STudent_v.0.1

Generated by Doxygen 1.9.1

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

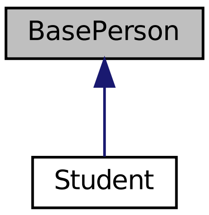
Class Documentation

4.1 BasePerson Class Reference

A base class representing a person with basic attributes like id, name, age, and phone number.

`#include <BasePerson.h>`

Inheritance diagram for BasePerson:



Public Member Functions

- BasePerson ()

  Default constructor that initializes the member variables to default values.
- BasePerson (int idParam, string nameParam, int ageParam, string phoneNumberParam)

  Parameterized constructor that initializes the member variables with provided values.
- void setId (int idParam)

  Sets the ID of the person.
- void setName (string nameParam)

  Sets the name of the person.
- void setAge (int ageParam)

  Sets the age of the person.
- void setPhoneNumber (string phoneNumberParam)

Sets the phone number of the person.
- int getId ()

  Gets the ID of the person.
- string getName ()

  Gets the name of the person.
- int getAge ()

  Gets the age of the person.
- string GetPhoneNumber ()

  Gets the phone number of the person with the country code +20 prefixed.
- virtual void print ()

  Prints the details of the person.

Protected Attributes

- int id

  ID of the person.
- string name

  Name of the person.
- int age

  Age of the person.
- string phoneNumber

  Phone number of the person.

4.1.1 Detailed Description

A base class representing a person with basic attributes like id, name, age, and phone number.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 BasePerson()

```
BasePerson::BasePerson (
                int idParam,
                string nameParam,
                int ageParam,
                string phoneNumberParam )
```

Parameterized constructor that initializes the member variables with provided values.

Parameters

idParam	The ID of the person.
nameParam	The name of the person.
ageParam	The age of the person.
phoneNumberParam	The phone number of the person.

4.1.3 Member Function Documentation

4.1.3.1 getAge()

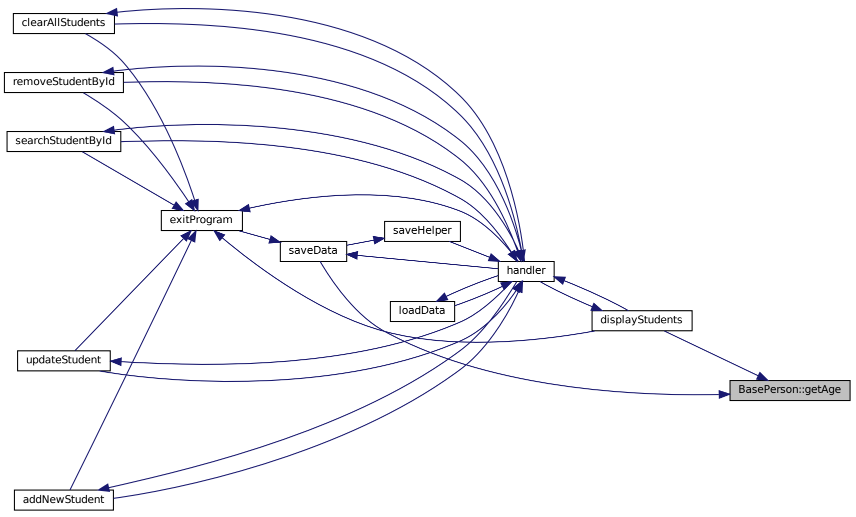
```
int BasePerson::getAge ( )
```

Gets the age of the person.

Returns

The age of the person.

Here is the caller graph for this function:



4.1.3.2 getId()

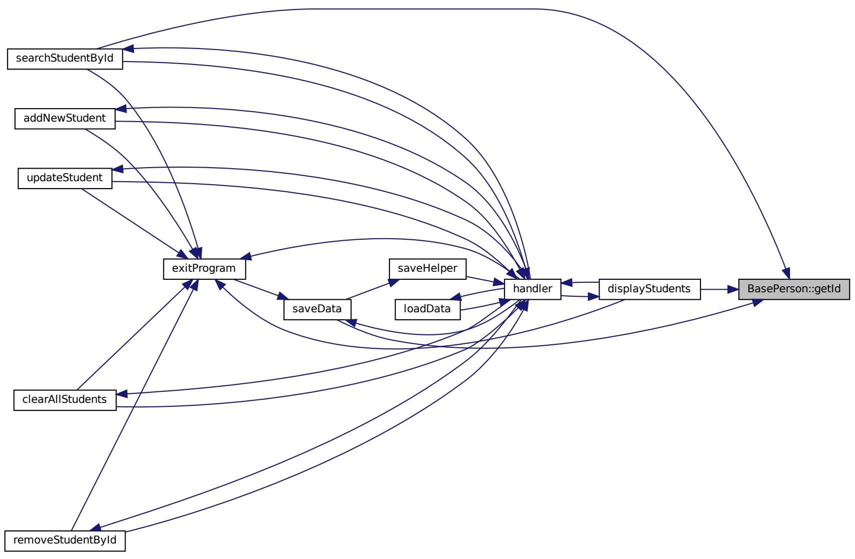
```
int BasePerson::getId ( )
```

Gets the ID of the person.

Returns

The ID of the person.

Here is the caller graph for this function:



4.1.3.3 getName()

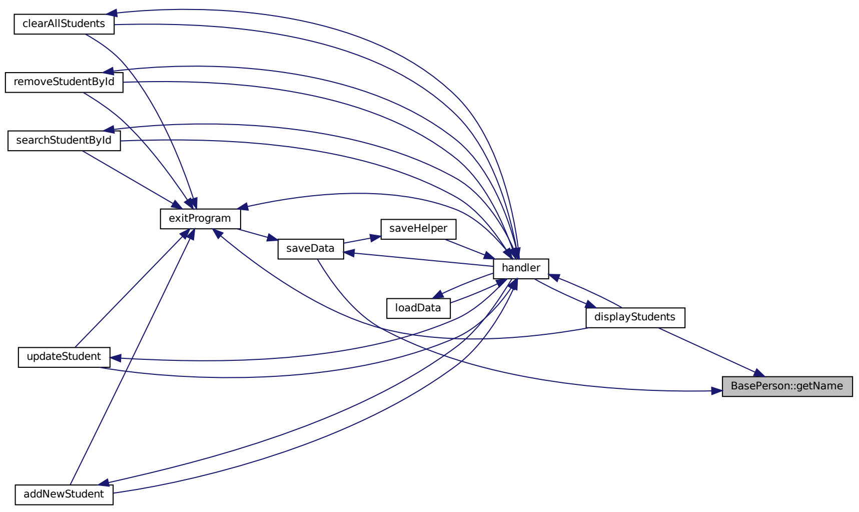
```
string BasePerson::getName ( )
```

Gets the name of the person.

Returns

The name of the person.

Here is the caller graph for this function:



4.1.3.4 GetPhoneNumber()

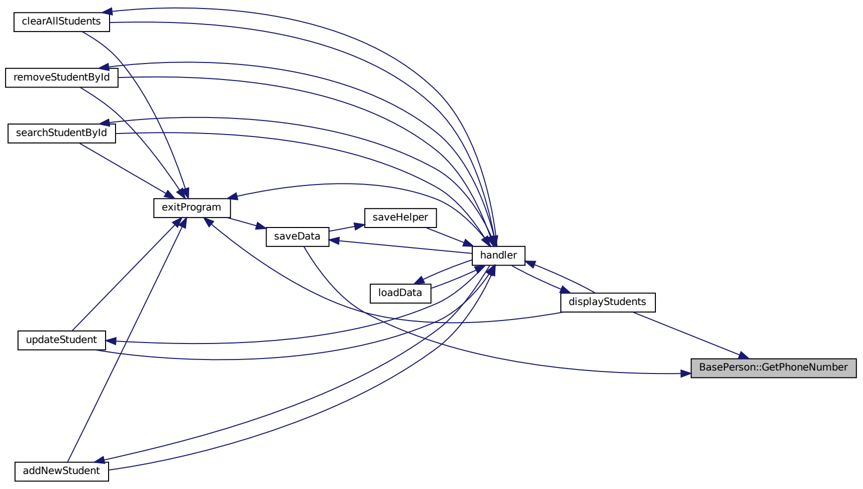
```
string BasePerson::GetPhoneNumber ( )
```

Gets the phone number of the person with the country code +20 prefixed.

Returns

The phone number of the person.

Here is the caller graph for this function:



4.1.3.5 setAge()

```
void BasePerson::setAge (
             int ageParam )
```

Sets the age of the person.

Parameters

ageParam	The age to set.

4.1.3.6 setId()

```
void BasePerson::setId (
            int idParam )
```

Sets the ID of the person.

Parameters

idParam	The ID to set.

4.1.3.7 setName()

```
void BasePerson::setName (
            string nameParam )
```

Sets the name of the person.

Parameters

nameParam	The name to set.

4.1.3.8 setPhoneNumber()

```
void BasePerson::setPhoneNumber (
            string phoneNumberParam )
```

Sets the phone number of the person.

Parameters

phoneNumberParam	The phone number to set.

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

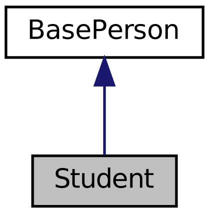
- BasePerson.h
- BasePerson.cpp

4.2 Student Class Reference

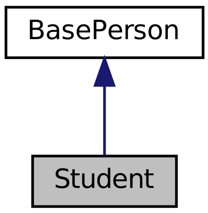
Represents a Student, derived from BasePerson.

```
#include <Student.h>
```

Inheritance diagram for Student:



Collaboration diagram for Student:



Public Member Functions

- Student ()

  Default constructor for the Student class.
- Student (int idParam, string nameParam, int ageParam, string departmentParam, string StudentYearParam, string phoneNumberParam, double gpaParam)

  Parameterized constructor for the Student class.
- void setGpa (double gpaParam)

  Sets the GPA for the Student.
- void setDeparment (string departmentParam)

  Sets the department for the Student.
- void setStudentYear (string StudentYearParam)

  Sets the year of study for the Student.
- double getGpa ()

  Gets the GPA of the Student.
- string getDepartment ()

  Gets the department of the Student.
- string getStudentYear ()

  Gets the year of study for the Student.
- void print ()

  Prints the details of the Student.

Additional Inherited Members

4.2.1 Detailed Description

Represents a Student, derived from BasePerson.

This class contains information about a Student, including their department, year of study, and GPA.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Student() [1/2]

```
Student::Student ( )
```

Default constructor for the Student class.

Initializes a Student object with default values.

This constructor initializes a Student object with default values.

4.2.2.2 Student() [2/2]

```
Student::Student (
              int idParam,
              string nameParam,
              int ageParam,
              string departmentParam,
              string StudentYearParam,
              string phoneNumberParam,
              double gpaParam )
```

Parameterized constructor for the Student class.

Initializes a Student object with specified values.

Parameters

idParam	The Student's ID.
nameParam	The Student's name.
ageParam	The Student's age.
departmentParam	The Student's department.
StudentYearParam	The year of study for the Student.
phoneNumberParam	The Student's phone number.
gpaParam	The Student's GPA.

This constructor initializes a Student object with specified values.

Parameters

idParam	The Student's ID.
nameParam	The Student's name.
ageParam	The Student's age.
departmentParam	The Student's department.
StudentYearParam	The year of study for the Student.
phoneNumberParam	The Student's phone number.
gpaParam	The Student's GPA.

4.2.3 Member Function Documentation

4.2.3.1 getDepartment()

```
string Student::getDepartment ( )
```

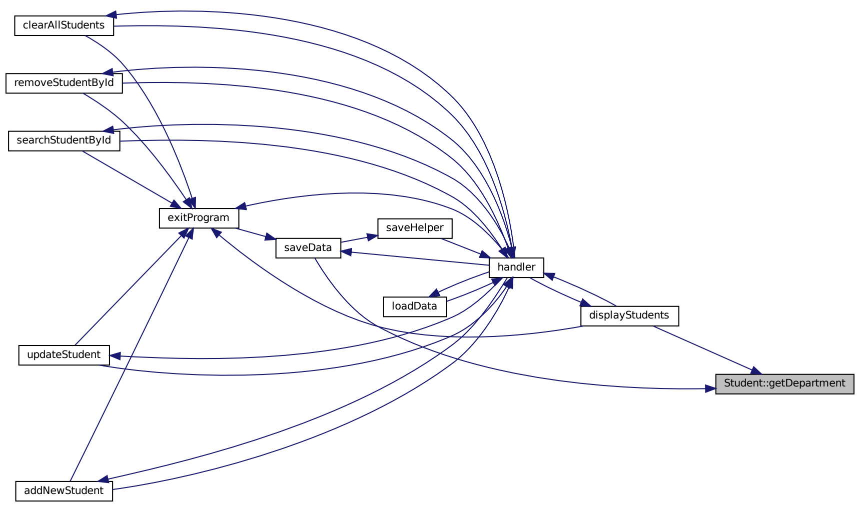
Gets the department of the Student.

Returns

The department of the Student.

The Student's department.

Here is the caller graph for this function:



4.2.3.2 getGpa()

```
double Student::getGpa ( )
```

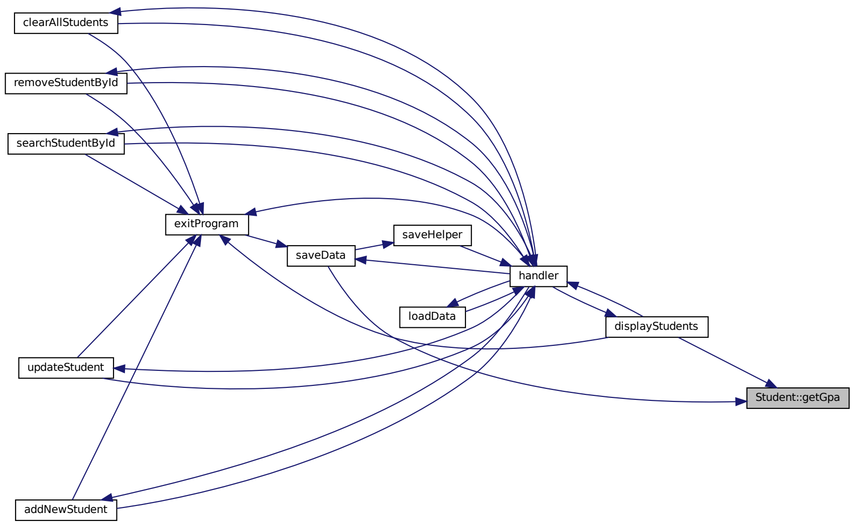
Gets the GPA of the Student.

Returns

The GPA of the Student.

The Student's GPA.

Here is the caller graph for this function:



4.2.3.3 getStudentYear()

```
string Student::getStudentYear ( )
```

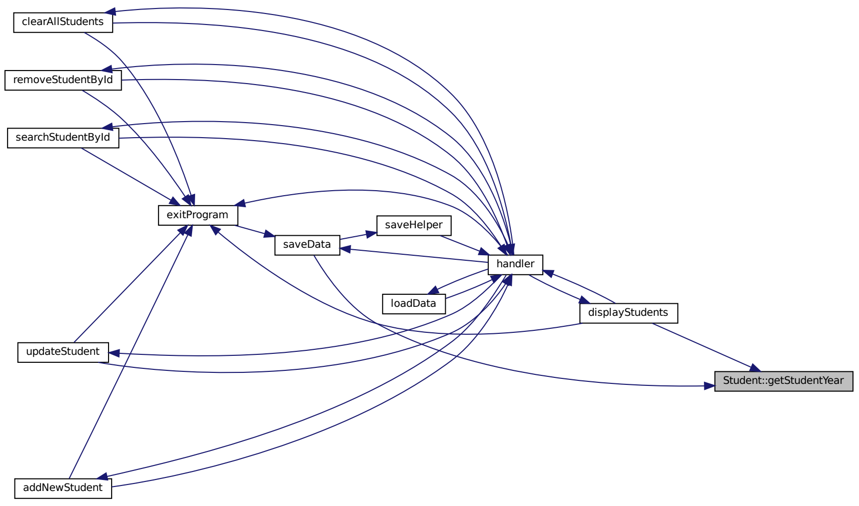
Gets the year of study for the Student.

Returns

The year of study of the Student.

The year of study.

Here is the caller graph for this function:



4.2.3.4 print()

```
void Student::print ( ) [virtual]
```

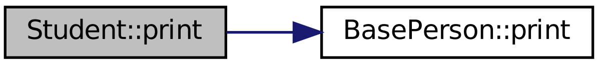
Prints the details of the Student.

This function prints the Student's GPA, department, and year of study, along with the base person details.

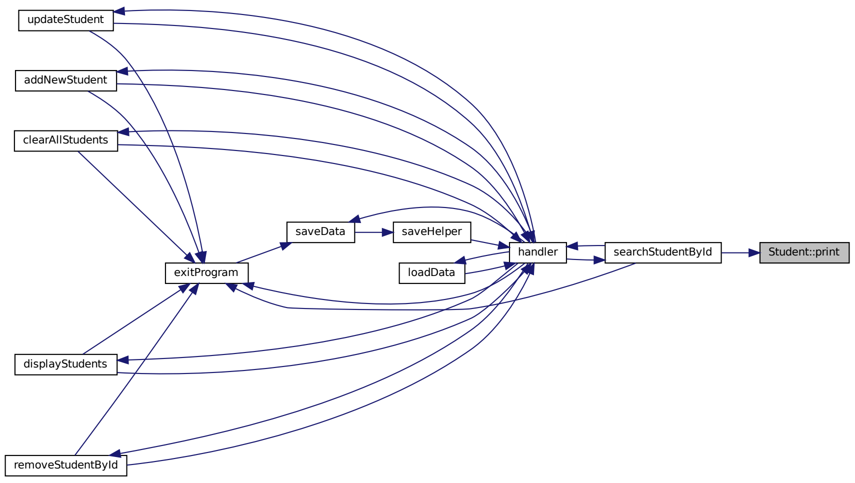
This function prints the Student's personal details along with GPA, department, and year of study.

Reimplemented from BasePerson.

Here is the call graph for this function:



Here is the caller graph for this function:



4.2.3.5 setDeparment()

```
void Student::setDeparment (
             string departmentParam )
```

Sets the department for the Student.

Sets the department of the Student.

Parameters

departmentParam	The department to set for the Student.
departmentParam	The department to set.

4.2.3.6 setGpa()

```
void Student::setGpa (
             double gpaParam )
```

Sets the GPA for the Student.

Sets the GPA of the Student.

Parameters

gpaParam	The GPA to set for the Student.
gpaParam	The GPA to set.

4.2.3.7 setStudentYear()

```
void Student::setStudentYear (
             string StudentYearParam )
```

Sets the year of study for the Student.

Parameters

StudentYearParam	The year of study to set for the Student.
StudentYearParam	The year of study to set.

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

- Student.h
- Student.cpp

Chapter 5

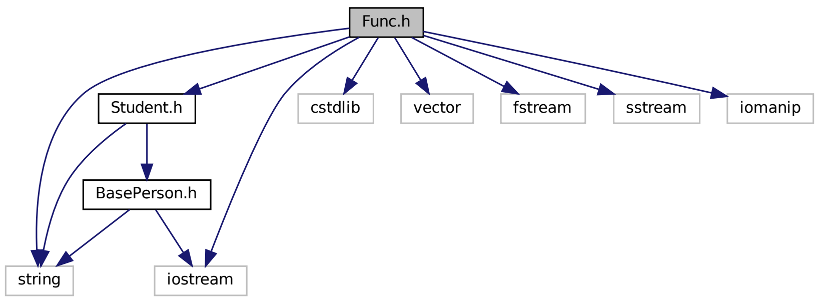
File Documentation

5.1 Func.h File Reference

Provides declarations for functions related to Student management.

```
#include <string>
#include <cstdlib>
#include <iostream>
#include <vector>
#include <fstream>
#include <sstream>
#include <iomanip>
#include "Student.h"
```

Include dependency graph for Func.h:



Functions

- void show ()

  Displays the main menu options for the Student management system.
- void handler ()

  Handles the user's input and processes the respective functions.
- void addNewStudent ()

  Adds a new Student to the system.

- void updateStudent ()

    Updates an existing Student's information.
- void searchStudentById ()

    Searches for a Student using their ID.
- void removeStudentById ()

    Removes a Student from the system using their ID.
- void displayStudents ()

    Displays all Students currently in the system.
- void clearAllStudents ()

    Clears all Student records from the system.
- void saveData ()

    Saves the current Student data to a file.
- void saveHelper ()

    A helper function used during the data saving process.
- void loadData ()

    Loads Student data from a file.
- void exitProgram ()

    Handles the process of exiting the program.

Variables

- vector< Student > Students

    A global vector to store all Student objects.
- int YesNo

    A global variable to capture Yes/No responses.
- bool exitStatus

    A global status variable to manage program exit status.

5.1.1 Detailed Description

Provides declarations for functions related to Student management.

5.1.2 Function Documentation

5.1.2.1 addNewStudent()

```
void addNewStudent ( )
```

Adds a new Student to the system.

Adds a new Student to the system.

This function prompts the user to enter details for a new student, checks if the entered ID is already taken, and then adds the new student to the list of students.

If the ID is already taken, the user is prompted to enter another ID, with a maximum of 3 attempts before the program exits. The ID of the student.

The name of the student.

The age of the student.

The department of the student.

The academic year of the student.

The phone number of the student.

The GPA of the student.

Static variable to track the number of attempts.

Check if the ID is already taken by any existing student.

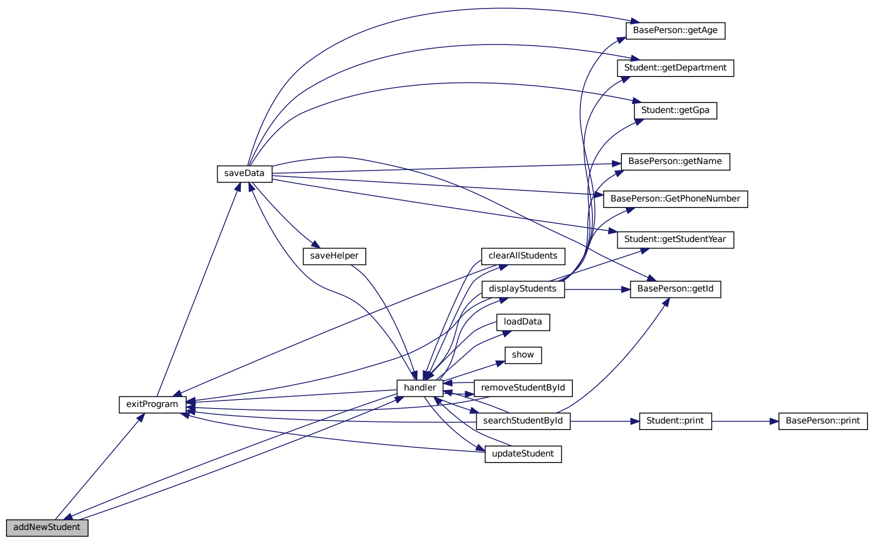
Loops through the list of students and compares their IDs to the newly entered one.

Recursive call if the ID is already taken, allowing up to 3 attempts.

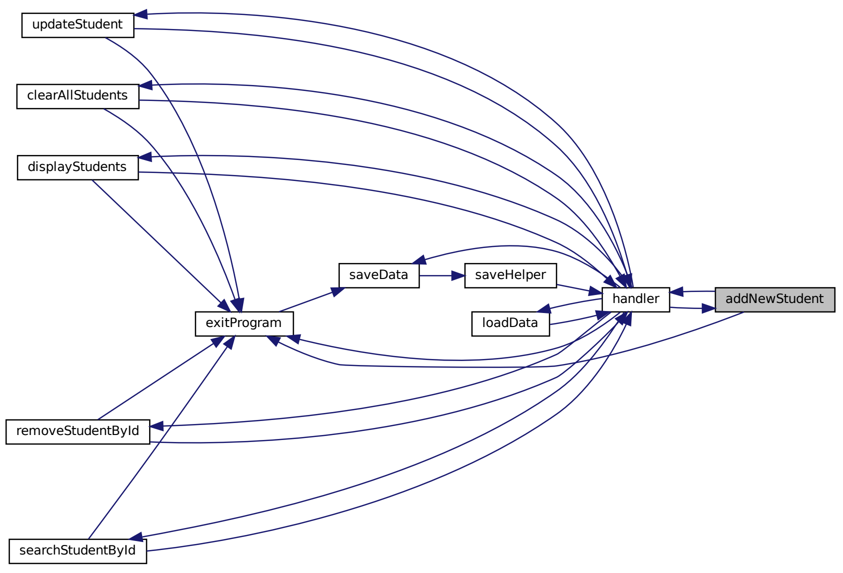
Create a new student object and add it to the list.

Prompt the user to continue or exit the program.

If the user selects "Yes", the handler function is called. Otherwise, the program exits.Here is the call graph for this function:



Here is the caller graph for this function:



5.1.2.2 clearAllStudents()

```
void clearAllStudents ( )
```

Clears all Student records from the system.

Clears all Student records from the system.

Provides three options for clearing students:

1. Clear from memory only.
2. Clear from the database only.
3. Clear from both memory and the database.

Parameters

choice	Variable to store the user's choice of action.
YesNo	Variable to store user input for processing another operation.

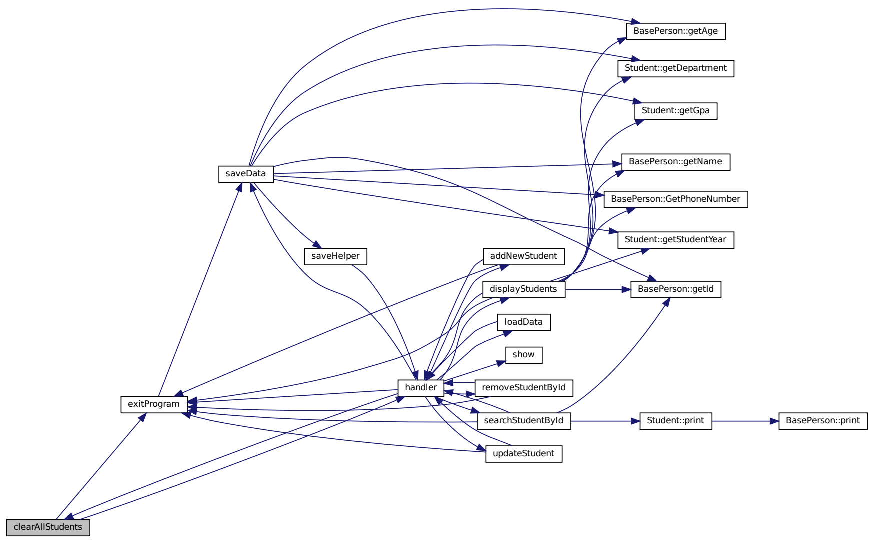
This function handles user input and processes clearing students from memory and/or database. < Variable to store the user's choice.

< Asks if the user wants to perform another operation.

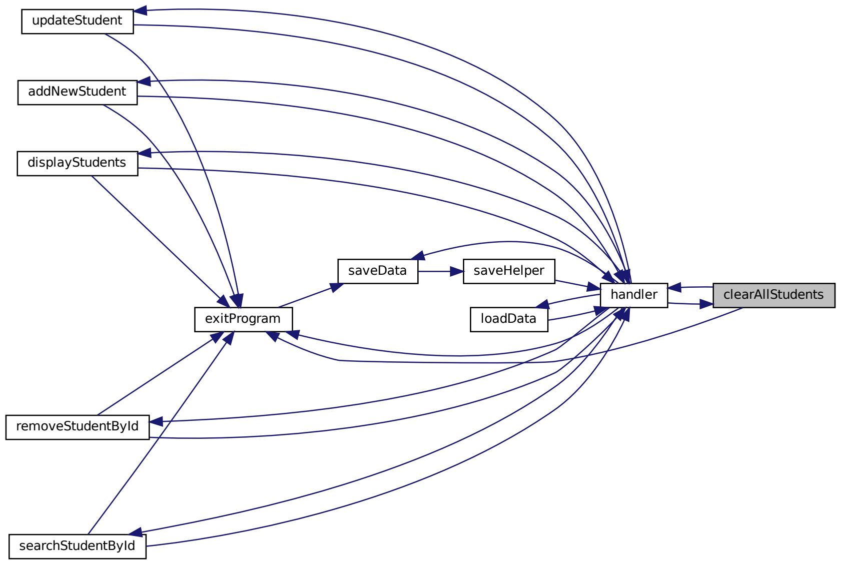
< Resets the YesNo variable for future use.

< Calls the handler function to process another operation.

< Exits the program if the user selects "No".Here is the call graph for this function:



Here is the caller graph for this function:



5.1.2.3 displayStudents()

```
void displayStudents ( )
```

Displays all Students currently in the system.

Displays all Students currently in the system.

If there are no students in the list (Students), it informs the user that no students are available to display. If there are students, it prints a table showing the ID, Name, Age, Department, StudentYear, PhoneNumber, and GPA of each student. Finally, it asks the user whether they want to perform another operation or exit.

Parameters

None	

Returns

void

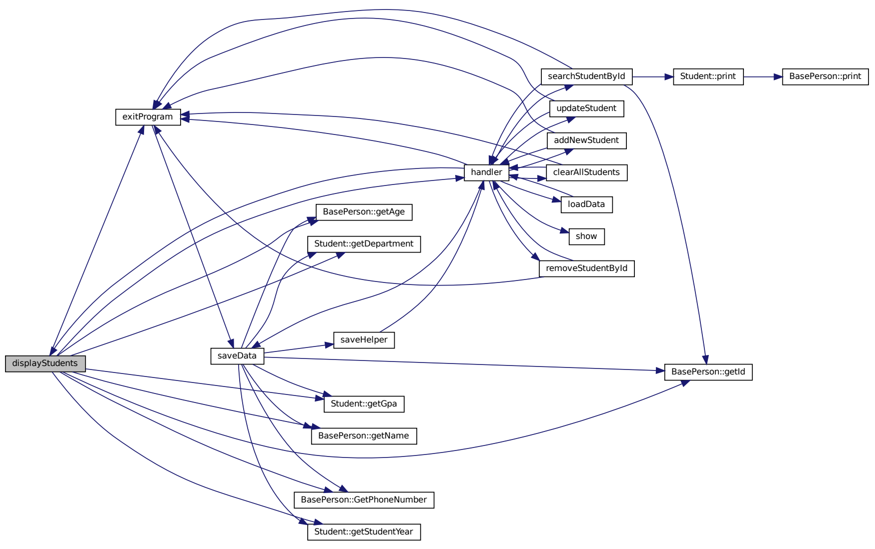
< Call the handler function if user wants to continue.

< Exit the program if user chooses to stop.

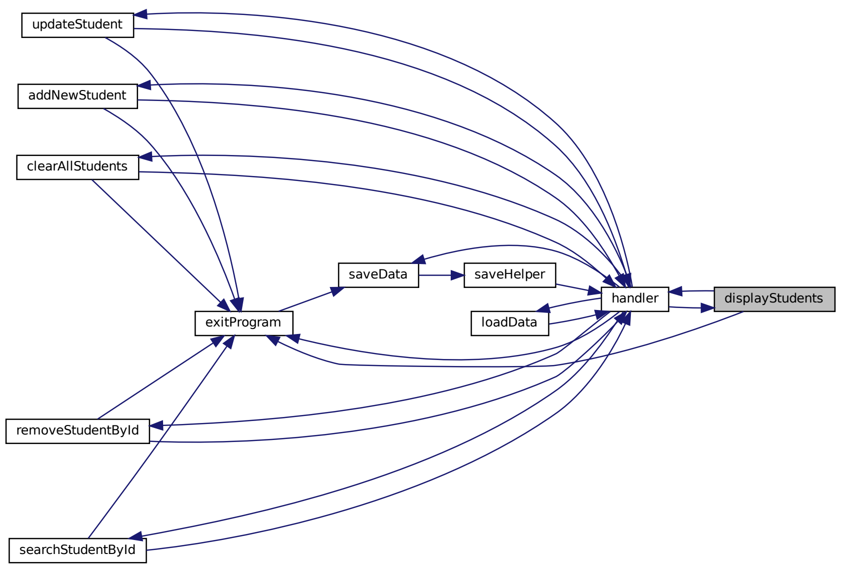
< Print separator line.

< Call the handler function if user wants to continue.

< Exit the program if user chooses to stop.Here is the call graph for this function:



Here is the caller graph for this function:



5.1.2.4 exitProgram()

```
void exitProgram ( )
```

Handles the process of exiting the program.

Handles the process of exiting the program.

This function prompts the user to save any changes before exiting. If the user chooses to save, the saveData function is called.

Parameters

YesNo	A variable to store the user's choice to save or not.
exitStatus	A flag to indicate the status before exiting.

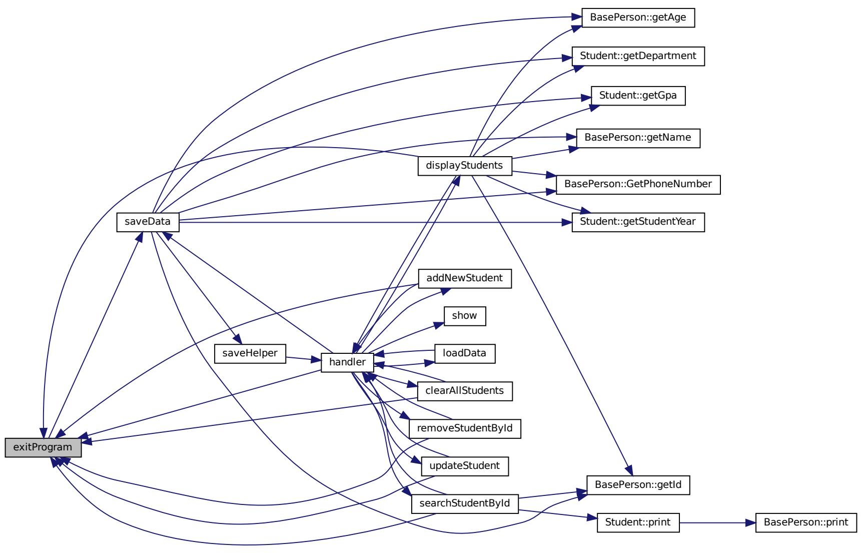
< Prompt the user to save changes before exiting.

< Set exit status to indicate that saveData should be called.

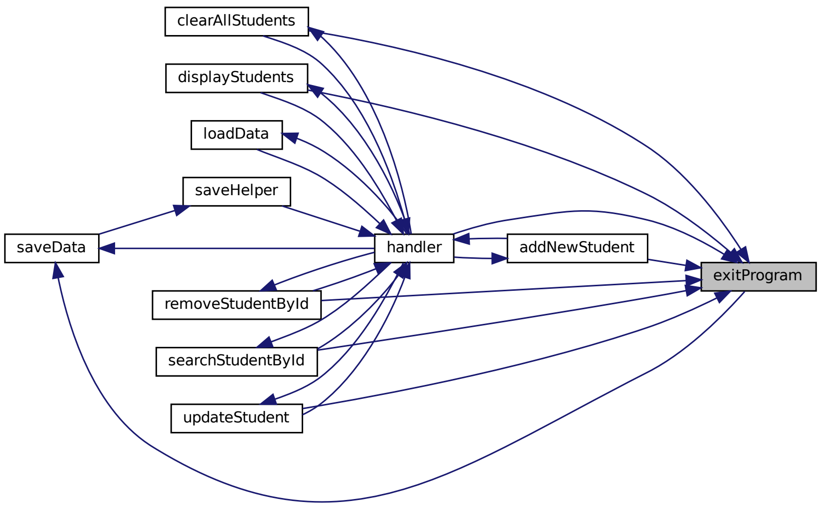
< Reset YesNo after user input.

< Call saveData to save changes.

< Terminate the program.Here is the call graph for this function:



Here is the caller graph for this function:



5.1.2.5 handler()

```
void handler ( )
```

Handles the user's input and processes the respective functions.

Handles the user's input and processes the respective functions.

This function takes the user's input and calls the corresponding function based on the choice. It uses a switch-case structure to determine the action to be taken. Static variable to store the user's choice.

Case 1: Display all students.

Case 2: Clear all student records.

Case 3: Add a new student.

Case 4: Remove a student by their ID.

Case 5: Update a student's information.

Case 6: Search for a student by their ID.

Case 7: Save all student data to a file.

Case 8: Load student data from a file.

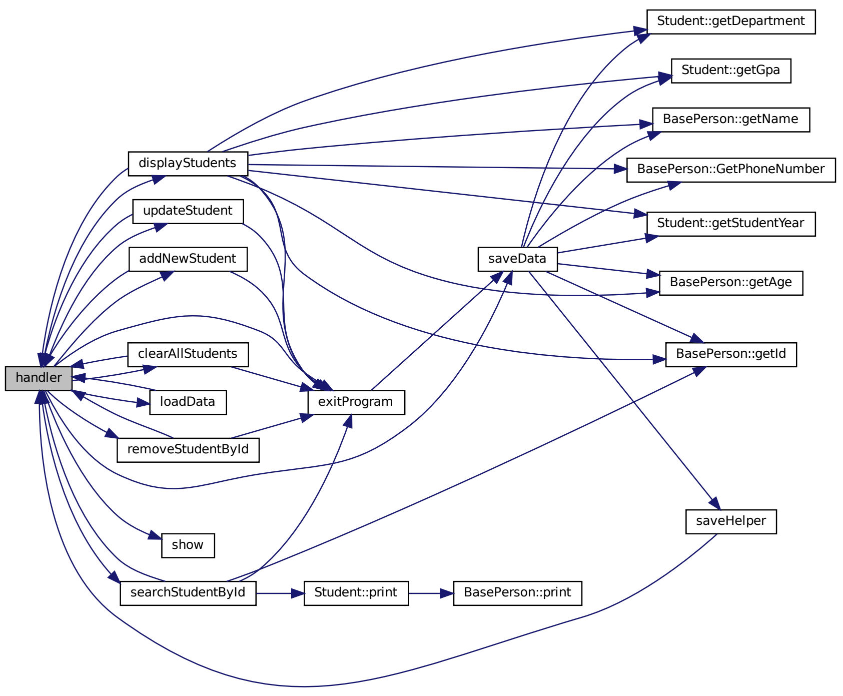
Case 9: Exit the program.

Default case: Handle invalid input from the user.

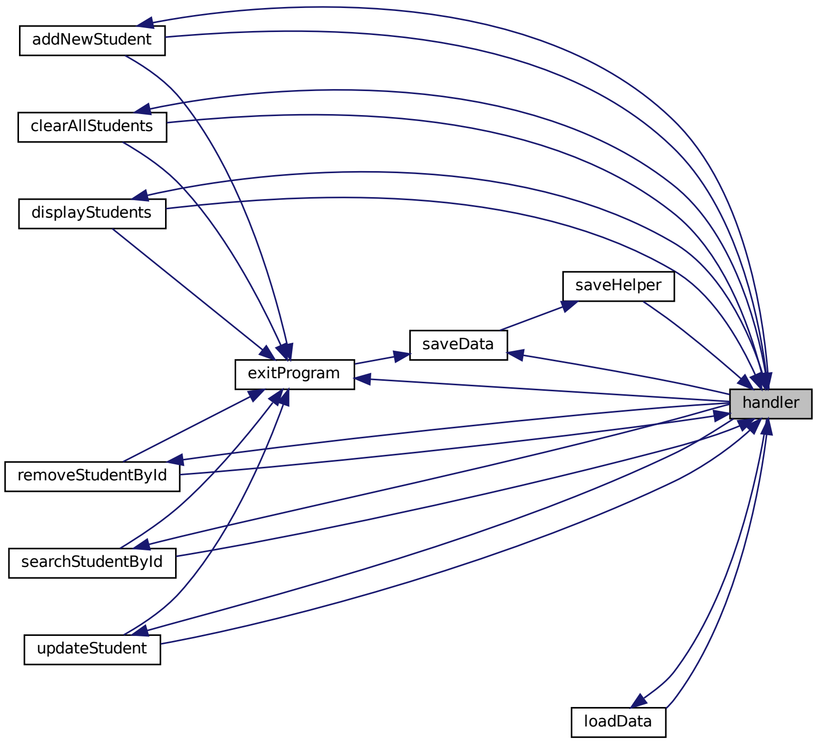
Static variable to track the number of incorrect attempts.

Check if the new input is valid and re-call handler.

Exits the program after three invalid attempts.Here is the call graph for this function:



Here is the caller graph for this function:



5.1.2.6 loadData()

```
void loadData ( )
```

Loads Student data from a file.

Loads Student data from a file.

This function reads student data from "data//Students.txt", parses the content, and adds each student to the Students vector. The first line (header) is skipped.

Parameters

loadNum	A static counter to track if this function has been called before.
YesNo	A variable to store the user's response for performing another operation.

If the file cannot be opened, an error message is displayed. After loading the data, the user is asked if they want to perform another operation if the function has been called more than once. < Static counter to track if loadData has been called before.

< Input file stream to read the data file.

< Temporary string to store each line of the file.

< Skip the header line.

< String stream to parse each line.

< Convert id string to integer.

< Convert age string to integer.

< Convert GPA string to double.

< Close the file after reading all data.

< Error message if file can't be opened.

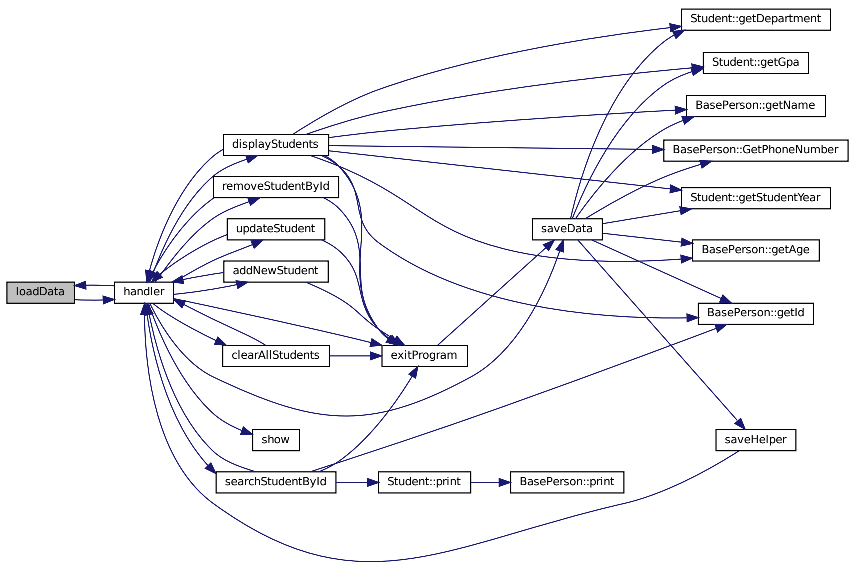
< Ask if the user wants to perform another operation.

< Reset YesNo to 0 for future use.

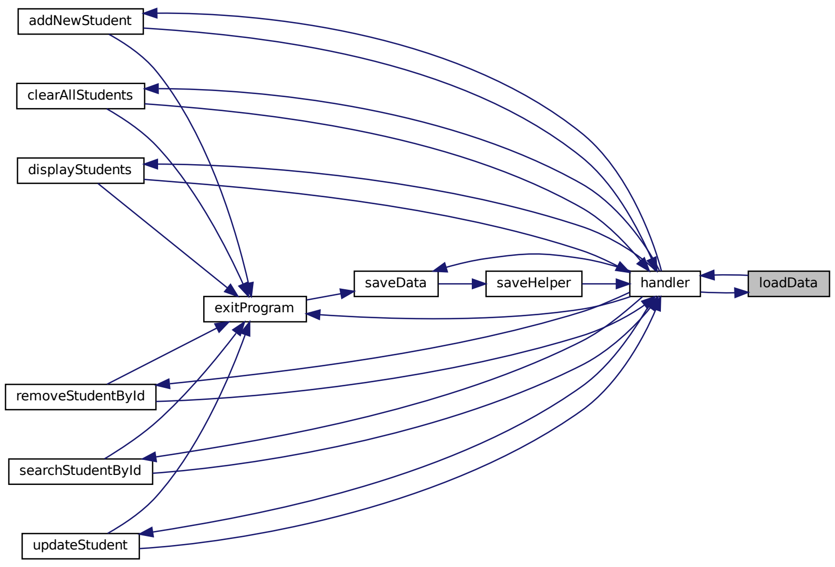
< Call handler to continue the program.

< Exit the program.

< Increment loadNum to indicate that loadData has been called.Here is the call graph for this function:



Here is the caller graph for this function:



5.1.2.7 removeStudentById()

```
void removeStudentById ( )
```

Removes a Student from the system using their ID.

Removes a Student from the system using their ID.

Prompts the user to enter a student ID, then searches the list of students (Students) and removes the student if found. If the student is removed, it confirms the removal. If not found, it informs the user. Finally, it asks the user whether they want to perform another operation or exit.

Parameters

None	

Returns

void

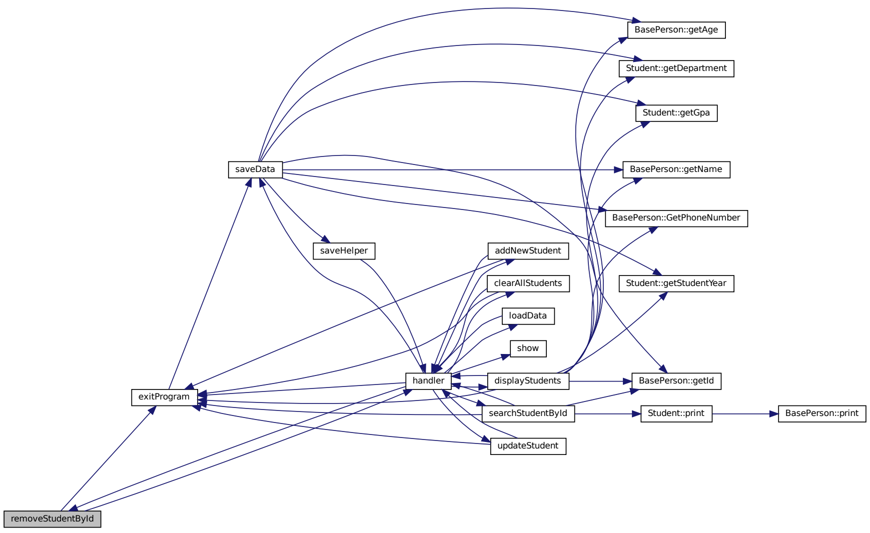
< Variable to store the student ID entered by the user.

< Flag to track if the student is found.

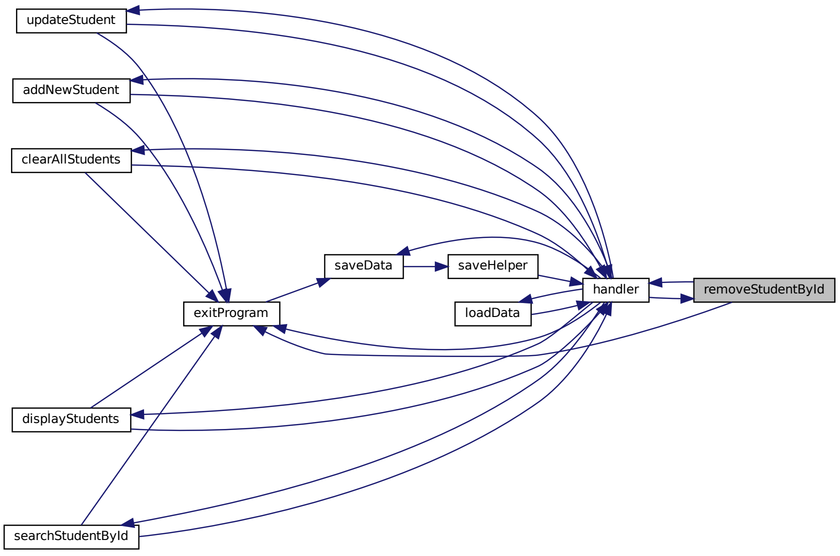
< Remove student from the list.

< Call the handler function if user wants to continue.

< Exit the program if user chooses to stop.Here is the call graph for this function:



Here is the caller graph for this function:



5.1.2.8 saveData()

```
void saveData ( )
```

Saves the current Student data to a file.

Saves the current Student data to a file.

This function writes all student information to a file located at "data//Students.txt". If the file cannot be opened, it will display an error message.

Parameters

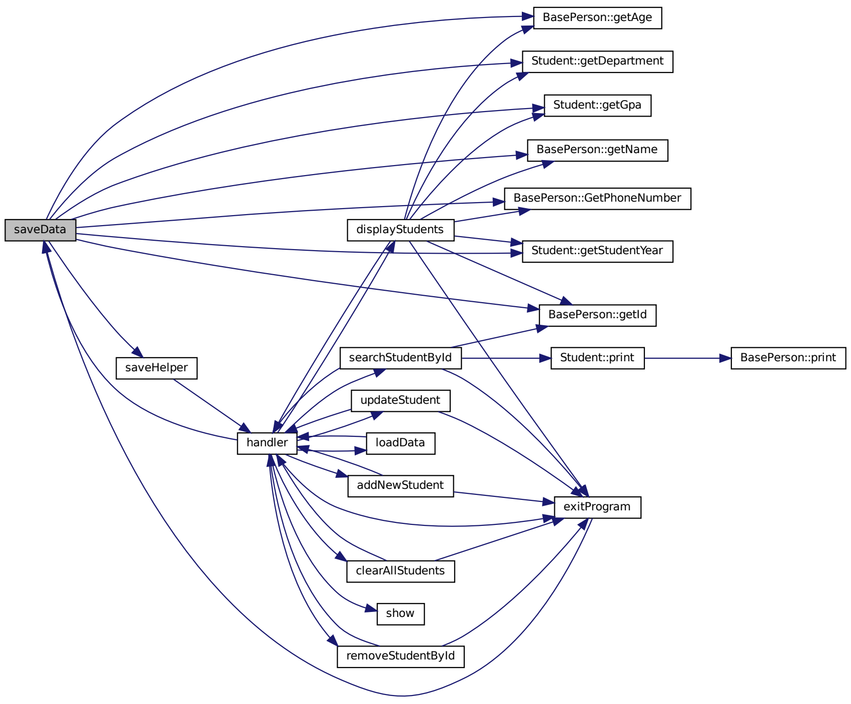
exitStatus	A status flag to track if the function should call saveHelper.

< Opens file to write student data.

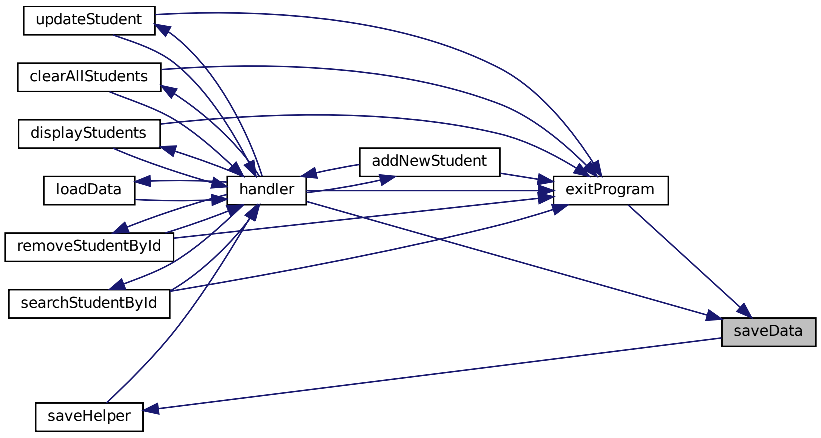
< Closes the file after writing.

< Calls helper function to ask if the user wants to perform another operation.

< Updates the exit status flag.Here is the call graph for this function:



Here is the caller graph for this function:



5.1.2.9 saveHelper()

```
void saveHelper ( )
```

A helper function used during the data saving process.

A helper function used during the data saving process.

Asks the user if they would like to perform another operation after saving data.

Parameters

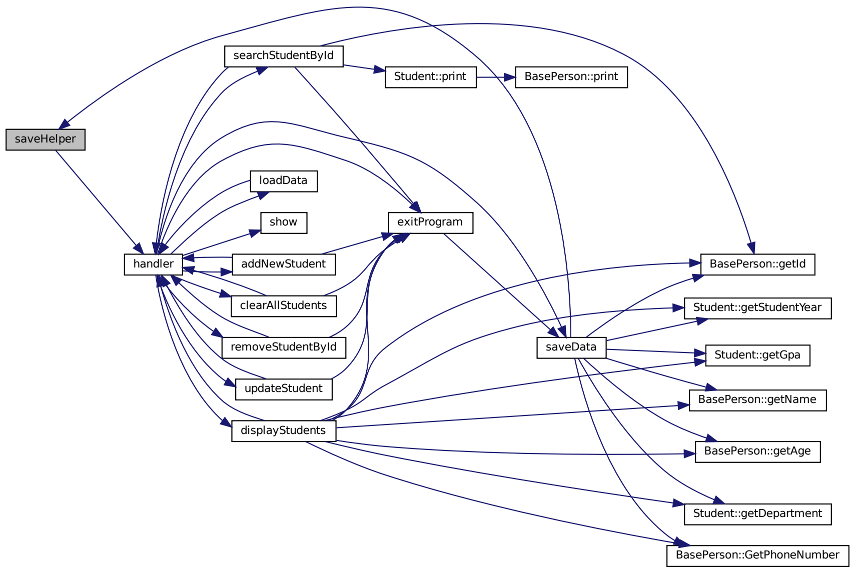
YesNo	Variable to store the user's response.

< Takes input from the user.

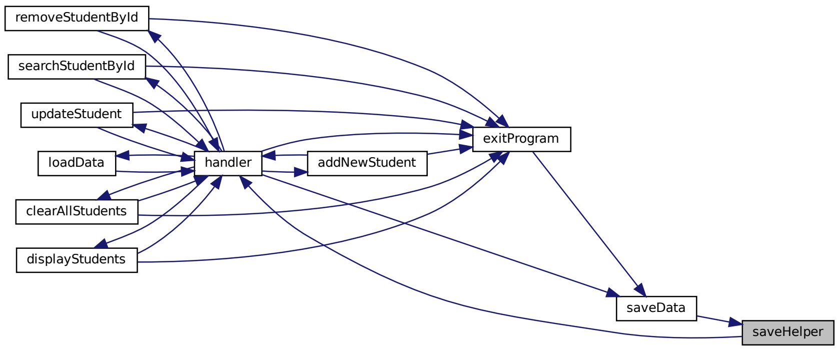
< Resets the YesNo variable.

< Calls the handler function for the next operation.

< Terminates the program.Here is the call graph for this function:



Here is the caller graph for this function:



5.1.2.10 searchStudentById()

```
void searchStudentById ( )
```

Searches for a Student using their ID.

Searches for a Student using their ID.

Prompts the user to enter a student ID, and then searches the list of students (Students) to find a matching student. If found, it displays the student's details. If not found, it informs the user. After that, the user is asked whether they want to perform another operation or exit the program.

Parameters

None	

Returns

void

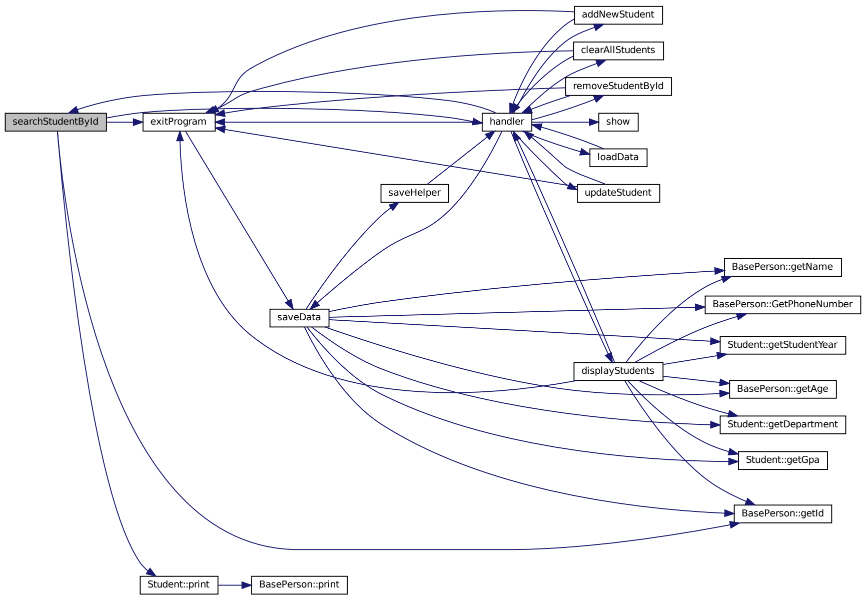
< Variable to store the student ID entered by the user.

< Flag to track if the student is found.

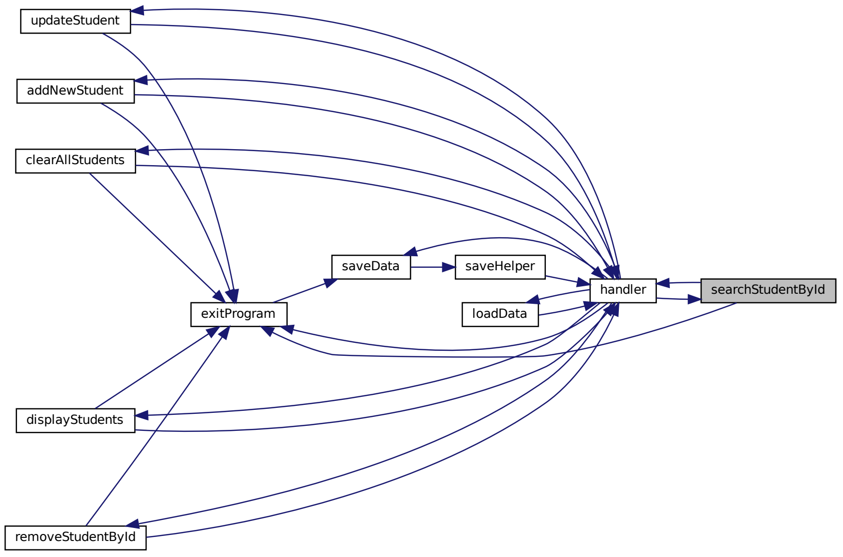
< Print student details if found.

< Call the handler function if user wants to continue.

< Exit the program if user chooses to stop.Here is the call graph for this function:



Here is the caller graph for this function:



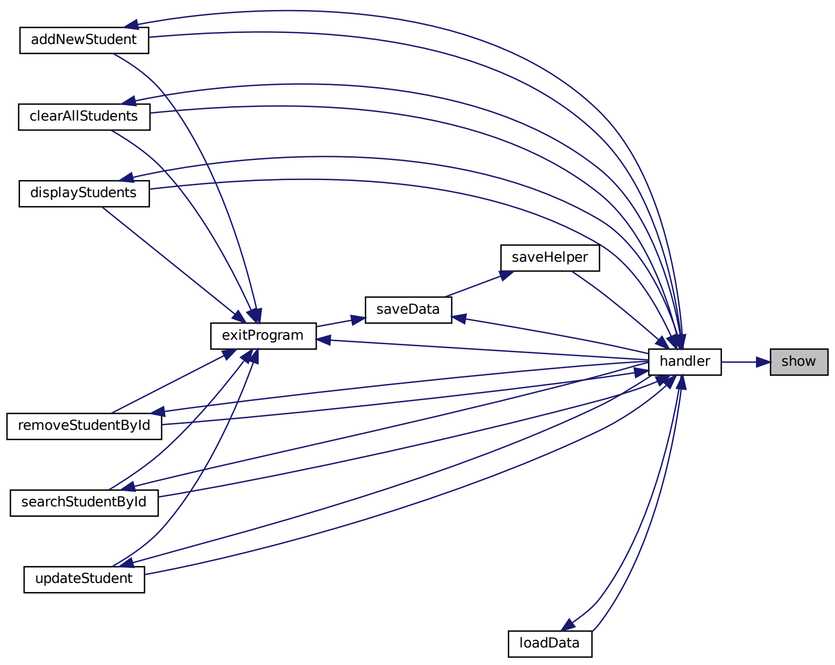
5.1.2.11 show()

```
void show ( )
```

Displays the main menu options for the Student management system.

Displays the main menu options for the Student management system.

This function prints the available options for the user to interact with the system. Here is the caller graph for this function:



5.1.2.12 updateStudent()

```
void updateStudent ( )
```

Updates an existing Student's information.

Updates an existing Student's information.

This function allows the user to update the information of a student by their ID. The user can update individual fields (Name, Age, Department, etc.) or update all fields at once. The function continues to prompt the user for further updates until they choose to exit. The name of the student to update.

The age of the student to update.

The department of the student to update.

The academic year of the student to update.

The phone number of the student to update.

The GPA of the student to update.

Iterator to traverse the list of students.

The ID of the student being searched for.

The user's choice of field to update.

Boolean to track if the student was found.

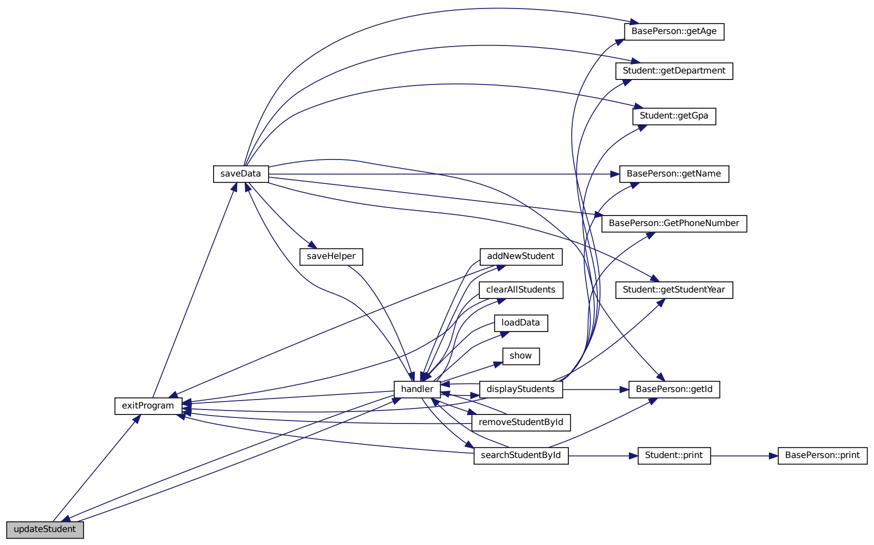
Loop through the list of students to find the matching ID.

Switch-case structure to handle the field selection and update.

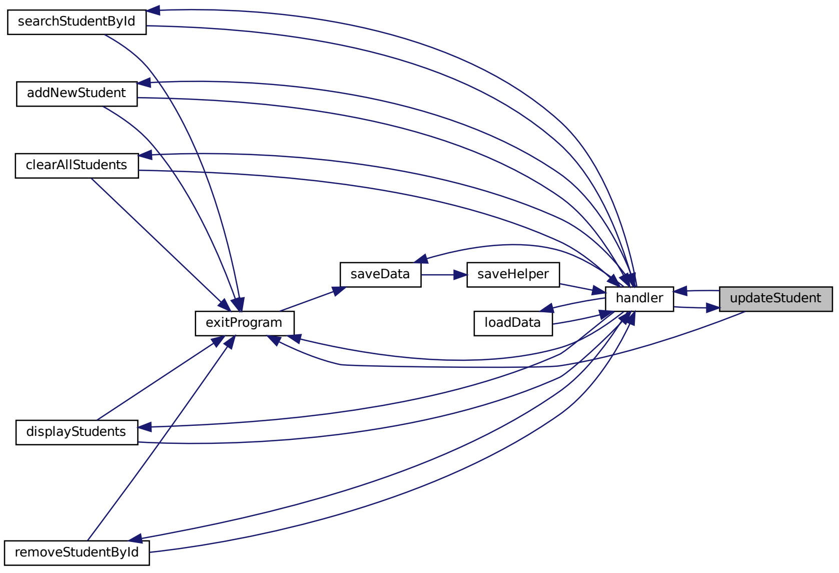
After the update is done, print the student's updated details.

If the student with the given ID was not found, notify the user.

Prompt the user to process another operation or exit the program.Here is the call graph for this function:



Here is the caller graph for this function:



5.1.3 Variable Documentation

5.1.3.1 exitStatus

```
bool exitStatus  [extern]
```

A global status variable to manage program exit status.

This variable prevents the program from prompting the user for another operation after saving data when you terminate the program. When set to zero, it indicates that data has been successfully
saved, and the program should exit immediately without asking the user if they want to perform another operation.

Index