**const EquationException**&);**  /\*\*  \* @brief konstanta yang menandakan nilai romawi diluar batas [1..3999]  \*\*/  const static int OutOfBoundRomawi**;**  /\*\*  \* @brief konstanta yang menandakan pembagian dengan nol.  \*\*/  const static int DivideByZero**;**  /\*\*  \* @brief konstanta yang menandakan modulo dengan bilangan yang tak positif.  \*\*/  const static int ModuloByNonPositif**;**  /\*\*  \* @brief konstanta yang menandakan operator yang tidak diperbolehkan.  \*\*/  const static int UndefinedOperator**;**  /\*\*  \* @brief konstanta yang menandakan penggunaan operator yang tidak sesuai.  \*\*/  const static int IllegalUsingOperator**;**  /\*\*  \* @brief konstanta yang menandakan program mencoba menyelesaikan Equation kosong.  \*\*/  const static int EmptyEquation**;**    //method  /\*\*  \* @brief Getter banyaknya exception yang telah terjadi.  \*\*/  static int getNumOfException**();**  /\*\*  \* @brief Getter nilai ID kesalahan.  \*\*/  const int getID**();**  /\*\*  \* @brief Getter pesan kesalahan.  \*\*/  string getMessage**();**    private**:**  static int \_num\_ex**;** // banyaknya exception yang telah terjadi.  static string \_message**[];** // pesan kesalahan.  const int \_msg\_id**;** // ID pesan kesalahan.  **};**  #endif |

* 1. EquationException.cpp

|  |
| --- |
| #include "EquationException.h"  int EquationException**::**\_num\_ex **=** 0**;**  const int EquationException**::**OutOfBoundRomawi **=** 0**;** // nilai romawi diluar batas [1..3999]  const int EquationException**::**DivideByZero **=** 1**;** // pembagian dengan nol  const int EquationException**::**ModuloByNonPositif **=** 2**;** // Modulo dengan bilangan yang tak positif  const int EquationException**::**UndefinedOperator **=** 3**;** // operator yang tidak diperbolehkan  const int EquationException**::**IllegalUsingOperator **=** 4**;** // penggunaan operator yang tidak sesuai  const int EquationException**::**EmptyEquation **=** 5**;** // Jika mencoba menyelesaikan Equation kosong  string EquationException**::**\_message**[]** **=** **{**string**(**"Angka Romawi hanya diperbolehkan pada batas nilai[1..3999]"**),**  string**(**"Tidak diijinkan pembagian dengan nol"**),**  string**(**"Tidak diijinkan operasi modulo oleh bilangan non positif"**),**  string**(**"Operator yang digunakan tidak terdefinisi"**),**  string**(**"Penggunaan operator tidak sesuai"**),**  string**(**"Equation kosong, tidak ada yang diselesaikan"**)**  **};**  //ctor  EquationException**::**EquationException**(**int id**):** \_msg\_id**(**id**)** **{**  \_num\_ex**++;**  **}**  EquationException**::**EquationException**(**const EquationException**&** s**):**\_msg\_id**(**s**.**\_msg\_id**)** **{**  **}**  EquationException**::~**EquationException**()** **{**  **}**    EquationException**&** EquationException**::operator=(**const EquationException**&)** **{**  **return** **\*this;**  **}**  int EquationException**::**getNumOfException**()** **{**  **return** \_num\_ex**;**  **}**  const int EquationException**::**getID**()** **{**  **return** \_msg\_id**;**  **}**  //method getMessage  string EquationException**::**getMessage**()** **{**  **return** \_message**[**\_msg\_id**];**  **}** |

1. Extension
   1. Extension.h

|  |
| --- |
| #ifndef \_\_EXTENSION\_H  #define \_\_EXTENSION\_H  /\*\*  \* @class Extension  \* @author Luqman A. Siswanto (13513024)  \* @version 1.0  \*  \* @section Description  \* @brief Kelas Extension berisi konstanta yang dibutuhkan dalam program  \*\*/  class Extension **{**  public**:**  /\*\*  \* @brief Konstruktor kelas Extension.  \*\*/  Extension**();**  /\*\*  \* @brief Desktruktor kelas Extension.  \*\*/  **~**Extension**();**  static const int Prefix**;** // konstanta mode prefiks  static const int Infix**;** // konstanta mode infiks  static const int Postfix**;** // konstanta mode postfiks    static const int NumberMode**;** // konstanta mode bilangan  static const int LogicMode**;** // konstanta mode logika    static const int ArabMode**;** // konstanta mode bilangan arab  static const int RomawiMode**;** // konstanta mode bilangan romawi    static const int defaultExpressionMode**;** // prefix, infix, postfix  static const int defaultEquationMode**;** // bilangan / logika  static const int defaultNumberMode**;** // arab / romawi  private**:**  **};**  #endif // \_\_EXTENSION\_H |

* 1. Extension.cpp

|  |
| --- |
| #include "Extension.h"  Extension**::**Extension**()** **{**  **}**  Extension**::~**Extension**()** **{**  **}**  const int Extension**::**Prefix **=** 0**;**  const int Extension**::**Infix **=** 1**;**  const int Extension**::**Postfix **=** 2**;**  const int Extension**::**NumberMode **=** 0**;**  const int Extension**::**LogicMode **=** 1**;**  const int Extension**::**ArabMode **=** 0**;**  const int Extension**::**RomawiMode **=** 1**;**  const int Extension**::**defaultExpressionMode **=** Infix**;**  const int Extension**::**defaultEquationMode **=** NumberMode**;**  const int Extension**::**defaultNumberMode **=** ArabMode**;** |

1. Package Token
   1. Token.h

|  |
| --- |
| /\*\*  \* @file Token.h  \*\*/  #ifndef Token\_H  #define Token\_H  #include <iostream>  **using** **namespace** std**;**  /\*\*  \* @class Token  \* @author Ahmad Darmawan (13513096)  \* @version 1.0  \*  \* @brief Kelas Token menyimpan operator dan operan.  \*\*/  class Token **{**  public **:**  /\*\*  \* @brief Konstruktor kelas Token dengan Parameter.  \* @param S - String yang menjadi anggota token.  \*\*/  Token**(**string S**);**  /\*\*  \* @brief Copy Konstruktor kelas Token.  \* @param T - Variabel Token masukan.  \* @return Ti - Variabel Token yang tercopy.  \*\*/  Token**(**const Token**&** T**);**  /\*\*  \* @brief Assignment kelas Logic.  \* @param T - Variabel Token masukan.  \* @return Ti - Variabel Token yang terassign.  \*\*/  Token**&** **operator=(**const Token**&** T**);**  /\*\*  \* @brief Destruktor kelas Token.  \*\*/  **~**Token**();**    /\*\*  \* @brief Getter untuk mendapatkan nilai \_symToken.  \* @return S - string nilai \_symToken.  \*\*/  string GetSymToken**()** const**;**  /\*\*  \* @brief Getter untuk mendapatkan nilai \_isOperator.  \* @return bool - boolean nilai \_isOperator.  \*\*/  bool GetIsOperator**();**  /\*\*  \* @brief Setter untuk mengubah nilai \_symToken.  \* @brief [I.S] String S merupakan simbol token yang valid.  \* @brief [F.S] Simbol token akan terset dengan S.  \* @param S - nilai \_symToken yang akan dimasukkan.  \*\*/  void SetSymToken**(**string S**);**  /\*\*  \* @brief Getter untuk mendapatkan nilai \_prior.  \* @return i - integer nilai \_prior.  \*\*/  int GetPrior**();**  private **:**  /\*\*  \* @brief Set nilai \_isOperator berdasarkan \_symToken.  \* @return bool - nilai \_isOperator  \*\*/  bool \_IdentifyToken**();**  /\*\*  \* @brief Set nilai \_prior berdasarkan \_symToken (untuk operator).  \* @brief [I.S] Token memiliki simbol token yang valid.  \* @brief [F.S] Priority token akan terset berdasarkan simbol token.  \*\*/  void \_IdentifyPrior**();**  string \_symToken**;** // Simbol Token.  bool \_isOperator**;** // Nilai 1 jika token merupakan operator.  int \_prior**;** // Nilai sesuai prioritas operator.  **};**  #endif |

* 1. Token.cpp

|  |
| --- |
| /\*\*  \* @file Token.h  \*\*/  #include <iostream>  #include "Token.h"  **using** **namespace** std**;**  /\*\*  \* @brief Konstruktor kelas Token dengan Parameter.  \* @param S - String yang menjadi anggota token.  \*\*/  Token**::**Token**(**string S**)** **{**  \_symToken **=** S**;**  \_isOperator **=** \_IdentifyToken**();**  \_IdentifyPrior**();**  **}**  /\*\*  \* @brief Copy Konstruktor kelas Token.  \* @param T - Variabel Token masukan.  \* @return Ti - Variabel Token yang tercopy.  \*\*/  Token**::**Token**(**const Token**&** T**)** **{**  **this->**\_symToken **=** T**.**\_symToken**;**  **this->**\_isOperator **=** T**.**\_isOperator**;**  **this->**\_prior **=** T**.**\_prior**;**  **}**  /\*\*  \* @brief Assignment kelas Logic.  \* @param T - Variabel Token masukan.  \* @return Ti - Variabel Token yang terassign.  \*\*/  Token**&** Token**::operator=(**const Token**&** T**)** **{**  **this->**\_symToken **=** T**.**\_symToken**;**  **this->**\_isOperator **=** T**.**\_isOperator**;**  **this->**\_prior **=** T**.**\_prior**;**  **return** **\*this;**  **}**  /\*\*  \* @brief Destruktor kelas Token.  \*\*/  Token**::~**Token**()** **{**  **}**    /\*\*  \* @brief Getter untuk mendapatkan nilai \_symToken.  \* @return S - string nilai \_symToken.  \*\*/  string Token**::**GetSymToken**()** const **{**  **return** \_symToken**;**  **}**  /\*\*  \* @brief Getter untuk mendapatkan nilai \_isOperator.  \* @return bool - boolean nilai \_isOperator.  \*\*/  bool Token**::**GetIsOperator**()** **{**  **return** \_isOperator**;**  **}**  /\*\*  \* @brief Setter untuk mengubah nilai \_symToken.  \* @brief [I.S] String S merupakan simbol token yang valid.  \* @brief [F.S] Simbol token akan terset dengan S.  \* @param S - nilai \_symToken yang akan dimasukkan.  \*\*/  void Token**::**SetSymToken**(**string S**)** **{**  \_symToken **=** S**;**  \_isOperator **=** \_IdentifyToken**();**  \_IdentifyPrior**();**  **}**  /\*\*  \* @brief Set nilai \_isOperator berdasarkan \_symToken.  \* @return bool - nilai \_isOperator  \*\*/  bool Token**::**\_IdentifyToken**()** **{**  **if** **((**\_symToken**[**0**]** **==** '-'**)** **&&** **(**\_symToken**.**length**()** **>** 1**))**  **return** **false;**  **else**  **return** **(**\_symToken**[**0**]** **<** '0' **||** \_symToken**[**0**]** **>** '9'**)** **&&** **(**\_symToken**[**0**]** **<** 'A' **||** \_symToken**[**0**]** **>** 'Z'**)** **&&** **(**\_symToken**[**0**]** **<** 'a' **||** \_symToken**[**0**]** **>** 'z'**);**  **}**  /\*\*  \* @brief Set nilai \_prior berdasarkan \_symToken (untuk operator).  \* @brief [I.S] Token memiliki simbol token yang valid.  \* @brief [F.S] Priority token akan terset berdasarkan simbol token.  \*\*/  void Token**::**\_IdentifyPrior**()** **{**  **if(**\_isOperator**)** **{**  /\* Prioritas operator relatif berdasarkan http://en.wikipedia.org/wiki/Order\_of\_operations \*/  /\* operator number to number \*/  **if(**\_symToken **==** "\*" **||** \_symToken **==** "/" **||** \_symToken **==** "%"**)**  \_prior **=** 10**;**  **else** **if(**\_symToken **==** "+" **||** \_symToken **==** "-"**)**  \_prior **=** 9**;**  /\* operator number to logic \*/  **else** **if(**\_symToken **==** "<" **||** \_symToken **==** "<=" **||** \_symToken **==** ">" **||** \_symToken **==** ">="**)**  \_prior **=** 8**;**  **else** **if(**\_symToken **==** "=" **||** \_symToken **==** "!="**)**  \_prior **=** 7**;**  /\* operator logic to logic \*/  **else** **if(**\_symToken **==** "~"**)**  \_prior **=** 6**;**  **else** **if(**\_symToken **==** "&"**)**  \_prior **=** 5**;**  **else** **if(**\_symToken **==** "^"**)**  \_prior **=** 4**;**  **else** **if(**\_symToken **==** "|"**)**  \_prior **=** 3**;**  **else**  \_prior **=** 2**;**  **}**  **else**  **if(**\_symToken **==** "TRUE" **||** \_symToken **==** "FALSE" **||** \_symToken **==** "true" **||** \_symToken **==** "false"**)** // a logic  \_prior **=** 0**;**  **else** // a number  \_prior **=** 1**;**  **}**  /\*\*  \* @brief Getter untuk mendapatkan nilai \_prior.  \* @return i - integer nilai \_prior.  \*\*/  int Token**::**GetPrior**()** **{**  **return** \_prior**;**  **}** |

* 1. Number.h

|  |
| --- |
| /\*\*  \* @file Number.h  \*\*/  #include <iostream>  #ifndef \_Number\_H\_  #define \_Number\_H\_  #include "Token.h"  #include "Logic.h"  #include <string>  **using** **namespace** std**;**  /\*\*  \* @class Number  \* @author Muhammad Aodyra Khaidir (13513063)  \* @version 1.0  \*  \* @section Description  \* @brief Kelas Number bertugas untuk mengelola operasi bilangan.  \*\*/  class Number**:** public Token**{**  public**:**  /\*\*  \* @brief Konstruktor kelas Number.  \*  \*\*/  Number**();**  /\*\*  \* @brief Konstruktor kelas Number dengan parameter.  \* @param s - String yang akan menjadi parameter Token.  \*\*/  Number**(**string s**);**  /\*\*  \* @brief Destruktor kelas Number.  \*\*/  virtual **~**Number**();**  /\*\*  \* @brief operator= untuk menyamakan nilai dari Number n.  \* @param n - Number yang nilainya akan disamakan.  \* @return n' - Objek Number yang telah diassign dari n.  \*\*/  virtual Number**&** **operator=** **(**Number**&** n**);**  /\*\*  \* @brief operator\* untuk mengali nilai objek dengan nilai dari Number n.  \* @param n - Number yang nilainya akan dikali.  \* @return n' - Objek Number yang telah diassign dari n.  \*\*/  virtual Number**&** **operator\*** **(**const Number**&)** **=** 0**;**  /\*\*  \* @brief operator+ untuk menambahkan nilai objek dengan nilai dari Number n.  \* @param n - Number yang nilainya akan ditambah.  \* @return n' - Objek Number yang telah diassign dari n.  \*\*/  virtual Number**&** **operator+** **(**const Number**&)** **=** 0**;**  /\*\*  \* @brief operator- untuk mengurangi nilai objek dengan nilai dari Number n.  \* @param n - Number yang nilainya akan dikurang.  \* @return n' - Objek Number yang telah diassign dari n.  \*\*/  virtual Number**&** **operator-** **(**const Number**&)** **=** 0**;**  /\*\*  \* @brief operator/ untuk membagi nilai objek dengan nilai dari Number n.  \* @param n - Number yang nilainya akan dibagi.  \* @return n' - Objek Number yang telah diassign dari n.  \*\*/  virtual Number**&** **operator/** **(**const Number**&)** **=** 0**;**  /\*\*  \* @brief operator% untuk modulo nilai objek dengan nilai dari Number n.  \* @param n - Number yang nilainya menjadi modulator.  \* @return n' - Objek Number yang telah diassign dari n.  \*\*/  virtual Number**&** **operator%** **(**const Number**&)** **=** 0**;**  /\*\*  \* @brief fungsi yang merubah string menjadi integer  \* @param s - string yang nantinya akan dirubah menjadi integer.  \* @return i - integer dari string n.  \*\*/  virtual int toInt**(**string**)** **=** 0**;**  /\*\*  \* @brief fungsi yang merubah integer menjadi string  \* @param n - nilai yang nantinya akan dirubah menjadi string.  \* @return s - string dari integer n.  \*\*/  virtual string toString**(**int**)** **=** 0**;**  /\*\*  \* @brief prosedur membandingkan nilai lebih kecil dengan nilai N  \* @param N - Number yang nilainya akan dibandingkan.  \* @return L' = L1 < L2.  \*\*/  Logic**&** **operator<** **(**const Number**&);**  /\*\*  \* @brief prosedur membandingkan nilai lebih kecil sama dengan nilai N.  \* @param N - Number yang nilainya akan dibandingkan.  \* @return L' = L1 <= L2.  \*\*/  Logic**&** **operator<=** **(**const Number**&);**  /\*\*  \* @brief prosedur membandingkan nilai lebih dari dengan nilai N.  \* @param N - Number yang nilainya akan dibandingkan.  \* @return L' = L1 > L2.  \*\*/  Logic**&** **operator>** **(**const Number**&);**  /\*\*  \* @brief prosedur membandingkan nilai lebih dari sama dengan \_nilai N.  \* @param N - Number yang nilainya akan dibandingkan.  \* @return L' = L1 >= L2.  \*\*/  Logic**&** **operator>=** **(**const Number**&);**  /\*\*  \* @brief prosedur membandingkan \_nilai sama dengan \_nilai N.  \* @param N - Number yang nilainya akan dibandingkan.  \* @return L' = L1 == L2.  \*\*/  Logic**&** **operator==** **(**const Number**&);**  /\*\*  \* @brief prosedur membandingkan \_nilai tidak sama dengan \_nilai dari N.  \* @param N - Number yang nilainya akan dibandingkan.  \* @return L' = L1 != L2.  \*\*/  Logic**&** **operator!=** **(**const Number**&);**  /\*\*  \* @brief prosedur mendapatkan \_nilai.  \* @return i - integer dari \_nilai  \*\*/  int getNilai**();**  /\*\*  \* @brief prosedur assign \_nilai  \* @param n - nilai yang nantinya akan di assign ke \_nilai.  \*\*/  void setNilai**(**int \_n**);**    //protected:  int \_nilai**;**  **};**  #endif |

* 1. Number.cpp

|  |
| --- |
| #include "Number.h"  /\*\*  \* @brief Konstruktor kelas Number.  \*  \*\*/  Number**::**Number**():** Token**(**"0"**){**  **}**  /\*\*  \* @brief Konstruktor kelas Number dengan parameter.  \* @param S - String yang akan menjadi nama file.  \*\*/  Number**::**Number**(**string s**):** Token**(**s**)** **{**  **}**  /\*\*  \* @brief Destruktor kelas Number.  \*\*/  Number**::~**Number**()** **{**  **}**  /\*\*  \* @brief operator= untuk menyamakan nilai dari Number n.  \* @param n - Number yang nilainya akan disamakan.  \* @return n' - Objek Number yang telah diassign dari n.  \*\*/  Number**&** Number**::operator=** **(**Number**&** n**)** **{**  \_nilai **=** n**.**\_nilai**;**  **this->**SetSymToken**(**n**.**GetSymToken**());**  **}**  /\*\*  \* @brief prosedur membandingkan nilai lebih kecil dengan nilai N  \* @param N - Number yang nilainya akan dibandingkan.  \* @return L' = L1 < L2.  \*\*/  Logic**&** Number**::operator<** **(**const Number**&** N**)** **{**  Logic**&** L **=** **\*(new** Logic**(**\_nilai **<** N**.**\_nilai**));**  **return** L**;**  **}**  /\*\*  \* @brief prosedur membandingkan nilai lebih kecil sama dengan nilai N.  \* @param N - Number yang nilainya akan dibandingkan.  \* @return L' = L1 <= L2.  \*\*/  Logic**&** Number**::operator<=** **(**const Number**&** N**)** **{**  Logic**&** L **=** **\*(new** Logic**(**\_nilai **<=** N**.**\_nilai**));**  **return** L**;**  **}**  /\*\*  \* @brief prosedur membandingkan nilai lebih dari dengan nilai N.  \* @param N - Number yang nilainya akan dibandingkan.  \* @return L' = L1 > L2.  \*\*/  Logic**&** Number**::operator>** **(**const Number**&** N**)** **{**  Logic**&** L **=** **\*(new** Logic**(**\_nilai **>** N**.**\_nilai**));**  **return** L**;**  **}**  /\*\*  \* @brief prosedur membandingkan nilai lebih dari sama dengan \_nilai N.  \* @param N - Number yang nilainya akan dibandingkan.  \* @return L' = L1 >= L2.  \*\*/  Logic**&** Number**::operator>=** **(**const Number**&** N**)** **{**  Logic**&** L **=** **\*(new** Logic**(**\_nilai **>=** N**.**\_nilai**));**  **return** L**;**  **}**  /\*\*  \* @brief prosedur membandingkan \_nilai sama dengan \_nilai N.  \* @param N - Number yang nilainya akan dibandingkan.  \* @return L' = L1 == L2.  \*\*/  Logic**&** Number**::operator==** **(**const Number**&** N**)** **{**  Logic**&** L **=** **\*(new** Logic**(**\_nilai **==** N**.**\_nilai**));**  **return** L**;**  **}**  /\*\*  \* @brief prosedur membandingkan \_nilai tidak sama dengan \_nilai dari N.  \* @param N - Number yang nilainya akan dibandingkan.  \* @return L' = L1 != L2.  \*\*/  Logic**&** Number**::operator!=** **(**const Number**&** N**)** **{**  Logic**&** L **=** **\*(new** Logic**(**\_nilai **!=** N**.**\_nilai**));**  **return** L**;**  **}**  /\*\*  \* @brief prosedur mendapatkan \_nilai.  \* @return i - integer dari \_nilai  \*\*/  int Number**::**getNilai**(){**  **return** \_nilai**;**  **}**  /\*\*  \* @brief prosedur assign \_nilai  \* @param n - nilai yang nantinya akan di assign ke \_nilai.  \*\*/  void Number**::**setNilai**(**int \_n**){**  \_nilai **=** \_n**;**  **}** |

* 1. NumberArab.h

|  |
| --- |
| /\*\*  \* @file NumberArab.h  \*\*/  #ifndef \_NumberArab\_H\_  #define \_NumberArab\_H\_  #include "Number.h"  #include "../Stack/stack.h"  #include <string>  **using** **namespace** std**;**  /\*\*  \* @class NumberArab  \* @author Muhammad Aodyra Khaidir (13513063)  \* @version 1.0  \*  \* @section Description  \* @brief Kelas NumberArav bertugas untuk mengelola operasi bilangan arab.  \*\*/  class NumberArab**:** public Number**{**  public**:**  /\*\*  \* @brief Konstruktor kelas NumberArab.  \*  \*\*/  NumberArab**();**  /\*\*  \* @brief Konstruktor kelas NumberArab dengan parameter.  \* @param s - String yang akan menjadi parameter Number  \*\*/  NumberArab**(**string s**);**  /\*\*  \* @brief Konstruktor kelas NumberArab dengan parameter.  \* @param \_n - integer yang akan di assign ke \_nilai  \*\*/  NumberArab**(**int \_n**);**  /\*\*  \* @brief operator\* untuk mengali nilai objek dengan nilai dari Number n.  \* @param n - Number yang nilainya akan dikali.  \* @return n' - Objek Number yang telah diassign dari n.  \*\*/  Number**&** **operator\*** **(**const Number**&);**  /\*\*  \* @brief operator+ untuk menambahkan nilai objek dengan nilai dari Number n.  \* @param n - Number yang nilainya akan ditambah.  \* @return n' - Objek Number yang telah diassign dari n.  \*\*/  Number**&** **operator+** **(**const Number**&);**  /\*\*  \* @brief operator- untuk mengurangi nilai objek dengan nilai dari Number n.  \* @param n - Number yang nilainya akan dikurang.  \* @return n' - Objek Number yang telah diassign dari n.  \*\*/  Number**&** **operator-** **(**const Number**&);**  /\*\*  \* @brief operator/ untuk membagi nilai objek dengan nilai dari Number n.  \* @param n - Number yang nilainya akan dibagi.  \* @return n' - Objek Number yang telah diassign dari n.  \*\*/  Number**&** **operator/** **(**const Number**&);**  /\*\*  \* @brief fungsi yang merubah string menjadi integer  \* @param s - string yang nantinya akan dirubah menjadi integer.  \* @return i - integer dari string n.  \*\*/  Number**&** **operator%** **(**const Number**&);**  /\*\*  \* @brief fungsi yang merubah string menjadi integer  \* @param s - string yang nantinya akan dirubah menjadi integer.  \* @return i - integer dari string n.  \*\*/  int toInt**(**string s**);**  /\*\*  \* @brief fungsi yang merubah integer menjadi string  \* @param n - nilai yang nantinya akan dirubah menjadi string.  \* @return s - string dari integer n.  \*\*/  string toString**(**int n**);**  /\*\*  \* @brief Destruktor kelas NumberArab.  \*\*/  **~**NumberArab**();**  **};**  #endif |

* 1. NumberArab.cpp

|  |
| --- |
| #include "NumberArab.h"  **using** **namespace** std**;**  /\*\*  \* @brief Konstruktor kelas NumberArab.  \*  \*\*/  NumberArab**::**NumberArab**()** **{**  \_nilai **=** 0**;**  **}**  /\*\*  \* @brief Konstruktor kelas NumberArab dengan parameter.  \* @param s - String yang akan menjadi parameter Number  \*\*/  NumberArab**::**NumberArab**(**string s**):** Number**(**s**){**  \_nilai **=** toInt**(**s**);**  **}**  /\*\*  \* @brief Konstruktor kelas NumberArab dengan parameter.  \* @param \_n - integer yang akan di assign ke \_nilai  \*\*/  NumberArab**::**NumberArab**(**int \_n**):** Number**(**toString**(**\_n**)){**  \_nilai **=** \_n**;**  **}**  /\*\*  \* @brief Destruktor kelas NumberArab.  \*\*/  NumberArab**::~**NumberArab**(){**  **}**  /\*\*  \* @brief operator\* untuk mengali nilai objek dengan nilai dari Number n.  \* @param n - Number yang nilainya akan dikali.  \* @return n' - Objek Number yang telah diassign dari n.  \*\*/  Number**&** NumberArab**::operator\*** **(**const Number**&** N**){**  Number**&** N1 **=** **\*(new** NumberArab**(**\_nilai **\*** N**.**\_nilai**));**  **return** N1**;**  **}**  /\*\*  \* @brief operator+ untuk menambahkan nilai objek dengan nilai dari Number n.  \* @param n - Number yang nilainya akan ditambah.  \* @return n' - Objek Number yang telah diassign dari n.  \*\*/  Number**&** NumberArab**::operator+** **(**const Number**&** N**){**  Number**&** N1 **=** **\*(new** NumberArab**(**\_nilai **+** N**.**\_nilai**));**  **return** N1**;**  **}**  /\*\*  \* @brief operator- untuk mengurangi nilai objek dengan nilai dari Number n.  \* @param n - Number yang nilainya akan dikurang.  \* @return n' - Objek Number yang telah diassign dari n.  \*\*/  Number**&** NumberArab**::operator-** **(**const Number**&** N**){**  Number**&** N1 **=** **\*(new** NumberArab**(**\_nilai **-** N**.**\_nilai**));**  **return** N1**;**  **}**  /\*\*  \* @brief operator/ untuk membagi nilai objek dengan nilai dari Number n.  \* @param n - Number yang nilainya akan dibagi.  \* @return n' - Objek Number yang telah diassign dari n.  \*\*/  Number**&** NumberArab**::operator/** **(**const Number**&** N**){**  Number**&** N1 **=** **\*(new** NumberArab**(**\_nilai **/** N**.**\_nilai**));**  **return** N1**;**  **}**  /\*\*  \* @brief operator% untuk modulo nilai objek dengan nilai dari Number n.  \* @param n - Number yang nilainya menjadi modulator.  \*\*/  Number**&** NumberArab**::operator%** **(**const Number**&** N**){**  /\* Sesuai Euclidean Definition nilai modulo selalu lebih besar atau sama dengan nol \*/  Number**&** N1 **=** **\*(new** NumberArab**(((**\_nilai **%** N**.**\_nilai**)+**N**.**\_nilai**)** **%** N**.**\_nilai**));**  **return** N1**;**  **}**  /\*\*  \* @brief fungsi yang merubah string menjadi integer  \* @param s - string yang nantinya akan dirubah menjadi integer.  \* @return i - integer dari string n.  \*\*/  int NumberArab**::**toInt**(**string s**){**  int nilai **=** 0**,** i **=** 0**;**  **if(**s**[**0**]** **==** '-'**)**  i**++;**  **for(;** i**<**s**.**length**();** i**++)**  nilai **=** nilai**\***10 **+** s**[**i**]** **-** '0'**;**  **if(**s**[**0**]** **==** '-'**)**  nilai **\*=** **-**1**;**  **return** nilai**;**  **}**  /\*\*  \* @brief fungsi yang merubah integer menjadi string  \* @param n - nilai yang nantinya akan dirubah menjadi string.  \* @return s - string dari integer n.  \*\*/  string NumberArab**::**toString**(**int n**)** **{**  string ret**;**  int x **=** n**;**  **if(**x **<** 0**)** **{**  ret**.**push\_back**(**'-'**);**  x **\*=** **-**1**;**  **}**  **if(**x **<** 10**)** **{**  ret**.**push\_back**(**'0'**+**x**);**  **return** ret**;**  **}**  stack**<**char**>** temp**;**  **while(**x**>**0**)** **{**  temp**.**push**((**x **%** 10**)+**'0'**);**  x **/=** 10**;**  **}**  **while(!**temp**.**empty**())** **{**  ret**.**push\_back**(**temp**.**top**());**  temp**.**pop**();**  **}**  **return** ret**;**  **}** |

* 1. NumberRomawi.h

|  |
| --- |
| /\*\*  \* @file NumberRomawi.h  \*\*/  #include "Number.h"  #ifndef \_NumberRomawi\_H\_  #define \_NumberRomawi\_H\_  #include "Number.h"  #include <string>  **using** **namespace** std**;**  /\*\*  \* @class NumberRomawi  \* @author Muhammad Aodyra Khaidir (13513063)  \* @version 1.0  \*  \* @section Description  \* @brief Kelas Number bertugas untuk mengelola operasi bilangan romawi.  \*\*/  class NumberRomawi**:** public Number**{**  public**:**  /\*\*  \* @brief Konstruktor kelas NumberRomawi.  \*  \*\*/  NumberRomawi**();**  /\*\*  \* @brief Konstruktor kelas Number dengan parameter.  \* @param s - String yang akan menjadi parameter Number.  \*\*/  NumberRomawi**(**string s**);**  /\*\*  \* @brief Konstruktor kelas Number dengan parameter.  \* @param \_n - integer yang akan di assign ke \_nilai  \*\*/  NumberRomawi**(**int \_n**);**  /\*\*  \* @brief operator\* untuk mengali nilai objek dengan nilai dari Number n.  \* @param n - Number yang nilainya akan dikali.  \* @return n' - Objek Number yang telah diassign dari n.  \*\*/  Number**&** **operator\*** **(**const Number**&);**  /\*\*  \* @brief operator+ untuk menambahkan nilai objek dengan nilai dari Number n.  \* @param n - Number yang nilainya akan ditambah.  \* @return n' - Objek Number yang telah diassign dari n.  \*\*/  Number**&** **operator+** **(**const Number**&);**  /\*\*  \* @brief operator- untuk mengurangi nilai objek dengan nilai dari Number n.  \* @param n - Number yang nilainya akan dikurang.  \* @return n' - Objek Number yang telah diassign dari n.  \*\*/  Number**&** **operator-** **(**const Number**&);**  /\*\*  \* @brief operator/ untuk membagi nilai objek dengan nilai dari Number n.  \* @param n - Number yang nilainya akan dibagi.  \* @return n' - Objek Number yang telah diassign dari n.  \*\*/  Number**&** **operator/** **(**const Number**&);**  /\*\*  \* @brief operator% untuk modulo nilai objek dengan nilai dari Number n.  \* @param n - Number yang nilainya menjadi modulator.  \* @return n' - Objek Number yang telah diassign dari n.  \*\*/  Number**&** **operator%** **(**const Number**&);**  /\*\*  \* @brief fungsi yang merubah string menjadi integer  \* @param s - string yang nantinya akan dirubah menjadi integer.  \* @return i - integer dari string n.  \*\*/  int toInt**(**string s**);**  /\*\*  \* @brief fungsi yang merubah integer menjadi string  \* @param n - nilai yang nantinya akan dirubah menjadi string.  \* @return s - string dari integer n.  \*\*/  string toString**(**int n**);**  /\*\*  \* @brief Destruktor kelas NumberRomawi.  \*\*/  **~**NumberRomawi**();**  **};**  #endif |

* 1. NumberRomawi.cpp

|  |
| --- |
| #include "NumberRomawi.h"  **using** **namespace** std**;**  /\*\*  \* @brief Konstruktor kelas NumberRomawi.  \*  \*\*/  NumberRomawi**::**NumberRomawi**()** **:** Number**(**"X"**)** **{**  \_nilai **=** 1**;**  **}**  /\*\*  \* @brief Konstruktor kelas Number dengan parameter.  \* @param s - String yang akan menjadi parameter Number.  \*\*/  NumberRomawi**::**NumberRomawi**(**string s**):** Number**(**s**){**  \_nilai **=** toInt**(**s**);**  **}**  /\*\*  \* @brief Konstruktor kelas Number dengan parameter.  \* @param \_n - integer yang akan di assign ke \_nilai  \*\*/  NumberRomawi**::**NumberRomawi**(**int \_n**)** **:** Number**(**toString**(**\_n**))** **{**  \_nilai **=** \_n**;**  **}**  /\*\*  \* @brief Destruktor kelas NumberRomawi.  \*\*/  NumberRomawi**::~**NumberRomawi**(){**  **}**  /\*\*  \* @brief operator\* untuk mengali nilai objek dengan nilai dari Number n.  \* @param n - Number yang nilainya akan dikali.  \* @return n' - Objek Number yang telah diassign dari n.  \*\*/  Number**&** NumberRomawi**::operator\*** **(**const Number**&** N**){**  Number**&** N1 **=** **\*(new** NumberRomawi**(**\_nilai **\*** N**.**\_nilai**));**  **return** N1**;**  **}**  /\*\*  \* @brief operator+ untuk menambahkan nilai objek dengan nilai dari Number n.  \* @param n - Number yang nilainya akan ditambah.  \* @return n' - Objek Number yang telah diassign dari n.  \*\*/  Number**&** NumberRomawi**::operator+** **(**const Number**&** N**){**  Number**&** N1 **=** **\*(new** NumberRomawi**(**\_nilai **+** N**.**\_nilai**));**  **return** N1**;**  **}**  /\*\*  \* @brief operator- untuk mengurangi nilai objek dengan nilai dari Number n.  \* @param n - Number yang nilainya akan dikurang.  \* @return n' - Objek Number yang telah diassign dari n.  \*\*/  Number**&** NumberRomawi**::operator-** **(**const Number**&** N**){**  Number**&** N1 **=** **\*(new** NumberRomawi**(**\_nilai **-** N**.**\_nilai**));**  **return** N1**;**  **}**  /\*\*  \* @brief operator/ untuk membagi nilai objek dengan nilai dari Number n.  \* @param n - Number yang nilainya akan dibagi.  \* @return n' - Objek Number yang telah diassign dari n.  \*\*/  Number**&** NumberRomawi**::operator/** **(**const Number**&** N**){**  Number**&** N1 **=** **\*(new** NumberRomawi**(**\_nilai **/** N**.**\_nilai**));**  **return** N1**;**  **}**  /\*\*  \* @brief operator% untuk modulo nilai objek dengan nilai dari Number n.  \* @param n - Number yang nilainya menjadi modulator.  \* @return n' - Objek Number yang telah diassign dari n.  \*\*/  Number**&** NumberRomawi**::operator%** **(**const Number**&** N**){**  /\* Sesuai Euclidean Definition nilai modulo selalu lebih besar atau sama dengan nol \*/  Number**&** N1 **=** **\*(new** NumberRomawi**(((**\_nilai **%** N**.**\_nilai**)+**N**.**\_nilai**)** **%** N**.**\_nilai**));**  **return** N1**;**  **}**  /\*\*  \* @brief fungsi yang merubah string menjadi integer  \* @param s - string yang nantinya akan dirubah menjadi integer.  \* @return i - integer dari string n.  \*\*/  int NumberRomawi**::**toInt**(**string s**){**  int satuan **=** 0**;**  int puluhan **=** 0**;**  int ratusan **=** 0**;**  int ribuan **=** 0**;**  **for** **(**int i **=** 0**;** i **<=** s**.**length**()** **-** 1**;** i**++){**  **if** **(**s**[**i**]** **==** 'I'**){**  **if(**i**<**s**.**length**()** **-** 1 **&&** **(**s**[**i**+**1**]** **==** 'X' **||** s**[**i**+**1**]** **==** 'V'**)){**    **}**  **else** satuan**++;**  **}**  **if** **(**s**[**i**]** **==** 'V'**){**  **if** **(**s**[**i**-**1**]** **==** 'I'**)** satuan **+=** 4**;**  **else** satuan **+=** 5**;**  **}**  **if** **(**s**[**i**]** **==** 'X'**){**  **if(**s**[**i**-**1**]** **==** 'I'**)** puluhan**+=**9**;**  **else** **if(**i**<**s**.**length**()** **-** 1 **&&** **(**s**[**i**+**1**]** **==** 'L' **||** s**[**i**+**1**]** **==** 'C'**)){**    **}**  **else** puluhan**+=**10**;**  **}**  **if** **(**s**[**i**]** **==** 'L'**){**  **if(**s**[**i**-**1**]** **==** 'X'**)** puluhan**+=**40**;**  **else** puluhan**+=**50**;**  **}**  **if** **(**s**[**i**]** **==** 'C'**){**  **if(**s**[**i**-**1**]** **==** 'X'**)** ratusan**+=**90**;**  **else** **if(**i**<**s**.**length**()** **-** 1 **&&** **(**s**[**i**+**1**]** **==** 'D' **||** s**[**i**+**1**]** **==** 'M'**)){**    **}**  **else** ratusan**+=**100**;**  **}**  **if** **(**s**[**i**]** **==** 'D'**){**  **if(**s**[**i**-**1**]** **==** 'C'**)** ratusan **+=** 400**;**  **else** ratusan **+=** 500**;**  **}**  **if** **(**s**[**i**]** **==** 'M'**){**  **if(**s**[**i**-**1**]** **==** 'C'**)** ribuan**+=**900**;**  **else** ribuan**+=**1000**;**  **}**  **}**  int nilai **=** ribuan **+** ratusan **+** puluhan **+** satuan**;**  **return** nilai**;**  **}**  /\*\*  \* @brief fungsi yang merubah integer menjadi string  \* @param n - nilai yang nantinya akan dirubah menjadi string.  \* @return s - string dari integer n.  \*\*/  string NumberRomawi**::**toString**(**int n**)** **{**  char rom**[]** **=** **{** 'I'**,** 'V'**,** 'X'**,** 'L'**,** 'C'**,** 'D'**,** 'M'**,** '#'**,** '#'**};**  string ret**;**  int x **=** n**,** y**,** z **=** 1000**;**  **for(**int i **=** 6**;** i**>=**0**;** i **-=** 2**)** **{**  y **=** x**/**z**;**  x **%=** z**;**  z **/=** 10**;**  **switch(**y**)** **{**  **case** 3 **:** ret**.**push\_back**(**rom**[**i**]);**  **case** 2 **:** ret**.**push\_back**(**rom**[**i**]);**  **case** 1 **:** ret**.**push\_back**(**rom**[**i**]);**  **break;**  **case** 4 **:** ret**.**push\_back**(**rom**[**i**]);**  **case** 5 **:** ret**.**push\_back**(**rom**[**i**+**1**]);**  **break;**  **case** 6 **:** ret**.**push\_back**(**rom**[**i**+**1**]);**  ret**.**push\_back**(**rom**[**i**]);**  **break;**  **case** 7 **:** ret**.**push\_back**(**rom**[**i**+**1**]);**  ret**.**push\_back**(**rom**[**i**]);**  ret**.**push\_back**(**rom**[**i**]);**  **break;**  **case** 8 **:** ret**.**push\_back**(**rom**[**i**+**1**]);**  ret**.**push\_back**(**rom**[**i**]);**  ret**.**push\_back**(**rom**[**i**]);**  ret**.**push\_back**(**rom**[**i**]);**  **break;**  **case** 9 **:** ret**.**push\_back**(**rom**[**i**]);**  ret**.**push\_back**(**rom**[**i**+**2**]);**  **}**  **}**  **return** ret**;**  **}** |

* 1. Logic.h

|  |
| --- |
| /\*\*  \* @file Logic.h  \*\*/  #ifndef Logic\_H  #define Logic\_H  #include <iostream>  #include "Token.h"  **using** **namespace** std**;**  /\*\*  \* @class Logic  \* @author Ahmad Darmawan (13513096)  \* @version 1.0  \*  \* @brief Kelas Logic mengatur perhitungan operasi logika (~, &, ^, |)  \*\*/  class Logic **:** public Token **{**  public**:**  /\*\*  \* @brief Konstruktor kelas Logic.  \*  \*\*/  Logic**();**  /\*\*  \* @brief Konstruktor kelas Logic dengan parameter string.  \* @param S - String berisi nilai logika (TRUE atau FALSE).  \*\*/  Logic**(**string s**);**  /\*\*  \* @brief Konstruktor kelas Logic dengan parameter integer.  \* @param i - Integer berisi nilai logika (0 atau 1).  \*\*/  Logic**(**int i**);**  /\*\*  \* @brief Copy Konstruktor kelas Logic.  \* @param L - Variabel Logic masukan.  \* @return Li - Variabel Logic yang tercopy.  \*\*/  Logic**(**const Logic**&** L**);**  /\*\*  \* @brief Assignment kelas Logic.  \* @param L - Variabel Logic masukan.  \* @return Li - Variabel Logic yang terassign.  \*\*/  Logic**&** **operator=(**const Logic**&** L**);**  /\*\*  \* @brief Destruktor kelas Logic.  \*\*/  **~**Logic**();**  /\*\*  \* @brief Getter untuk mendapatkan nilai logic.  \* @return i - integer nilai logic (0 atau 1).  \*\*/  int GetLogic**();**  /\*\*  \* @brief Setter untuk mengubah nilai logic.  \* @brief [I.S] integer i terdefinisi (0 atau 1).  \* @brief [F.S] \_logic diassign dengan nilai i.  \* @param S - nilai logic yang akan dimasukkan.  \*\*/  void SetLogic**(**int i**);**  /\*\*  \* @brief Melakukan proses "not" pada nilai logika (~L).  \* @param L - Objek Logika awal.  \* @return ~L - Objek Logika akhir.  \*\*/  Logic**&** **operator~();**  /\*\*  \* @brief Melakukan proses "and" pada nilai logika (L1 & L2).  \* @param L1 - Objek Logika pertama.  \* @param L2 - Objek Logika kedua.  \* @return L' = L1 & L2.  \*\*/  Logic**&** **operator&(**const Logic**&** L**);**  /\*\*  \* @brief Melakukan proses "or" pada nilai logika (L1 | L2).  \* @param L1 - Objek Logika pertama.  \* @param L2 - Objek Logika kedua.  \* @return L' = L1 | L2.  \*\*/  Logic**&** **operator|(**const Logic**&** L**);**  /\*\*  \* @brief Melakukan proses "xor" pada nilai logika (L1 ^ L2).  \* @param L1 - Objek Logika pertama.  \* @param L2 - Objek Logika kedua.  \* @return L' = L1 ^ L2.  \*\*/  Logic**&** **operator^(**const Logic**&** L**);**  /\*\*  \* @brief Mengubah nilai logika dalam bentuk string (TRUE atau FALSE) menjadi bentuk integer (1 & 0).  \* @brief Prekondisi : string terdefinisi (TRUE atau FALSE)  \* @param S - nilai logika dalam string.  \* @return i - nilai logika dalam integer.  \*\*/  int toInt**(**string**);**  /\*\*  \* @brief Mengubah nilai logika dalam bentuk bentuk integer (1 & 0) menjadi string (TRUE atau FALSE).  \* @brief Prekondisi : int terdefinisi (0 atau 1)  \* @param i - nilai logika dalam integer.  \* @return S - nilai logika dalam string.  \*\*/  string toString**(**int**);**  private**:**  int \_logic**;** // nilai logika dalam integer (0 dan 1).  **};**  #endif |

* 1. Locgic.cpp

|  |
| --- |
| /\*\*  \* @file Logic.cpp  \*\*/  #include <iostream>  #include "Logic.h"  **using** **namespace** std**;**  /\*\*  \* @brief Konstruktor kelas Logic.  \*  \*\*/  Logic**::**Logic**()** **:** Token**(**toString**(**0**))** **{**  \_logic **=** 0**;**  **}**  /\*\*  \* @brief Konstruktor kelas Logic dengan parameter string.  \* @param S - String berisi nilai logika (TRUE atau FALSE).  \*\*/  Logic**::**Logic**(**string s**)** **:** Token**(**s**)** **{**  \_logic **=** toInt**(**s**);**  **}**  /\*\*  \* @brief Konstruktor kelas Logic dengan parameter integer.  \* @param i - Integer berisi nilai logika (0 atau 1).  \*\*/  Logic**::**Logic**(**int i**)** **:** Token**(**toString**(**i**))** **{**  \_logic **=** i**;**  **}**  /\*\*  \* @brief Copy Konstruktor kelas Logic.  \* @param L - Variabel Logic masukan.  \* @return Li - Variabel Logic yang tercopy.  \*\*/  Logic**::**Logic**(**const Logic**&** L**)** **:** Token**(**toString**(**L**.**\_logic**))** **{**  **this->**\_logic **=** L**.**\_logic**;**  **}**  /\*\*  \* @brief Assignment kelas Logic.  \* @param L - Variabel Logic masukan.  \* @return Li - Variabel Logic yang terassign.  \*\*/  Logic**&** Logic**::operator=(**const Logic**&** L**)** **{**  **this->**\_logic **=** L**.**\_logic**;**  **this->**SetSymToken**(**L**.**GetSymToken**());**  **return** **\*this;**  **}**  /\*\*  \* @brief Destruktor kelas Logic.  \*\*/  Logic**::~**Logic**()** **{**  **}**  /\*\*  \* @brief Getter untuk mendapatkan nilai logic.  \* @return i - integer nilai logic (0 atau 1).  \*\*/  int Logic**::**GetLogic**()** **{**  **return** \_logic**;**  **}**  /\*\*  \* @brief Setter untuk mengubah nilai logic.  \* @param S - nilai logic yang akan dimasukkan.  \*\*/  void Logic**::**SetLogic**(**int i**)** **{**  \_logic **=** i**;**  **}**  /\*\*  \* @brief Melakukan proses "not" pada nilai logika (~L).  \* @param L - Objek Logika awal.  \* @return ~L - Objek Logika akhir.  \*\*/  Logic**&** Logic**::operator~()** **{**  **return** **\*(new** Logic**(!**\_logic**));**  **}**  /\*\*  \* @brief Melakukan proses "and" pada nilai logika (L1 & L2).  \* @param L1 - Objek Logika pertama.  \* @param L2 - Objek Logika kedua.  \* @return L' = L1 & L2.  \*\*/  Logic**&** Logic**::operator&(**const Logic**&** L**)** **{**  **return** **\*(new** Logic**(**\_logic **&&** L**.**\_logic**));**  **}**  /\*\*  \* @brief Melakukan proses "or" pada nilai logika (L1 | L2).  \* @param L1 - Objek Logika pertama.  \* @param L2 - Objek Logika kedua.  \* @return L' = L1 | L2.  \*\*/  Logic**&** Logic**::operator|(**const Logic**&** L**)** **{**  **return** **\*(new** Logic**(**\_logic **||** L**.**\_logic**));**  **}**  /\*\*  \* @brief Melakukan proses "xor" pada nilai logika (L1 ^ L2).  \* @param L1 - Objek Logika pertama.  \* @param L2 - Objek Logika kedua.  \* @return L' = L1 ^ L2.  \*\*/  Logic**&** Logic**::operator^(**const Logic**&** L**)** **{**  **return** **\*(new** Logic**(!(**\_logic**)** **!=** **!(**L**.**\_logic**)));**  **}**  /\*\*  \* @brief Mengubah nilai logika dalam bentuk string (TRUE atau FALSE) menjadi bentuk integer (1 & 0).  \* @param S - nilai logika dalam string.  \* @return i - nilai logika dalam integer.  \*\*/  int Logic**::**toInt**(**string s**)** **{**  **if(**s **==** "FALSE" **||** s **==** "false"**)**  **return** 0**;**  **return** 1**;**  **}**  /\*\*  \* @brief Mengubah nilai logika dalam bentuk bentuk integer (1 & 0) menjadi string (TRUE atau FALSE).  \* @param i - nilai logika dalam integer.  \* @return S - nilai logika dalam string.  \*\*/  string Logic**::**toString**(**int n**)** **{**  string ret**;**  **if(**n**)**  ret **=** string**(**"TRUE"**);**  **else**  ret **=** string**(**"FALSE"**);**  **return** ret**;**  **}** |

1. Package Stack
   1. Stack

|  |
| --- |
| /\*\*  \* @file stack.h  \*\*/  #ifndef \_\_STACK\_H\_\_  #define \_\_STACK\_H\_\_  #include <cstdlib>  #include <cassert>  /\*\*  \* @class stack  \* @author Wiwit Rifa'i (13513073)  \* @version 1.0  \*  \* @section Description  \* @brief kelas stack adalah implementasi vector yang ekuivalen stack STL C++  \*\*/  template**<** class T **>**  class stack **{**  public**:**  /\*\*  \* @brief Konstruktor kelas stack.  \*\*/  stack**();**  /\*\*  \* @brief Copy constructor kelas stack.  \* @param stack : yang akan di-copy.  \*\*/  stack**(**const stack**<**T**>&** **);**  /\*\*  \* @brief Destruktor kelas stack.  \*\*/  **~**stack**();**  /\*\*  \* @brief Operator assignment kelas stack.  \* @param stack : yang akan di-copy.  \*\*/  stack**<**T**>&** **operator=(**const stack**<**T**>&** **);**  /\*\*  \* @brief Mengembalikan predikat apakah stack kosong.  \* @return bool : predikat kosong stack.  \*\*/  bool empty**();**  /\*\*  \* @brief Mengembalikan ukuran stack.  \* @return int - ukuran stack.  \*\*/  int size**();**  /\*\*  \* @brief Mengembalikan elemen teratas stack  \* @return reference class T : elemen teratas  \*\*/  T**&** top**();**  /\*\*  \* @brief Menambahkan item pada paling atas stack  \* @param class T : item yang akan ditambahkan  \*\*/  void push**(**const T**&);**  /\*\*  \* @brief Menghapus item pada paling atas stack  \* @brief I. S. stack tidak kosong.  \*\*/  void pop**();**    private**:**  /\*\*  \* @class Stack\_Elmt  \* @brief kelas Stack\_Elmt bertanggung jawab pada satu buah elemen stack.  \* @brief kelas Stack\_Elmt merupakan inner class dari kelas stack dan hanya bisa dipakai oleh stack.  \*\*/  class Stack\_Elmt **{**  public **:**  /\*\*  \* @brief Konstruktor kelas Stack\_Elmt.  \* @param Stack\_Elmt \* \_tnext - berisi address elemen selanjutnya ( default = NULL ).  \*\*/  Stack\_Elmt**(**Stack\_Elmt **\*** \_tnext **=** **NULL);**  /\*\*  \* @brief Konstruktor kelas Stack\_Elmt dengan parameter.  \* @param T \_tval - nilai elemen yang akan diisi.  \* @param Stack\_Elmt \* \_tnext - berisi address elemen selanjutnya ( default = NULL ).  \*\*/  Stack\_Elmt**(**const T**&** \_tval**,** Stack\_Elmt **\*** \_tnext **=** **NULL);**  /\*\*  \* @brief Copy Constructor kelas stack  \* @param Stack\_Elmt& - yang akan di copy.  \*\*/  Stack\_Elmt**(**const Stack\_Elmt**&** **);**  /\*\*  \* @brief Destruktor kelas stack  \*\*/  **~**Stack\_Elmt**();**  /\*\*  \* @brief Operator assignment kelas stack  \* @param Stack\_Elmt& - yang akan di copy.  \*\*/  Stack\_Elmt**&** **operator=(**const Stack\_Elmt**&** li**);**  Stack\_Elmt **\*** \_next**;** // address elemen berikutnya  T \_val**;** // nilai elemen yang disimpan  **};**    Stack\_Elmt **\*** \_top**;**  int \_size**;**  **};**  /\*\*\*\*\* Implementasi class Stack\_Elmt \*\*\*\*\*/  template**<** class T **>**  stack**<**T**>::**Stack\_Elmt**::**Stack\_Elmt**(**Stack\_Elmt **\*** \_tnext**)** **{**  \_next **=** **NULL;**  **}**  template**<** class T **>**  stack**<**T**>::**Stack\_Elmt**::**Stack\_Elmt**(**const T**&** \_tval**,** Stack\_Elmt **\*** \_tnext**)** **{**  \_val **=** \_tval**;**  \_next **=** \_tnext**;**  **}**  template**<** class T **>**  stack**<**T**>::**Stack\_Elmt**::**Stack\_Elmt**(**const Stack\_Elmt**&** li**)** **{**  \_val **=** li**.**\_val**;**  \_next **=** li**.**\_next**;**  **}**  template**<** class T **>**  stack**<**T**>::**Stack\_Elmt**::~**Stack\_Elmt**()** **{**  **}**  template**<** class T **>**  typename stack**<**T**>::**Stack\_Elmt**&** stack**<**T**>::**Stack\_Elmt**::operator=(**const stack**<**T**>::**Stack\_Elmt**&** li**)** **{**  \_val **=** li**.**\_val**;**  \_next **=** li**.**\_next**;**  **return** **\*this;**  **}**  /\*\*\*\*\* implementasi class stack \*\*\*\*\*/  template**<** class T **>**  stack**<**T**>::**stack**()** **{**  \_size **=** 0**;**  \_top **=** **NULL;**  **}**  template**<** class T **>**  stack**<**T**>::**stack**(**const stack**<**T**>&** S**)** **{**  \_size **=** S**.**\_size**;**  Stack\_Elmt **\*** src\_temp **=** S**.**\_top**;**  **if(**src\_temp **!=** **NULL)** **{**  Stack\_Elmt **\*** des\_temp**;**  des\_temp **=** **new** Stack\_Elmt**(\*** src\_temp**);**  \_top **=** des\_temp**;**  src\_temp **=** src\_temp**->**\_next**;**  **while(**src\_temp **!=** **NULL)** **{**  des\_temp**->**\_next **=** **new** Stack\_Elmt**(\*** src\_temp**);**  des\_temp **=** des\_temp**->**\_next**;**  src\_temp **=** src\_temp**->**\_next**;**  **}**  **}**  **}**  template**<** class T **>**  stack**<**T**>::~**stack**()** **{**  /\* Delete last stack \*/  Stack\_Elmt **\*** del\_temp**;**  **while(**\_top **!=** **NULL)** **{**  del\_temp **=** \_top**;**  \_top **=** \_top**->**\_next**;**  **delete** del\_temp**;**  **}**  **}**  template**<** class T **>**  stack**<**T**>&** stack**<**T**>::operator=(**const stack**<**T**>&** S**)** **{**  /\* Delete last stack \*/  Stack\_Elmt **\*** del\_temp**;**  **while(**\_top **!=** **NULL)** **{**  del\_temp **=** \_top**;**  \_top **=** \_top**->**\_next**;**  **delete** del\_temp**;**  **}**  /\* Copy from stack S \*/  \_size **=** S**.**\_size**;**  Stack\_Elmt **\*** src\_temp **=** S**.**\_top**;**  **if(**src\_temp **!=** **NULL)** **{**  Stack\_Elmt **\*** des\_temp**;**  des\_temp **=** **new** Stack\_Elmt**(\*** src\_temp**);**  \_top **=** des\_temp**;**  src\_temp **=** src\_temp**->**\_next**;**  **while(**src\_temp **!=** **NULL)** **{**  des\_temp**->**\_next **=** **new** Stack\_Elmt**(\*** src\_temp**);**  des\_temp **=** des\_temp**->**\_next**;**  src\_temp **=** src\_temp**->**\_next**;**  **}**  **}**  **return** **\*this;**  **}**  template**<** class T **>**  bool stack**<**T**>::**empty**()** **{**  **return** \_size **==** 0**;**  **}**  template**<** class T **>**  int stack**<**T**>::**size**()** **{**  **return** \_size**;**  **}**  template**<** class T **>**  T**&** stack**<**T**>::**top**()** **{**  **return** \_top**->**\_val**;**  **}**  template**<** class T **>**  void stack**<**T**>::**push**(**const T**&** \_new**)** **{**  \_top **=** **new** Stack\_Elmt**(**\_new**,** \_top**);**  \_size**++;**  **}**  template**<** class T **>**  void stack**<**T**>::**pop**()** **{**  assert**(**\_top **!=** **NULL);**  Stack\_Elmt **\*** delTop **=** \_top**;**  \_top **=** \_top**->**\_next**;**  **delete** delTop**;**  \_size**--;**  **}**  #endif |

1. Package Vector
   1. Vector

|  |
| --- |
| #ifndef \_\_OWN\_VECTOR\_H\_  #define \_\_OWN\_VECTOR\_H\_  #include <algorithm>  #include <cassert>  /\*\*  \* @class vector  \* @author Luqman A. Siswanto (13513024)  \* @version 1.0  \*  \* @section Description  \* @brief Vector adalah implementasi vector yang ekuivalen vector STL C++  \*  \*\*/  template **<**class T**>**  class vector **{**  public**:**  /\*\*  \* @brief Konstruktor kelas vector.  \*\*/  vector**();**  /\*\*  \* @brief Copy constructor kelas vector.  \* @param vector : yang akan di-copy  \*\*/  vector**(**const vector**<**T**>&);**  /\*\*  \* @brief Operator assignment kelas vector.  \* @param vector : yang akan di-copy  \*\*/  vector**<**T**>&** **operator=** **(**vector**<**T**>);**  /\*\*  \* @brief Destruktor kelas vector.  \*\*/  **~**vector**();**    /\*\*  \* @brief Mengembalikan ukuran vector.  \* @return int - ukuran vector  \*\*/  int size**();**  /\*\*  \* @brief Mengembalikan ukuran maksimal vector saat ini sebelum alokasi kembali.  \* @return int - ukuran max vector  \*\*/  int max\_size**();**  /\*\*  \* @brief Mengubah ukuran vector  \* @param ukuran vector tujuan  \*\*/  void resize**(**int**);**  /\*\*  \* @brief Mengembalikan ukuran maksimal vector saat ini sebelum alokasi kembali.  \* @return int - ukuran max vector  \*\*/  int capacity**();**  /\*\*  \* @brief Mengembalikan predikat apakah vector kosong.  \* @return bool : predikat kosong vector  \*\*/  bool empty**();**    // element access  /\*\*  \* @brief Mengembalikan isi kontainer pada indeks tertentu  \* @param int - indeks vector  \* @return reference class T : item pada indeks tertentu  \*\*/  T**&** **operator[](**int**);**  /\*\*  \* @brief Mengembalikan isi kontainer pada indeks tertentu  \* @param int - indeks vector  \* @return reference class T : item pada indeks tertentu  \*\*/  T**&** at**(**int**);**  /\*\*  \* @brief Mengembalikan isi kontainer paling awal  \* @return reference class T : item pada indeks terawal  \*\*/  T**&** front**();**  /\*\*  \* @brief Mengembalikan isi kontainer paling akhir  \* @return reference class T : item pada indeks paling belakang  \*\*/  T**&** back**();**    // modifiers  /\*\*  \* @brief Memasukkan item pada akhir kontainer  \* @brief Bila vector penuh, maka mengalokasikan memori tambahan sebesar default Size  \* @param class T : item yang akan dimasukkan  \*\*/  void push\_back**(**T**);**  /\*\*  \* @brief Melepaskan item paling belakang vector  \* @brief I. S. vector tidak kosong  \*\*/  void pop\_back**();**  /\*\*  \* @brief Menukar kontainer vector beserta atributnya dengan vector lain  \* @param vector : vector yang akan ditukar dengan object this  \*\*/  void swap**(**vector**<**T**>&);**  /\*\*  \* @brief Mengkosongkan isi vector  \*\*/  void clear**();**    private**:**  static const int \_defaultSize**;** // konstanta ukuran default vector  T**\*** \_tab**;** // kontainer berupa array of class T  int \_maxSize**;** // ukuran maksimal vector saat ini  int \_size**;** // ukuran vector sekarang yang terisi  **};**  template **<**class T**>**  const int vector**<**T**>::**\_defaultSize **=** 100**;**  template **<**class T**>**  vector**<**T**>::**vector**()** **{**  \_tab **=** **new** T**[**\_defaultSize**];**  \_maxSize **=** \_defaultSize**;**  \_size **=** 0**;**  **}**  template **<**class T**>**  vector**<**T**>::**vector**(**const vector**<**T**>&** v**)** **{**  \_tab **=** **new** T**[**v**.**\_maxSize**];**  \_maxSize **=** v**.**\_maxSize**;**  \_size **=** v**.**\_size**;**  **}**  template **<**class T**>**  vector**<**T**>&** vector**<**T**>::operator=(**vector**<**T**>** v**)** **{**  **delete[]** \_tab**;**  \_tab **=** **new** T**[**v**.**\_maxSize**];**  \_maxSize **=** v**.**\_maxSize**;**  \_size **=** v**.**\_size**;**  **for(**int i **=** 0**;** i **<** \_maxSize**;** i**++)** **{**  \_tab**[**i**]** **=** v**.**\_tab**[**i**];**  **}**  **}**  template **<**class T**>**  vector**<**T**>::~**vector**()** **{**  **delete[]** \_tab**;**  **}**  template **<**class T**>**  int vector**<**T**>::**size**()** **{**  **return** \_size**;**  **}**  template **<**class T**>**  int vector**<**T**>::**max\_size**()** **{**  **return** \_maxSize**;**  **}**  template **<**class T**>**  void vector**<**T**>::**resize**(**int n**)** **{**  int block **=** n **/** \_defaultSize **+** 1**;**  **delete[]** \_tab**;**  \_tab **=** **new** T**[**block **\*** \_defaultSize**];**  \_size **=** 0**;**  \_maxSize **=** block **\*** \_defaultSize**;**  **}**  template **<**class T**>**  int vector**<**T**>::**capacity**()** **{**  **return** \_maxSize**;**  **}**  template **<**class T**>**  bool vector**<**T**>::**empty**()** **{**  **return** \_size **==** 0**;**  **}**  template **<**class T**>**  T**&** vector**<**T**>::operator[](**int i**)** **{**  assert**(**0 **<=** i **&&** i **<** \_size**);**  **return** \_tab**[**i**];**  **}**  template **<**class T**>**  T**&** vector**<**T**>::**at**(**int i**)** **{**  assert**(**0 **<=** i **&&** i **<** \_size**);**  **return** \_tab**[**i**];**  **}**  template **<**class T**>**  T**&** vector**<**T**>::**front**()** **{**  **return** \_tab**[**0**];**  **}**  template **<**class T**>**  T**&** vector**<**T**>::**back**()** **{**  **return** \_tab**[**\_size **-** 1**];**  **}**  template **<**class T**>**  void vector**<**T**>::**push\_back**(**T e**)** **{**  **if(**\_size **==** \_maxSize**)** **{**  T**\*** newTab **=** **new** T**[**\_maxSize **+** \_defaultSize**];**  **for(**int i **=** 0**;** i **<** \_maxSize**;** i**++)** **{**  newTab**[**i**]** **=** \_tab**[**i**];**  **}**  \_maxSize **+=** \_defaultSize**;**  **delete[]** \_tab**;**  \_tab **=** newTab**;**  **}**  \_tab**[**\_size**]** **=** e**;**  \_size**++;**  **}**  template **<**class T**>**  void vector**<**T**>::**pop\_back**()** **{**  assert**(**\_size **>** 0**);**  \_size**--;**  **}**  template **<**class T**>**  void vector**<**T**>::**swap**(**vector**<**T**>&** v**)** **{**  swap**(**v**.**\_tab**,** **this.**\_tab**);**  swap**(**v**.**\_maxSize**,** **this.**\_maxSize**);**  swap**(**v**.**\_size**,** **this.**\_size**);**  **}**  template **<**class T**>**  void vector**<**T**>::**clear**()** **{**  \_size **=** 0**;**  **}**  #endif |