Components del grup: Roger Casals
Grup:L5
Professors de l'assignatura: Pep Santacruz
Grau en desenvolupament d'aplicacions web i mòbil

Estrategia

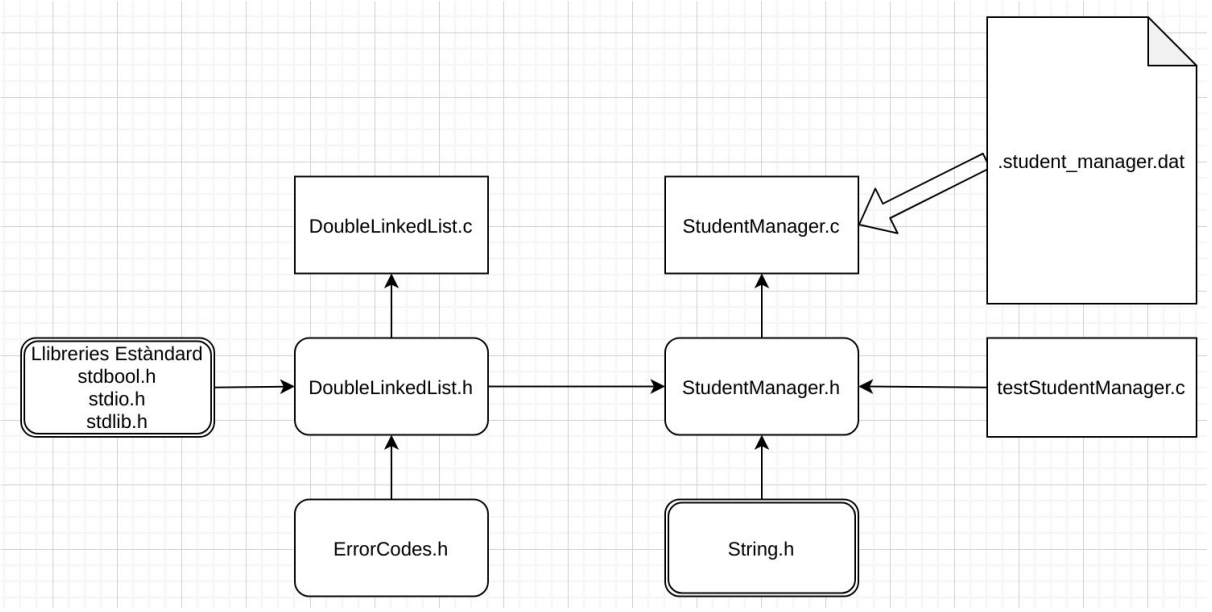
Per a aquesta pràctica hem d'implementar un manegador d'estudiants, pel que caldrà implementar una llista que contingui les dades dels estudiants. En el nostre cas, hem implementat una llista doblement encadenada amb punt d'interès. Aquesta llista contindrà les estructures de dades i funcions necessàries per a manipular l'estructura amb seguretat desde un altre programa.

Aquesta llista contindrà les dades dels estudiants, amb dades heterogènies, pel que caldrà implementar un registre que contingui aquestes dades. Cal destacar que l'estudiant conté un camp per a les seves notes que també s'ha d'implementar amb una llista encadenada, pero en aquest cas guardant reals, pel que farem servir la mateixa llista, però per la màxima de no repetir codi i no implementar dues llistes diferents farem aquesta llista genèrica. Per a aconseguir tal finalitat seguirem dues estratègies:

- Utilitzar punters a estructures de dades genèriques (`void *`) per a tractar els elements de la llista.
- Deixar forats en la implementació dependents d'una funció que es passa per paràmetre que tracti el tipus de dades específic de la llista en *runtime*. Per exemple, passar a la funció per afegir ordenadament una funció comparadora d'estudiants o una funció comparadora de reals o enters per a saber com ordenar les dades encara que tracti dades genèriques desconegudes.

Un cop implementades les operacions de les llistes la majoria del codi que queda es d'entrada sortida i de funcions que es passaran per paràmetre específiques del manegador d'estudiants i no de la llista.

Finalment, cal destacar la organització de fitxers del projecte:



- `DoubleLinkedList.c`: Conté les funcions per a manipular la llista doblement encadendada genèrica.
- `DoubleLinkedList.h`: Conté les dependencies, definició de camps de la llista, simbols i la importació dels codis d'error.
- `ErrorCodes.h`: Conté els simbols de codis d'error de sortida del programa per quan hi ha un error fatal.
- `StudentManager.h`: Conté el registre del manegador, la dependencia cap a la llibreria string, una definició per a la mida dels camps i una taula per evaluar la lletra del DNI.
- `StudentManager.c`: Conté les funcions per a passar per paràmetre en el cas d'utilitzar estudiants a la llista i el programa principal.
- `.student_manager.dat`: Conté les dades del programa principal.
- `testStudentManager.c`.

Pseudocodi de l'algorisme principal

L'algorisme s'ha implementat directament en C i no hem tingut temps de traduir tot el codi (de més de 1000 linies) a pseudocodi, pero s'hagués fet el pseudocodi en cas d'haver tingut algun procediment algorítmicament complex.

Les funcions s'han dissenyat tenint en compte el disseny descendent de forma que els blocs de codi eren massa curts per a fer un pseudocodi o bé en el cas del menú es tracta majoritàriament d'entrada-sortida, cosa que tampoc no es reflexaria en un pseudocodi. L'ús de la funció scanf i fscanf pot ser tosc degut a la impossibilitat de controlar realment el que l'usuari introduirà fins que no apreti el caràcter de nova línia. En el nostre cas ha sigut més cosa de prova i error i no un problema d'algorítmica.

Documentació i Joc de proves

L'algorisme s'ha comentat degudament en les parts menys òbvies i també s'han afegit capçaleres a les funcions de la llista doblement encadenada, que ha sigut la part més difícil.

S'ha dissenyat un script que crida a les funcions de la llista i fa proves amb ella. També s'han fet moltes proves amb el menú assegurant que no hi hagués cap resultat inesperat.

Qüestions extra

- S'han implementat un makefile per a compilar y executar el projecte amb les comandes `make` i `make run`, respectivament.
- S'ha afegit una opció que permet esborrar tota la llista.