3802ICT

Programming Languages

Practice-based Assignment

Assignment 1

Trimester 2, 2021

1. Background

Dataframe is a 2-dimensional labeled data structure with columns of potentially different types. You can think of it like a spreadsheet or SQL table, or a dictionary of objects. Features of DataFrame are as follows.

- Potentially columns are of different types
- Size – Mutable
- Labeled axes (rows and columns)
- Can perform arithmetic operations on rows and columns

Dataframes are useful to work with large data. It can be used to analyse and process data easily. C++ does not come with a built-in feature to create and handle dataframes. Your task is to write a C++ program to enable users to create and interact with dataframes with the following features.

- Columns are of the same type
- Size – Mutable
- Labeled columns
- Can perform arithmetic operations on columns

The Data Frame class should be declared as a generic class to accommodate type T of values as follows:

```
template<typename T>

class DataFrame {

private:

        //data_vec is the storage of the data frames

        vector<vector<T>> *data_vec;

};
```

1.1 Initialisation

A data frame can be created without any initialisation (DataFrame object construction) as follows.

DataFrame df();

However, if the user wants, he/she can initialise it as follows:

DataFrame df(vectors); //vectors is a vector of vectors.

1.2 Load Data

Assume df is a data frame of type DataFrame. A user should be able to load data into df through a vector of vectors (this may overwrite the initial data).

df.load_data(vectors); //vectors is a vector of vectors

load_data() should be able to load data in one column at a time as follows:

df.add_data(_vector);// _vector is a vector, add _vector in the dataframe without column name df.add_data(_vectors);// _vectors is a vector of vectors, add _vectors in the dataframe without column names

df.add_data(_vector, column_name); // _vector is a vector and column_name is the column name

df.add_data(_vectors, column_names); // _vectors is a vector of vectors and column_names are the column names

1.3 Setting Column Names

Assume df is a data frame of type DataFrame. A user should be able to set the column names of the data frame as follows:

string column_names[] = {"col1","col2", "col3"};

df.set_columns(column_names, size(column_names));

set_columns() must set the column names of all the existing columns in the data frame. However, a user should be allowed to add a new column anytime in the data frame with/without a column name.

1.4 Updating Column Names

A user should be able to update the column names in a data frame as follows.

df.update_column(old_name, new_name);

df.update_columns(column, name)); //column is the index and name is the column name

1.5 Updating Columns

A user should be able to update the column values as follows:

df.update_columnval(column_idx, _vector);// column_idx is the column index

df.update_columnval(column_name, _vector); // column_name is the column name

df.delete_column(column_idx); //this would drop the column

df.delete_column(column_name); //this would drop the column

1.6 Get Columns

A user should be able to access column data as follows:

df[column_name]; // column_name is the column name, string type. This should return the values in the column as a vector

df[column_idx]; // column_idx is the column id. This should return the values in the column as a vector

1.7 Data Frame Metadata

df.size();//should return the total number of entries in the data frame

df.shape();// should return the number of columns

1.8 Statistical Measures

A user should be able to perform the following statistical operations on the column data:

df[column_name].min(); // this returns the smallest value of the column

df[column_name].max(); // this returns the largest value of the column

df[column_name].mean(); // this returns the mean value the column data

df[column_name].median(); // this returns the median value the column data

df[column_name].mode(); // this returns the mode value of the column data

df[column_name].summary(); // this returns all the above values together as a map where the keys are the names of the measures such as “min”, “max”, “mean” “median” and “mode”.

2. Constraints

1. You are allowed to use C++ built-in libraries including Standard Template Libraries (STLs).
2. All cin and cout must be implemented as scan(), print() and println() via ***lambda function***.
3. The program should print an error message for a user if it does not fit within the constraints of the data frame, e.g., a user is trying to update a column name or access a column which does not exist in the data frame. [***Hints***: Use the try-catch feature of C++].
4. The developed program will be judged based on the use of generics (the more generics, the better - e.g., use of templates), reduced line of codes (LoCs), auto ***type deducibility*** and use of ***functional programming features*** such as lambda expressions and ***concepts***.
5. A generic requirement is to write readable code, refactoring and well-commented code.

3. Marking Criteria

Section	Mark Detail	Section Total
Initialisation	Dataframe creation with and without data: 1 + 1	2
Load Data	Load full data, load column data, and load column data with/without column name: 2 + 1 + 2	5
Setting Column Names	Setting column names and exception handling: 2 + 1	3
Updating Columns	Update column and delete column: 1 + 1	2
Get Columns	Get column values with column_name and index: 2 + 2	4
Data Frame Metadata	Calculating size and shape of the dataframe: 1 + 2	3
Statistical Measures	Calculating min, max, mean, median, mode and summary: 1 + 1 + 1 + 1 + 1 + 1	6
Total		**25**

Good luck ☺