Academic Social Network Simulator

1.0

Generated by Doxygen 1.8.3.1

Tue Nov 19 2013 22:38:05

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Namespace Documentation

4.1 readml Namespace Reference

Variables

- tuple R = open('graph.graphml','r')
- tuple f1 = open('adj.txt','w')
- tuple f2 = open('vmap.txt','w')
- tuple parts = line.split('"') f2.write(parts[1] + " " + parts[3] + " " + parts[5] + "\n") elif line[1:5] == 'edge': parts = line.split('"')

4.1.1 Variable Documentation

4.1.1.1 tuple readml.f1 = open('adj.txt','w')

Definition at line 2 of file readml.py.

4.1.1.2 tuple readml.f2 = open('vmap.txt','w')

Definition at line 3 of file readml.py.

4.1.1.3 tuple readml.parts = line.split('"') f2.write(parts[1] + " " + parts[3] + " " + parts[5] + "\n") elif line[1:5] == 'edge': parts = line.split('"')

Definition at line 8 of file readml.py.

4.1.1.4 tuple readml.R = open('graph.graphml','r')

Definition at line 1 of file readml.py.

Chapter 5

Class Documentation

5.1 Course Class Reference

`#include <containers.h>`

Public Member Functions

- Course (string, float, Department ∗)
  Course definition.
- ∼Course ()
- string getName () const
- float getFrequencyPerYear () const
- Department ∗ getDepartment () const

Private Attributes

- string name
- float frequencyPerYear
- Department ∗ department

5.1.1 Detailed Description

Definition at line 12 of file containers.h.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 Course::Course (string *_name*, float *_frequencyPerYear*, Department ∗ *_department*)

Course definition.

Constructor for the Course class

Definition at line 9 of file containers.cpp.

5.1.2.2 Course::∼Course ( )

Destructor for the Course class

Definition at line 16 of file containers.cpp.

5.1.3 Member Function Documentation

5.1.3.1 Department * Course::getDepartment ( ) const

Definition at line 21 of file containers.cpp.

5.1.3.2 float Course::getFrequencyPerYear ( ) const

Returns the expected number of times in a year that the given course is floated.

Definition at line 25 of file containers.cpp.

5.1.3.3 string Course::getName ( ) const

Definition at line 30 of file containers.cpp.

5.1.4 Member Data Documentation

5.1.4.1 Department* Course::department [private]

Definition at line 15 of file containers.h.

5.1.4.2 float Course::frequencyPerYear [private]

Definition at line 14 of file containers.h.

5.1.4.3 string Course::name [private]

Definition at line 13 of file containers.h.

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

- containers.h
- containers.cpp

5.2 Department Class Reference

`#include <containers.h>`

Public Member Functions

- Department (string, int, int, float, float, University *)
  Department definition.
- ~Department ()
- void addCourse (Course *)
- string getName () const
- int getNoOfStudentsPerYear () const
- int getNumFaculty () const
- University * getUniversity () const
- float getDeptCourses () const
- float getNonDeptCourses () const
- std::vector< Course * > getCourse () const

Private Attributes

- string name
- int noOfStudentsPerYear
- int numFaculty
- std::vector< Course * > course
- University * university
- float deptCourses
- float nonDeptCourses

5.2.1 Detailed Description

Definition at line 26 of file containers.h.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 Department::Department (string *_name,* int *num1,* int *num2,* float *num3,* float *num4,* University * *_univ*)

Department definition.

Constructor for the Department class

Definition at line 35 of file containers.cpp.

5.2.2.2 Department::~Department ( )

Destructor for the Department class

Definition at line 45 of file containers.cpp.

5.2.3 Member Function Documentation

5.2.3.1 void Department::addCourse (Course * *_course*)

Definition at line 50 of file containers.cpp.

5.2.3.2 std::vector< Course * > Department::getCourse ( ) const

Definition at line 53 of file containers.cpp.

5.2.3.3 float Department::getDeptCourses ( ) const

deptCourses is the expected number of courses that a student of this department does in his/her own department per semester.

Definition at line 57 of file containers.cpp.

5.2.3.4 string Department::getName ( ) const

Definition at line 62 of file containers.cpp.

5.2.3.5 float Department::getNonDeptCourses ( ) const

nonDeptCourses is the expected number of courses that a student of this department does in a department other than his/her own per semester.

Definition at line 66 of file containers.cpp.

5.2.3.6 int Department::getNoOfStudentsPerYear ( ) const

noOfStudentsPerYear is the number of students that are enrolled every year in this department.

Definition at line 71 of file containers.cpp.

5.2.3.7 int Department::getNumFaculty ( ) const

numFaculty is the number of faculty members in the department

Definition at line 76 of file containers.cpp.

5.2.3.8 University * Department::getUniversity ( ) const

Definition at line 81 of file containers.cpp.

5.2.4 Member Data Documentation

5.2.4.1 std::vector<Course*> Department::course [private]

Definition at line 30 of file containers.h.

5.2.4.2 float Department::deptCourses [private]

Definition at line 32 of file containers.h.

5.2.4.3 string Department::name [private]

Definition at line 27 of file containers.h.

5.2.4.4 float Department::nonDeptCourses [private]

Definition at line 33 of file containers.h.

5.2.4.5 int Department::noOfStudentsPerYear [private]

Definition at line 28 of file containers.h.

5.2.4.6 int Department::numFaculty [private]

Definition at line 29 of file containers.h.

5.2.4.7 University* Department::university [private]

Definition at line 31 of file containers.h.

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

- containers.h
- containers.cpp

5.3 Faculty Class Reference

`#include <faculty.h>`

Public Member Functions

- Faculty (int, string, Department *, string, std::vector< string >)
  Faculty definition.
- ~Faculty ()
- Department * getDepartment () const
- int getFacultyId () const
- string getHouse () const
- std::vector< string > getInterest () const
- string getName () const

Public Attributes

- string home
- std::vector< string > interest
- string university
- string dept
- std::vector< string > courses

Private Attributes

- int facultyId
- string name
- Department * department

5.3.1 Detailed Description

Definition at line 10 of file faculty.h.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 Faculty::Faculty (int *_facultyId,* string *_name,* Department * *_department,* string *_house,* std::vector< string > *_interest*)

Faculty definition.

Constructor for Faculty class

Definition at line 15 of file faculty.cpp.

5.3.2.2 Faculty::~Faculty ( )

Definition at line 24 of file faculty.cpp.

5.3.3 Member Function Documentation

5.3.3.1 Department * Faculty::getDepartment ( ) const

Definition at line 28 of file faculty.cpp.

5.3.3.2 int Faculty::getFacultyId ( ) const

Definition at line 31 of file faculty.cpp.

5.3.3.3 string Faculty::getHouse ( ) const

Definition at line 34 of file faculty.cpp.

5.3.3.4 std::vector< string > Faculty::getInterest ( ) const

Definition at line 37 of file faculty.cpp.

5.3.3.5 string Faculty::getName ( ) const

Definition at line 40 of file faculty.cpp.

5.3.4 Member Data Documentation

5.3.4.1 std::vector<string> Faculty::courses

Definition at line 19 of file faculty.h.

5.3.4.2 Department* Faculty::department [private]

Definition at line 13 of file faculty.h.

5.3.4.3 string Faculty::dept

Definition at line 18 of file faculty.h.

5.3.4.4 int Faculty::facultyId [private]

Definition at line 11 of file faculty.h.

5.3.4.5 string Faculty::home

Definition at line 15 of file faculty.h.

5.3.4.6 std::vector<string> Faculty::interest

Definition at line 16 of file faculty.h.

5.3.4.7 string Faculty::name [private]

Definition at line 12 of file faculty.h.

5.3.4.8 string Faculty::university

Definition at line 17 of file faculty.h.

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

- faculty.h
- faculty.cpp

5.4 ms Struct Reference

Public Member Functions

- ms ()

Public Attributes

- long int type

  The structure for a message that is sent back and forth among the two processes TimeKeeper and Generator. Each message has a tag that is used to differentiate it from system-generated messages present in the message queue or messages meant for use by other programs.
- int Time
- int tid

5.4.1 Detailed Description

Definition at line 37 of file main.cpp.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 ms::ms ( ) [inline]

Definition at line 43 of file main.cpp.

5.4.3 Member Data Documentation

5.4.3.1 int ms::tid

Definition at line 42 of file main.cpp.

5.4.3.2 int ms::Time

Definition at line 41 of file main.cpp.

5.4.3.3 long int ms::type

The structure for a message that is sent back and forth among the two processes TimeKeeper and Generator. Each message has a tag that is used to differentiate it from system-generated messages present in the message queue or messages meant for use by other programs.

Definition at line 40 of file main.cpp.

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

- main.cpp

5.5 mycomparison Class Reference

Public Member Functions

- mycomparison (const bool &revparam=false)
- bool operator() (const PI &lhs, const PI &rhs) const

Private Attributes

- bool reverse

5.5.1 Detailed Description

Definition at line 48 of file main.cpp.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 mycomparison::mycomparison (const bool & *revparam* = `false`) [inline]

The priority operation for the priority queue

Definition at line 51 of file main.cpp.

5.5.3 Member Function Documentation

5.5.3.1 bool mycomparison::operator() (const PI & *lhs,* const PI & *rhs*) const [inline]

Definition at line 55 of file main.cpp.

5.5.4 Member Data Documentation

5.5.4.1 bool mycomparison::reverse [private]

Definition at line 49 of file main.cpp.

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

- main.cpp

5.6 Student Class Reference

`#include <student.h>`

Public Member Functions

- Student (int, string, Department *, string, std::vector< string >)
  Student definition.
- ~Student ()
- Department * getDepartment () const
- int getStudentId () const
- string getHostel () const
- std::vector< string > getInterest () const
- string getName () const

Public Attributes

- string university
- string home
- std::vector< string > interest
- std::vector< string > courses
- string dept
- int year

Private Attributes

- int studentId
- string name
- Department * department

5.6.1 Detailed Description

Definition at line 10 of file student.h.

5.6.2 Constructor & Destructor Documentation

5.6.2.1 Student::Student (int *_studentId,* string *_name,* Department * *_department,* string *_hostel,* std::vector< string > *_interest*)

Student definition.

Definition at line 15 of file student.cpp.

5.6.2.2 Student::~Student ( )

Definition at line 24 of file student.cpp.

5.6.3 Member Function Documentation

5.6.3.1 Department * Student::getDepartment ( ) const

Definition at line 29 of file student.cpp.

5.6.3.2 string Student::getHostel ( ) const

Definition at line 35 of file student.cpp.

5.6.3.3 std::vector< string > Student::getInterest ( ) const

Definition at line 38 of file student.cpp.

5.6.3.4 string Student::getName ( ) const

Definition at line 41 of file student.cpp.

5.6.3.5 int Student::getStudentId ( ) const

Definition at line 32 of file student.cpp.

5.6.4 Member Data Documentation

5.6.4.1 std::vector<string> Student::courses

Definition at line 18 of file student.h.

5.6.4.2 Department* Student::department [private]

Definition at line 13 of file student.h.

5.6.4.3 string Student::dept

Definition at line 19 of file student.h.

5.6.4.4 string Student::home

Definition at line 16 of file student.h.

5.6.4.5 std::vector<string> Student::interest

Definition at line 17 of file student.h.

5.6.4.6 string Student::name [private]

Definition at line 12 of file student.h.

5.6.4.7 int Student::studentId [private]

Definition at line 11 of file student.h.

5.6.4.8 string Student::university

Definition at line 15 of file student.h.

5.6.4.9 int Student::year

Definition at line 20 of file student.h.

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

- student.h
- student.cpp

5.7 University Class Reference

`#include <containers.h>`

Public Member Functions

- University (string)

  University definition.
- ~University ()
- void addInterest (string, float)
- void addHostel (string)
- void addHouse (string)
- void addDepartment (Department *)
- void setFriendshipRate (float)
- void setFriendliness (float)
- void setOpenness (float)
- string getName () const
- std::vector< Department * > getDepartment () const
- std::vector< pair< string, float > > getInterest () const
- std::vector< string > getHostel () const
- std::vector< string > getHouse () const
- float getFriendshipRate () const
- float getFriendliness () const
- float getOpenness () const

Public Attributes

- int sId

Private Attributes

- string name
- std::vector< Department * > department
- std::vector< pair< string, float > > interest
- std::vector< string > hostel
- std::vector< string > house
- float friendshipRate
- float friendliness
- float openness

5.7.1 Detailed Description

Definition at line 48 of file containers.h.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 University::University (string *_name*)

University definition.

Constructor for the University class

Definition at line 86 of file containers.cpp.

5.7.2.2 University::~University ( )

Destructor for the University class

Definition at line 91 of file containers.cpp.

5.7.3 Member Function Documentation

5.7.3.1 void University::addDepartment (Department * *_department*)

Definition at line 100 of file containers.cpp.

5.7.3.2 void University::addHostel (string *_hostel*)

Definition at line 114 of file containers.cpp.

5.7.3.3 void University::addHouse (string *_house*)

Definition at line 118 of file containers.cpp.

5.7.3.4 void University::addInterest (string *_name*, float *_popularity*)

Definition at line 122 of file containers.cpp.

5.7.3.5 std::vector< Department * > University::getDepartment ( ) const

Definition at line 131 of file containers.cpp.

5.7.3.6 float University::getFriendliness ( ) const

Definition at line 135 of file containers.cpp.

5.7.3.7 float University::getFriendshipRate ( ) const

Definition at line 139 of file containers.cpp.

5.7.3.8 std::vector< string > University::getHostel ( ) const

The student residences

Definition at line 143 of file containers.cpp.

5.7.3.9 std::vector< string > University::getHouse ( ) const

The faculty residences

Definition at line 148 of file containers.cpp.

5.7.3.10 std::vector< pair< string, float > > University::getInterest ( ) const

Definition at line 153 of file containers.cpp.

5.7.3.11 string University::getName ( ) const

Definition at line 157 of file containers.cpp.

5.7.3.12 float University::getOpenness ( ) const

Definition at line 161 of file containers.cpp.

5.7.3.13 void University::setFriendliness (float *friendliness*)

friendliness is the probability that a person from this university accepts a friend request sent to him/her

Definition at line 104 of file containers.cpp.

5.7.3.14 void University::setFriendshipRate (float *friendshipRate*)

friendshipRate is the total number of friend requests generated every minute by people of this university

Definition at line 109 of file containers.cpp.

5.7.3.15 void University::setOpenness (float *openness*)

openness is the probability that a person from this university will attempt to make friends with someone with whom he/she does not share a common interest, the same hostel, the same department or a common course.

Definition at line 126 of file containers.cpp.

5.7.4 Member Data Documentation

5.7.4.1 std::vector<Department*> University::department [private]

Definition at line 50 of file containers.h.

5.7.4.2 float University::friendliness [private]

Definition at line 55 of file containers.h.

5.7.4.3 float University::friendshipRate [private]

Definition at line 54 of file containers.h.

5.7.4.4 std::vector<string> University::hostel [private]

Definition at line 52 of file containers.h.

5.7.4.5 std::vector<string> University::house [private]

Definition at line 53 of file containers.h.

5.7.4.6 std::vector<pair<string, float> > University::interest [private]

Definition at line 51 of file containers.h.

5.7.4.7 string University::name [private]

Definition at line 49 of file containers.h.

5.7.4.8 float University::openness [private]

Definition at line 56 of file containers.h.

5.7.4.9 int University::sId

Definition at line 67 of file containers.h.

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

- containers.h
- containers.cpp

Chapter 6

File Documentation

6.1 containers.cpp File Reference

```
#include <string>
#include <map>
#include <vector>
#include "containers.h"
```

6.2 containers.h File Reference

```
#include <string>
#include <map>
#include <vector>
```

Classes

- class Course
- class Department
- class University

6.3 courses.cpp File Reference

```
#include <string>
#include <vector>
#include <map>
#include <cstdlib>
#include <algorithm>
#include "containers.h"
#include "globals.h"
#include "environment.h"
```

Functions

- void addRandomDepCourse (Student *, Department *)

- void clearCourses ()
- void generateCourses (std::vector< University * > _universities, int _courseRandom)
- void printCourses (std::vector< Department * > _departments)

Variables

- map< Department *, std::vector < Course * > > departmentCourseMap

6.3.1 Function Documentation

6.3.1.1 void addRandomDepCourse (Student * _student, Department * _department)

The input list of students is assigned courses randomly from the input list of departments based on the department and non-department requirements of the students.

Definition at line 106 of file courses.cpp.

6.3.1.2 void clearCourses ( )

This function deregisters the students from the courses they registered for in the previous semester.

Definition at line 130 of file courses.cpp.

6.3.1.3 void generateCourses (std::vector< University * > _universities, int _courseRandom)

```
Based on the value of a seeded random int specified in the initialisation of the simulation, a set of courses
```

Whether a particular course is floated or not depends on the frequencyPerYear of that course. A frequencyPerYear of 1.3 for a course means that the course is floated atleast once, and floated twice with a probability of 0.3.

Each course is then taken up by prospective students from the same and other departments. A student randomly chooses among the floated courses those which he/she intends to take up. This randomness is based on the seed provided in the variable _courseRandom during initialisation of the environment. Other departments are also randomly chosen, from which courses are also chosen randomly.

The faculty members are assigned courses in a Round-Robin fashion.

Definition at line 14 of file courses.cpp.

6.3.1.4 void printCourses (std::vector< Department * > _departments)

Definition at line 121 of file courses.cpp.

6.3.2 Variable Documentation

6.3.2.1 map<Department*, std::vector<Course*> > departmentCourseMap

Definition at line 10 of file courses.cpp.

6.4 courses.h File Reference

```
#include <string>
#include <vector>
#include <map>
#include "containers.h"
#include "globals.h"
```

Functions

- void generateCourses (std::vector< University * >, int)
- void printCourses (std::vector< Department * >)

6.4.1 Function Documentation

6.4.1.1 void generateCourses (std::vector< University * > , int)

```
Based on the value of a seeded random int specified in the initialisation of the simulation, a set of courses
```

Whether a particular course is floated or not depends on the frequencyPerYear of that course. A frequencyPerYear of 1.3 for a course means that the course is floated atleast once, and floated twice with a probability of 0.3.

Each course is then taken up by prospective students from the same and other departments. A student randomly chooses among the floated courses those which he/she intends to take up. This randomness is based on the seed provided in the variable _courseRandom during initialisation of the environment. Other departments are also randomly chosen, from which courses are also chosen randomly.

The faculty members are assigned courses in a Round-Robin fashion.

Definition at line 14 of file courses.cpp.

6.4.1.2 void printCourses (std::vector< Department * >)

Definition at line 121 of file courses.cpp.

6.5 environment.cpp File Reference

```
#include <vector>
#include "containers.h"
#include "parser.h"
#include "globals.h"
#include "logger.h"
```

Functions

- void setEnvironment (char *filename)

6.5.1 Function Documentation

6.5.1.1 void setEnvironment (char * *filename*)

This function sets the value of various initialisation parameters and random seeds needed to run the simulation. There is a different random seed associated with the generation of faculty, students, courses and friendship activity.

Definition at line 6 of file environment.cpp.

6.6 environment.h File Reference

```
#include <vector>
#include "containers.h"
#include "parser.h"
#include "globals.h"
```

Functions

- void setEnvironment (char *)

6.6.1 Function Documentation

6.6.1.1 void setEnvironment (char *)

This function sets the value of various initialisation parameters and random seeds needed to run the simulation. There is a different random seed associated with the generation of faculty, students, courses and friendship activity.

Definition at line 6 of file environment.cpp.

6.7 faculty.cpp File Reference

```
#include <string>
#include <map>
#include <vector>
#include <cstdlib>
#include <stdio.h>
#include <iostream>
#include "faculty.h"
#include "globals.h"
#include "namesAccessor.h"
```

Macros

- #define mp make_pair
- #define pb push_back

Functions

- void generateFaculties (std::vector< University * > _universities, int _facultySeed)
- void generateFaculty (University *_university, Department *_department, int _facultyId)
- void printFaculties ()

6.7.1 Macro Definition Documentation

6.7.1.1 #define mp make_pair

Definition at line 10 of file faculty.cpp.

6.7.1.2 #define pb push_back

Definition at line 11 of file faculty.cpp.

6.7.2 Function Documentation

6.7.2.1 void generateFaculties (std::vector< University * > *_universities,* int *_facultySeed*)

A list of the required number of faculty members is generated by this function, by randomly picking names from an associated names database.

Definition at line 44 of file faculty.cpp.

6.7.2.2 void generateFaculty (University * *_university,* Department * *_department,* int *_facultyId*)

The faculty members, randomly generated in generateFaculties, are randomly assigned to a unniversity and department. They are also associated with a unique and randomly generated ID, facultyID.

Definition at line 62 of file faculty.cpp.

6.7.2.3 void printFaculties ( )

Definition at line 91 of file faculty.cpp.

6.8 faculty.h File Reference

```
#include <string>
#include <map>
#include <vector>
#include "containers.h"
```

Classes

- class Faculty

Functions

- void generateFaculties (std::vector< University * >, int)
- void generateFaculty (University *, Department *, int)
- void printFaculties ()

6.8.1 Function Documentation

6.8.1.1 void generateFaculties (std::vector< University * > , int)

A list of the required number of faculty members is generated by this function, by randomly picking names from an associated names database.

Definition at line 44 of file faculty.cpp.

6.8.1.2 void generateFaculty (University * , Department * , int)

The faculty members, randomly generated in generateFaculties, are randomly assigned to a unniversity and department. They are also associated with a unique and randomly generated ID, facultyID.

Definition at line 62 of file faculty.cpp.

6.8.1.3 void printFaculties ( )

Definition at line 91 of file faculty.cpp.

6.9 friend.cpp File Reference

```
#include <string>
#include <vector>
#include <map>
#include <cstdlib>
#include <algorithm>
#include "containers.h"
#include "globals.h"
#include "environment.h"
#include "friend.h"
#include "logger.h"
```

Functions

- PERSON randomPerson (std::vector< PERSON > personList)
- void handleRequest (PERSON _randomSender, PERSON _randomReceiver)
- void generateRequest (int _friendSeed)

6.9.1 Function Documentation

6.9.1.1 void generateRequest (int *_friendSeed*)

A random person is selected. Another random person is selected with the probability out_probability, and a person sharing a common interest/course/hostel/department is selected with probability 1-out_probability. The request is then accepted with the parameters of reciprocity and openness. This function implements the friendship activities as described in the documentation.

Definition at line 15 of file friend.cpp.

6.9.1.2 void handleRequest (PERSON *_randomSender,* PERSON *_randomReceiver*)

This function processes the information that a new accepted friend request

Definition at line 181 of file friend.cpp.

6.9.1.3 PERSON randomPerson (std::vector< PERSON > *personList*)

picks a random person from the list

Definition at line 170 of file friend.cpp.

6.10 friend.h File Reference

```
#include <string>
#include <vector>
#include <map>
#include <cstdlib>
#include <algorithm>
#include "containers.h"
#include "globals.h"
#include "environment.h"
```

Functions

- void generateRequest (int)

6.10.1 Function Documentation

6.10.1.1 void generateRequest (int)

A random person is selected. Another random person is selected with the probability out_probability, and a person sharing a common interest/course/hostel/department is selected with probability 1-out_probability. The request is then accepted with the parameters of reciprocity and openness. This function implements the friendship activities as described in the documentation.

Definition at line 15 of file friend.cpp.

6.11 globals.cpp File Reference

```
#include <vector>
#include <map>
#include <string>
#include <set>
#include <fstream>
#include "containers.h"
#include "faculty.h"
#include "student.h"
```

Macros

- #define PERSON pair<string,int>
- #define DEPARTMENT pair<string,string>
- #define INTEREST string

- #define COURSE pair<string,string>
- #define HOME pair<string,string>

Variables

- std::vector< University * > universities
- std::vector< Course * > courses
- std::vector< Department * > departments
- map< string, Department * > stringToDepartment
- int facultyRandom
- int studentRandom
- int courseRandom
- int friendRandom
- int roundRobinCounter = 0
- std::vector< Faculty * > faculties
- std::vector< Student * > students
- std::vector< string > namesList
- pthread_mutex_t mutex
- map< PERSON, set< PERSON > > adj
- map< COURSE, vector< PERSON > > courseMap
- map< DEPARTMENT, vector< PERSON > > departmentMap
- map< INTEREST, vector< PERSON > > interestMap
- map< HOME, vector< PERSON > > homeMap
- map< PERSON, Faculty * > facultyMap
- map< PERSON, Student * > studentMap
- fstream fl

6.11.1 Macro Definition Documentation

6.11.1.1 #define COURSE pair<string,string>

Definition at line 26 of file globals.cpp.

6.11.1.2 #define DEPARTMENT pair<string,string>

Definition at line 24 of file globals.cpp.

6.11.1.3 #define HOME pair<string,string>

Definition at line 27 of file globals.cpp.

6.11.1.4 #define INTEREST string

Definition at line 25 of file globals.cpp.

6.11.1.5 #define PERSON pair<string,int>

Definition at line 23 of file globals.cpp.

6.11.2 Variable Documentation

6.11.2.1 map< PERSON , set<PERSON > > adj

Definition at line 28 of file globals.cpp.

6.11.2.2 map< COURSE , vector<PERSON> > courseMap

Definition at line 30 of file globals.cpp.

6.11.2.3 int courseRandom

Definition at line 16 of file globals.cpp.

6.11.2.4 std::vector<Course*> courses

Definition at line 11 of file globals.cpp.

6.11.2.5 map< DEPARTMENT , vector<PERSON> > departmentMap

Definition at line 31 of file globals.cpp.

6.11.2.6 std::vector<Department*> departments

Definition at line 12 of file globals.cpp.

6.11.2.7 std::vector<Faculty*> faculties

Definition at line 19 of file globals.cpp.

6.11.2.8 map< PERSON , Faculty*> facultyMap

Definition at line 35 of file globals.cpp.

6.11.2.9 int facultyRandom

Definition at line 14 of file globals.cpp.

6.11.2.10 fstream fl

Definition at line 38 of file globals.cpp.

6.11.2.11 int friendRandom

Definition at line 17 of file globals.cpp.

6.11.2.12 map< HOME , vector<PERSON> > homeMap

Definition at line 33 of file globals.cpp.

6.11.2.13 map< INTEREST , vector<PERSON> > interestMap

Definition at line 32 of file globals.cpp.

6.11.2.14 pthread_mutex_t mutex

Definition at line 22 of file globals.cpp.

6.11.2.15 std::vector<string> namesList

Definition at line 21 of file globals.cpp.

6.11.2.16 int roundRobinCounter = 0

Definition at line 18 of file globals.cpp.

6.11.2.17 map<string,Department*> stringToDepartment

Definition at line 13 of file globals.cpp.

6.11.2.18 map< PERSON , Student*> studentMap

Definition at line 36 of file globals.cpp.

6.11.2.19 int studentRandom

Definition at line 15 of file globals.cpp.

6.11.2.20 std::vector<Student*> students

Definition at line 20 of file globals.cpp.

6.11.2.21 std::vector<University*> universities

Definition at line 10 of file globals.cpp.

6.12 globals.h File Reference

```
#include <vector>
#include <map>
#include <set>
#include <string>
#include <fstream>
#include "containers.h"
#include "faculty.h"
#include "student.h"
```

Macros

- #define PERSON pair<string,int>
- #define DEPARTMENT pair<string,string>
- #define INTEREST string
- #define COURSE pair<string,string>
- #define HOME pair<string,string>

Variables

- std::vector< University * > universities
- std::vector< Course * > courses
- std::vector< Department * > departments
- map< string, Department * > stringToDepartment
- int facultyRandom
- int studentRandom
- int courseRandom
- int friendRandom
- int roundRobinCounter
- std::vector< Faculty * > faculties
- std::vector< Student * > students
- std::vector< string > namesList
- pthread_mutex_t mutex
- map< PERSON, set< PERSON > > adj
- map< COURSE, vector< PERSON > > courseMap
- map< DEPARTMENT, vector< PERSON > > departmentMap
- map< INTEREST, vector< PERSON > > interestMap
- map< HOME, vector< PERSON > > homeMap
- map< PERSON, Faculty * > facultyMap
- map< PERSON, Student * > studentMap
- fstream fl

6.12.1 Macro Definition Documentation

6.12.1.1 #define COURSE pair<string,string>

Definition at line 27 of file globals.h.

6.12.1.2 #define DEPARTMENT pair<string,string>

Definition at line 25 of file globals.h.

6.12.1.3 #define HOME pair<string,string>

Definition at line 28 of file globals.h.

6.12.1.4 #define INTEREST string

Definition at line 26 of file globals.h.

6.12.1.5 #define PERSON pair<string,int>

Definition at line 24 of file globals.h.

6.12.2 Variable Documentation

6.12.2.1 map< PERSON , set<PERSON > > adj

Definition at line 28 of file globals.cpp.

6.12.2.2 map< COURSE , vector<PERSON> > courseMap

Definition at line 30 of file globals.cpp.

6.12.2.3 int courseRandom

Definition at line 16 of file globals.cpp.

6.12.2.4 std::vector<Course*> courses

Definition at line 11 of file globals.cpp.

6.12.2.5 map< DEPARTMENT , vector<PERSON> > departmentMap

Definition at line 31 of file globals.cpp.

6.12.2.6 std::vector<Department*> departments

Definition at line 12 of file globals.cpp.

6.12.2.7 std::vector<Faculty*> faculties

Definition at line 19 of file globals.cpp.

6.12.2.8 map< PERSON , Faculty*> facultyMap

Definition at line 35 of file globals.cpp.

6.12.2.9 int facultyRandom

Definition at line 14 of file globals.cpp.

6.12.2.10 fstream fl

Definition at line 38 of file globals.cpp.

6.12.2.11 int friendRandom

Definition at line 17 of file globals.cpp.

6.12.2.12 map< HOME , vector<PERSON> > homeMap

Definition at line 33 of file globals.cpp.

6.12.2.13 map< INTEREST , vector<PERSON> > interestMap

Definition at line 32 of file globals.cpp.

6.12.2.14 pthread_mutex_t mutex

Definition at line 22 of file globals.cpp.

6.12.2.15 std::vector<string> namesList

Definition at line 21 of file globals.cpp.

6.12.2.16 int roundRobinCounter

Definition at line 18 of file globals.cpp.

6.12.2.17 map<string,Department*> stringToDepartment

Definition at line 13 of file globals.cpp.

6.12.2.18 map< PERSON , Student*> studentMap

Definition at line 36 of file globals.cpp.

6.12.2.19 int studentRandom

Definition at line 15 of file globals.cpp.

6.12.2.20 std::vector<Student*> students

Definition at line 20 of file globals.cpp.

6.12.2.21 std::vector<University*> universities

Definition at line 10 of file globals.cpp.

6.13 graphml.cpp File Reference

```
#include <iostream>
#include <vector>
#include <set>
#include <map>
#include <string>
#include <fstream>
#include "globals.h"
#include "graphml.h"
#include "student.h"
#include "faculty.h"
#include "containers.h"
#include "logger.h"
```

Macros

- #define mp make_pair
- #define pb push_back
- #define PERSON pair<string,int>

Functions

- void extendMap ()
- int getNode (PERSON x)
- void writeNodes ()
- void writeEdges ()
- void writeHeader ()
- void writeFooter ()
- void makeGraph ()

Variables

- static int id = 0
- vector< PERSON > arr
- map< PERSON, int > idx
- vector< pair< int, int > > edge
- ofstream f

6.13.1 Macro Definition Documentation

6.13.1.1 #define mp make_pair

Definition at line 15 of file graphml.cpp.

6.13.1.2 #define pb push_back

Definition at line 16 of file graphml.cpp.

6.13.1.3 #define PERSON pair<string,int>

Definition at line 17 of file graphml.cpp.

6.13.2 Function Documentation

6.13.2.1 void extendMap ( )

This function adds zero-degree vertices to the graph

Definition at line 25 of file graphml.cpp.

6.13.2.2 int getNode (PERSON *x*)

Returns the address of given person in the graph

Definition at line 35 of file graphml.cpp.

6.13.2.3 void makeGraph ( )

Iterates over the list of all people and all friend-pairs and calls the functions to write this information in the GraphML file format.

Definition at line 73 of file graphml.cpp.

6.13.2.4 void writeEdges ( )

Streams the information about graph edges into the GraphML file format

Definition at line 52 of file graphml.cpp.

6.13.2.5 void writeFooter ( )

Definition at line 69 of file graphml.cpp.

6.13.2.6 void writeHeader ( )

Streams the required header to the GraphML file

Definition at line 60 of file graphml.cpp.

6.13.2.7 void writeNodes ( )

Streams the information about graph edges into the GraphML file format

Definition at line 43 of file graphml.cpp.

6.13.3 Variable Documentation

6.13.3.1 vector<PERSON > arr

Definition at line 20 of file graphml.cpp.

6.13.3.2 vector<pair<int, int> > edge

Definition at line 22 of file graphml.cpp.

6.13.3.3 ofstream f

Definition at line 23 of file graphml.cpp.

6.13.3.4 int id = 0 [static]

Definition at line 19 of file graphml.cpp.

6.13.3.5 map<PERSON, int> idx

Definition at line 21 of file graphml.cpp.

6.14 graphml.h File Reference

Functions

- void makeGraph ()

6.14.1 Function Documentation

6.14.1.1 void makeGraph ( )

Iterates over the list of all people and all friend-pairs and calls the functions to write this information in the GraphML file format.

Definition at line 73 of file graphml.cpp.

6.15 logger.cpp File Reference

```
#include <fstream>
#include "logger.h"
#include "globals.h"
```

Functions

- void startlog ()
- void log (string s)
- void endlog ()
- void clearlog ()

6.15.1 Function Documentation

6.15.1.1 void clearlog ( )

Clears the log file

Definition at line 17 of file logger.cpp.

6.15.1.2 void endlog ( )

Closes the log file

Definition at line 13 of file logger.cpp.

6.15.1.3 void log (string *s*)

Puts an entry into the log

Definition at line 9 of file logger.cpp.

6.15.1.4 void startlog ( )

Initialises a log of the social network activity

Definition at line 5 of file logger.cpp.

6.16 logger.h File Reference

```
#include <fstream>
```

Functions

- void log (string s)
- void clearlog ()
- void startlog ()
- void endlog ()

6.16.1 Function Documentation

6.16.1.1 void clearlog ( )

Clears the log file

Definition at line 17 of file logger.cpp.

6.16.1.2 void endlog ( )

Closes the log file

Definition at line 13 of file logger.cpp.

6.16.1.3 void log (string *s*)

Puts an entry into the log

Definition at line 9 of file logger.cpp.

6.16.1.4 void startlog ( )

Initialises a log of the social network activity

Definition at line 5 of file logger.cpp.

6.17 main.cpp File Reference

```
#include <stdio.h>
```

```
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <sys/stat.h>
#include <string.h>
#include <signal.h>
#include <sstream>
#include <queue>
#include <iostream>
#include "globals.h"
#include "environment.h"
#include "containers.h"
#include "faculty.h"
#include "friend.h"
#include "student.h"
#include "logger.h"
#include "parser.h"
#include "graphml.h"
#include "courses.h"
```

Classes

- struct ms
- class mycomparison

Macros

- #define YEAR 31622400
- #define HOUR 3600
- #define SEC 1
- #define DAY 24∗HOUR
- #define WEEK 7∗DAY
- #define SEM 15811200
- #define MIN 60
- #define mp make_pair
- #define pb push_back

Typedefs

- typedef pair< int, int > PI

Functions

- void ∗ c_thread (void ∗)
- void ∗ s_thread (void ∗)
- void ∗ f_thread (void ∗)
- void sendRequest (int Time, int tid)
- ms recAlarm (int)
- void Timekeeper ()
- int main (int argc, char ∗argv[])

Variables

- int MAXTIME = 14∗YEAR
- int ts
- int tc
- int tf
- int gt
- int tg
- priority_queue< PI, vector< PI >
  , greater< PI > > q
- int currentTime
- float rrq

6.17.1 Macro Definition Documentation

6.17.1.1 #define DAY 24∗HOUR

Definition at line 28 of file main.cpp.

6.17.1.2 #define HOUR 3600

Definition at line 26 of file main.cpp.

6.17.1.3 #define MIN 60

Definition at line 31 of file main.cpp.

6.17.1.4 #define mp make_pair

Definition at line 33 of file main.cpp.

6.17.1.5 #define pb push_back

Definition at line 34 of file main.cpp.

6.17.1.6 #define SEC 1

Definition at line 27 of file main.cpp.

6.17.1.7 #define SEM 15811200

Definition at line 30 of file main.cpp.

6.17.1.8 #define WEEK 7∗DAY

Definition at line 29 of file main.cpp.

6.17.1.9 #define YEAR 31622400

Definition at line 25 of file main.cpp.

6.17.2 Typedef Documentation

6.17.2.1 typedef pair<int, int> PI

Definition at line 47 of file main.cpp.

6.17.3 Function Documentation

6.17.3.1 void* c_thread (void *)

6.17.3.2 void* f_thread (void *)

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

This process bifurcates into the two processes Timekeeper and Generator

Definition at line 73 of file main.cpp.

6.17.3.4 ms recAlarm (int)

6.17.3.5 void* s_thread (void *)

6.17.3.6 void sendRequest (int *Time,* int *tid*)

A thread for GenerateStudents function

A thread for GenerateCourses function

A thread for GenerateFriends function

Definition at line 153 of file main.cpp.

6.17.3.7 void Timekeeper ( )

6.17.4 Variable Documentation

6.17.4.1 int currentTime

Definition at line 65 of file main.cpp.

6.17.4.2 int gt

Definition at line 62 of file main.cpp.

6.17.4.3 int MAXTIME = 14*YEAR

Definition at line 36 of file main.cpp.

6.17.4.4 priority_queue<PI, vector<PI>, greater<PI> > q

Definition at line 63 of file main.cpp.

6.17.4.5 float rrq

Definition at line 66 of file main.cpp.

6.17.4.6 int tc

Definition at line 62 of file main.cpp.

6.17.4.7 int tf

Definition at line 62 of file main.cpp.

6.17.4.8 int tg

Definition at line 62 of file main.cpp.

6.17.4.9 int ts

Definition at line 62 of file main.cpp.

6.18 namesAccessor.cpp File Reference

```
#include <stdlib.h>
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include "globals.h"
```

Functions

- string getRandomName ()
- void loadNamesList (string fileName)
- void destroyNamesList ()

6.18.1 Function Documentation

6.18.1.1 void destroyNamesList ( )

Clears the names database from memory

Definition at line 35 of file namesAccessor.cpp.

6.18.1.2 string getRandomName ( )

Returns a random name from the names database

Definition at line 9 of file namesAccessor.cpp.

6.18.1.3 void loadNamesList (string *fileName*)

Reads the contents of the file into the program memory

Definition at line 17 of file namesAccessor.cpp.

6.19 namesAccessor.h File Reference

```
#include <stdlib.h>
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include "globals.h"
```

Functions

- string getRandomName ()
- void loadNamesList (string fileName)
- void destroyNamesList ()

6.19.1 Function Documentation

6.19.1.1 void destroyNamesList ( )

Clears the names database from memory

Definition at line 35 of file namesAccessor.cpp.

6.19.1.2 string getRandomName ( )

Returns a random name from the names database

Definition at line 9 of file namesAccessor.cpp.

6.19.1.3 void loadNamesList (string *fileName*)

Reads the contents of the file into the program memory

Definition at line 17 of file namesAccessor.cpp.

6.20 parser.cpp File Reference

```
#include <fstream>
#include <string>
#include <set>
#include <iostream>
#include <sstream>
#include <map>
#include <vector>
#include "containers.h"
#include "environment.h"
#include "parser.h"
```

Functions

- void readFile (string s, std::vector< Department * > &dep, std::vector< Course * > &course, std::vector< University * > &univ, int &facultyRandom, int &studentRandom, int &courseRandom, int &friendRandom)

6.20.1 Function Documentation

6.20.1.1 void readFile (string *s*, std::vector< **Department** * > & *dep*, std::vector< **Course** * > & *course*, std::vector< **University** * > & *univ*, int & *facultyRandom*, int & *studentRandom*, int & *courseRandom*, int & *friendRandom*)

Parses the contents of the provided SocialNetworkEnvironment file to set initialisation parameters for the simulation

Definition at line 14 of file parser.cpp.

6.21 parser.h File Reference

```
#include <fstream>
#include <string>
#include <set>
#include <iostream>
#include <sstream>
#include <map>
#include <vector>
#include "containers.h"
#include "globals.h"
```

Functions

- void readFile (string, std::vector< Department * > &, std::vector< Course * > &, std::vector< University * > &, int &, int &, int &, int &)

6.21.1 Function Documentation

6.21.1.1 void readFile (string , std::vector< **Department** * > & , std::vector< **Course** * > & , std::vector< **University** * > & , int & , int & , int & , int &)

Parses the contents of the provided SocialNetworkEnvironment file to set initialisation parameters for the simulation

Definition at line 14 of file parser.cpp.

6.22 readml.py File Reference

Namespaces

- namespace readml

Variables

- tuple readml.R = open('graph.graphml','r')
- tuple readml.f1 = open('adj.txt','w')
- tuple readml.f2 = open('vmap.txt','w')
- tuple readml.parts = line.split('"') f2.write(parts[1] + " " + parts[3] + " " + parts[5] + "\n") elif line[1:5] == 'edge': parts = line.split('"')

6.23 solve.cpp File Reference

```
#include <vector>
#include <queue>
#include <string>
#include <cstring>
#include <cstdio>
#include <cstdlib>
#include <fstream>
#include <iostream>
#include <map>
```

Macros

- #define mp make_pair
- #define pb push_back
- #define INF 1000000000

Typedefs

- typedef pair< int, int > PI
- typedef pair< PI, int > PPI
- typedef vector< int > VI
- typedef vector< PI > VP

Functions

- int getPerson (string s, int x)
- int dijkstra (int source, int destination)
- void shortestDistance (int a, int b)
- void shortestPath (int a, int b)
- void storeFW (int n)

  count grid for floyd warshall
- void floydWarshall (int n)
- float getImportance (int k, int n)
- void importance (int k, int n)
- void shortestDistance (int n)
- void moreImportant (int k, int n)
- void cliqueSize (int k, int n)
- void make_graph (int &n)
- int main (int argc, char *argv[])

Variables

- vector< VI > arr
- VI vis

  adjecency list
- VI par

  visited list
- vector< pair< string, int > > vmap

  parent list
- map< pair< string, int >, int > person
- vector< VI > dis
- vector< VI > cnt

  distance grid for floyd warshall
- int indices [20]

6.23.1 Macro Definition Documentation

6.23.1.1 #define INF 1000000000

Definition at line 18 of file solve.cpp.

6.23.1.2 #define mp make_pair

Definition at line 16 of file solve.cpp.

6.23.1.3 #define pb push_back

Definition at line 17 of file solve.cpp.

6.23.2 Typedef Documentation

6.23.2.1 typedef pair<int,int> PI

Definition at line 12 of file solve.cpp.

6.23.2.2 typedef pair<PI,int> PPI

Definition at line 13 of file solve.cpp.

6.23.2.3 typedef vector<int> VI

Definition at line 14 of file solve.cpp.

6.23.2.4 typedef vector<PI> VP

Definition at line 15 of file solve.cpp.

6.23.3 Function Documentation

6.23.3.1 void cliqueSize (int *k*, int *n*)

Calculates the largest completely connected subgraph that contains the given person

Definition at line 178 of file solve.cpp.

6.23.3.2 int dijkstra (int *source*, int *destination*)

calculates the shortest path between the source and destination, returns -1 when not connected.

Definition at line 32 of file solve.cpp.

6.23.3.3 void floydWarshall (int *n*)

Reads the output of the Floyd Warshall computation from the file and stores the information of shortest paths in program memory.

Definition at line 113 of file solve.cpp.

6.23.3.4 float getImportance (int *k*, int *n*)

Determines the number of shortest paths passing through the given person.

Definition at line 130 of file solve.cpp.

6.23.3.5 int getPerson (string *s*, int *x*)

Definition at line 25 of file solve.cpp.

6.23.3.6 void importance (int *k*, int *n*)

Interfaces the getImportance function

Definition at line 142 of file solve.cpp.

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

This function interfaces with the query engine in perl and relays answers to queries.

Definition at line 234 of file solve.cpp.

6.23.3.8 void make_graph (int & *n*)

Makes an adjacency list out of the graph file

Definition at line 207 of file solve.cpp.

6.23.3.9 void moreImportant (int *k*, int *n*)

Evaluates if any friend of given person is more important that him.

Definition at line 161 of file solve.cpp.

6.23.3.10 void shortestDistance (int *a,* int *b*)

calls the Dijkstra algorithm and returns the shortest distance between two people

Definition at line 51 of file solve.cpp.

6.23.3.11 void shortestDistance (int *n*)

Returns the diameter of graph

Definition at line 148 of file solve.cpp.

6.23.3.12 void shortestPath (int *a,* int *b*)

calls the Dijkstra algorithm and returns the shortest path between two people

Definition at line 59 of file solve.cpp.

6.23.3.13 void storeFW (int *n*)

count grid for floyd warshall

Initialises the grid and implements the Floyd-Warshall all-pair shortest path algorithm

Definition at line 78 of file solve.cpp.

6.23.4 Variable Documentation

6.23.4.1 vector<VI> arr

Definition at line 20 of file solve.cpp.

6.23.4.2 vector<VI> cnt

distance grid for floyd warshall

Definition at line 77 of file solve.cpp.

6.23.4.3 vector<VI> dis

Definition at line 76 of file solve.cpp.

6.23.4.4 int indices[20]

Definition at line 177 of file solve.cpp.

6.23.4.5 VI par

visited list

Definition at line 22 of file solve.cpp.

6.23.4.6 map<pair<string,int>,int> person

Definition at line 24 of file solve.cpp.

6.23.4.7 VI vis

adjecency list

Definition at line 21 of file solve.cpp.

6.23.4.8 vector<pair<string,int> > vmap

parent list

Definition at line 23 of file solve.cpp.

6.24 student.cpp File Reference

```
#include <string>
#include <map>
#include <vector>
#include <cstdlib>
#include <stdio.h>
#include <iostream>
#include <unistd.h>
#include "student.h"
#include "globals.h"
#include "namesAccessor.h"
```

Macros

- #define PERSON pair<string,int>
- #define DEPARTMENT pair<string,string>
- #define INTEREST string
- #define COURSE pair<string,string>
- #define HOME pair<string,string>
- #define UNIV string
- #define VP vector<PERSON>
- #define SP set<PERSON >
- #define ADM map<PERSON,SP >

Functions

- void updateYear ()
- void generateStudents (std::vector< University * > _universities, int _studentSeed)
- void generateStudent (University *_university, Department *_department, int _studentId)
- void printStudents ()

6.24.1 Macro Definition Documentation

6.24.1.1 #define ADM map<PERSON,SP >

6.24.1.2 #define COURSE pair<string,string>

6.24.1.3 #define DEPARTMENT pair<string,string>

6.24.1.4 #define HOME pair<string,string>

6.24.1.5 #define INTEREST string

6.24.1.6 #define PERSON pair<string,int>

6.24.1.7 #define SP set<PERSON >

6.24.1.8 #define UNIV string

6.24.1.9 #define VP vector<PERSON>

6.24.2 Function Documentation

6.24.2.1 void generateStudent (University * *_university,* Department * *_department,* int *_studentId*)

Generates a new student in the given university and department, and assigns interests and hostel randomly based on the seed StudentRandom

Definition at line 65 of file student.cpp.

6.24.2.2 void generateStudents (std::vector< University * > *_universities,* int *_studentSeed*)

Generates a new batch of students every year, and assigns them to universities and departments

Definition at line 45 of file student.cpp.

6.24.2.3 void printStudents ( )

Definition at line 203 of file student.cpp.

6.24.2.4 void updateYear ( )

Updates interests and courses information every year and stores the updated information into compiled lists of students sharing the common interest/course.

Definition at line 97 of file student.cpp.

6.25 student.h File Reference

```
#include <string>
#include <map>
#include <vector>
#include "containers.h"
```

Classes

- class Student

Functions

- void generateStudents (std::vector< University * >, int)

- void generateStudent (University *, Department *, int)
- void printStudents ()

6.25.1 Function Documentation

6.25.1.1 void generateStudent (University * , Department * , int)

Generates a new student in the given university and department, and assigns interests and hostel randomly based on the seed StudentRandom

Definition at line 65 of file student.cpp.

6.25.1.2 void generateStudents (std::vector< University * > , int)

Generates a new batch of students every year, and assigns them to universities and departments

Definition at line 45 of file student.cpp.

6.25.1.3 void printStudents ( )

Definition at line 203 of file student.cpp.

Index