1. What’s it?  
   It’s a repository that stores all the data (no matter what type) we want and care;
2. What attribute does it have?  
   It’s public! Everyone in this project is free to add new type data, modify data or retrieve data. For now, it has 3 levels.
3. What do we do to it?  
   Like mentioned above, we can do everything to these data. So basically, our job is to build and maintain it.
4. Where do we build it up?  
   We can do that job by setting up a specific header file. In this header file, we put this data structure in. Every group will include this header file in their project.
5. How can we update it after built up?  
   We put it on github. Everytime, after modification (i.e add new type data), we push new version and the others pull back when they start their work. By doing that, each group will have same version.
6. What ‘s the profit?  
   Save time!
7. What’s next?  
   Determine the rule of naming member, speak out what do you need and keep maintaining headfile and update log(already built on github).

Example: https://github.com/soberkiller/Unzip\_Assignment\_Project/blob/master/Unzip\_Assignment\_Ver01/Assignment\_Unzip.hh

|  |  |  |
| --- | --- | --- |
|  | #ifndef \_COURSE\_INFO // struct for course including three structs: student information, assignment information, file(assignment itself) information | |
| typedef struct{ | |
|  | | // information, assignment information, file(assignment itself) information |
|  | | typedef struct{ |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | | #ifndef \_STU\_INFO |
|  | |  |
|  | | typedef struct{ |
|  | | int Stu\_Index; // index for retrieving information of student |
|  | | int Stu\_Assigment\_Grade\_Current; // grade students will get from this assignment |
|  | | int Stu\_Assigment\_Grade\_Min; // the minimum grade that students can get |
|  | | int Stu\_Assigment\_Grade\_Max; // the maximum grade that students can get |
|  | | double Stu\_ID; // the ID number of student |
|  | | const char \*Stu\_Name; // student's name which has to be english |
|  | | const char \*Stu\_Mail\_Addr; // student's email address |
|  | |  |
|  | |  |
|  | | } Stu\_info; // stu\_info stores student information |
|  | |  |
|  | | #define \_STU\_INFO |
|  | | #endif |
|  | |  |
|  | | #ifndef \_FILE\_INFO |
|  | |  |
|  | | typedef struct { |
|  | | const char \*File\_name\_Origin[256]; // files's name |
|  | | const char \*File\_Ext\_name; // file's extra name |
|  | | // name if download group do not tell |
|  | | const char \*File\_dir\_Origin, \*File\_dir\_N; // position that file is located and the position where |
|  | | // the copies of files and unzip files are?? we are going to store these new files under the same level where orignal files are sitting |
|  | | const char \*File\_Extname\_N; // give something if u dont do anything |
|  | |  |
|  | | int file\_number; // how many files student submit |
|  | | int file\_dir\_err; // tell us whether certain file has vaild path |
|  | | int file\_name\_err; // tell us whether certain file has vaild name |
|  | | int file\_name\_valid; // whether file name is valid |
|  | | int file\_zip\_valid; // whether zipfile can be unzipped |
|  | | // before unzipping file, we have to list files (by using |
|  | | // listgetFileList(QString fileCompressed)) |
|  | | // and see whether these files follow name rule. If they dont, |
|  | | // set file\_name\_valid to 0 and file\_zip\_valid 1 |
|  | |  |
|  | |  |
|  | | } File\_info; // stores file (download from canvas, actually they are assignments) information. There are some seats reserved for class unzip, for now, we are not sure. |
|  | |  |
|  | | #define \_FILE\_INFO |
|  | | #endif |
|  | |  |
|  | | #ifndef \_ASSIGNMENT\_INFO |
|  | |  |
|  | | typedef struct { |
|  | |  |
|  | | const char \*Assigment\_Comment; // grader's comment to a assignment |
|  | | const char \*Assignment\_Graph\_Title; |
|  | | const char \*Assignment\_Title; // |
|  | | } Assignment\_info; // all the things about assignment |
|  | | #define \_ASSIGNMENT\_INFO |
|  | | #endif |
|  | | Stu\_info S\_info; |
|  | | File\_info F\_info; |
|  | | Assignment\_info A\_info; |
|  | |  |
|  | | } Course\_info; // Assigment\_information is to store all information about assigment like |
|  | |  |
|  | | #define \_COURSE\_INFO |
|  | | #endif |

Example of usage:

https://github.com/soberkiller/Unzip\_Assignment\_Project/blob/master/Unzip\_Assignment\_Ver01/main.cpp

|  |  |
| --- | --- |
|  | int HOWMANY\_STU = 10; // for simulating |
|  |  |
|  | Course\_info \*stu=new Course\_info[HOWMANY\_STU]; // HOWMANY\_STU is dynamic, given by download group |
|  |  |
|  | for(int i=HOWMANY\_STU-1; i>=0; i--){ |
|  |  |
|  | stu[i].A\_info.Assigment\_Comment = "good!"; |
|  | stu[i].F\_info.file\_number = 3; |
|  | for(int j = stu[i].F\_info.file\_number-1; j>=0; j--) // student may submit mutiple assignments |
|  | { |
|  | stu[i].F\_info.File\_name\_Origin[j] = "HW6.zip"; |
|  | } |
|  | stu[i].F\_info.File\_Ext\_name = ".cpp"; |
|  |  |
|  | stu[i].F\_info.File\_dir\_Origin = "/Users/fangmingzhao/course/Project/build-Unzip\_Assignment\_Ver01-Desktop\_Qt\_5\_7\_0\_clang\_64bit-Debug/59380\_0"; // given by download |
|  |  |
|  | stu[i].F\_info.file\_name\_valid = -1; // initialize file name ok flag; |
|  | stu[i].F\_info.file\_zip\_valid = -1; // initialize zip ok flag |
|  | // stu[i].F\_info = {"noway",".txt", "\\new1\\hello", "\\old1\\byebye", 1,1,0,0}; |
|  |  |
|  | stu[i].S\_info.Stu\_Assigment\_Grade\_Current = 90; |
|  | stu[i].S\_info.Stu\_Assigment\_Grade\_Max = 100; |
|  | stu[i].S\_info.Stu\_Assigment\_Grade\_Min = 0; |
|  | stu[i].S\_info.Stu\_ID = 10419999; |
|  | stu[i].S\_info.Stu\_Index = i; |
|  | stu[i].S\_info.Stu\_Mail\_Addr = "cplusplus@qt.com"; |
|  | stu[i].S\_info.Stu\_Name = "Michael"; |
|  |  |
|  |  |
|  | } |
|  |  |
|  | delete[] stu; |
|  |  |
|  | cout.precision(10); |
|  | cout << stu[13].S\_info.Stu\_Index << '\n'; |