

МIНIСТЕРСТВО ОСВIТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**Лабораторна робота №2**

з дисципліни “Основи програмування”

тема “Рядки символів і формат даних CSV”

|  |  |  |
| --- | --- | --- |
| Виконав  студент I курсу  групи КП-92  Мовчан Максим Олександрович  (*прізвище, ім’я, по батькові*)  варіант № 9 |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладач  Гадиняк Руслан Анатолійович  (*прізвище, ім’я, по батькові*) |

Київ 2020

**Мета роботи**

Ознайомитись із стандартним типом std::string.  
Опанувати формат даних CSV та навчитись його зчитувати і формувати.  
Навчитись збирати проекти з багатьох вихідних файлів за допомогою CMake.

**Постановка завдання**

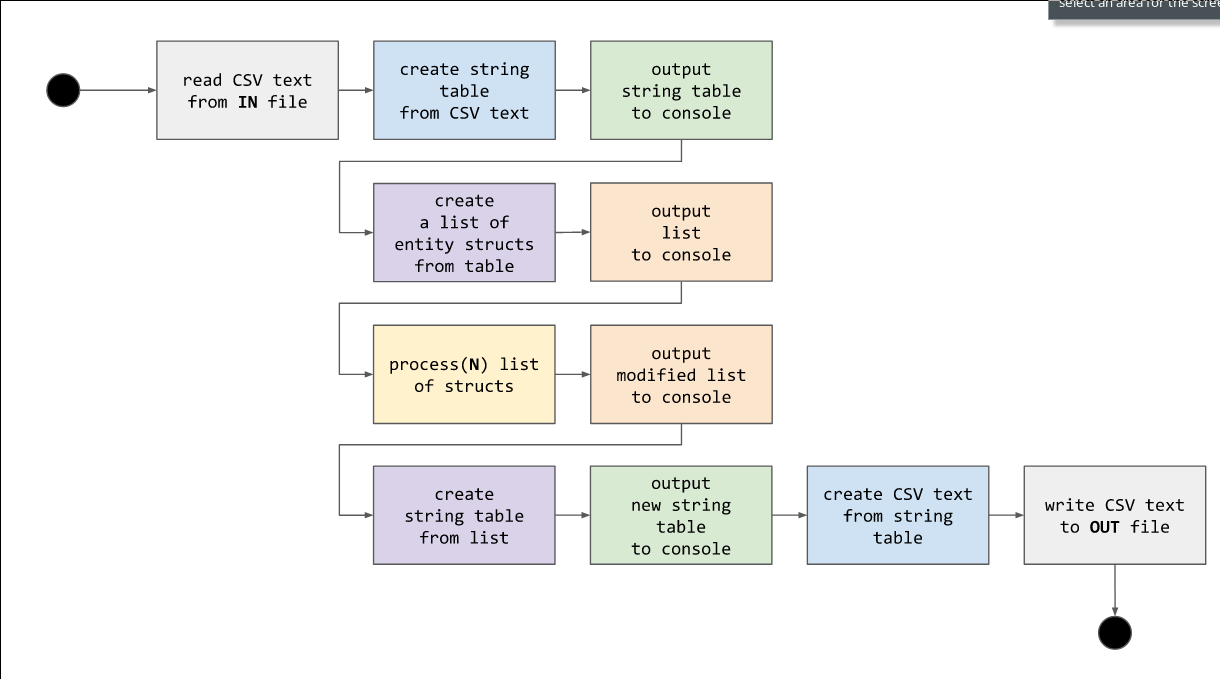
Створити консольну програму, що дозволяє працювати з CSV файлами.

Зчитати CSV дані з файлу data.csv у таблицю рядків.  
Вивести отриману таблицю у консоль (форматування при виводі довільне).  
Сформувати на основі таблиці список сутностей типу Організація.  
Вивести список сутностей у консоль (в довільному форматі).

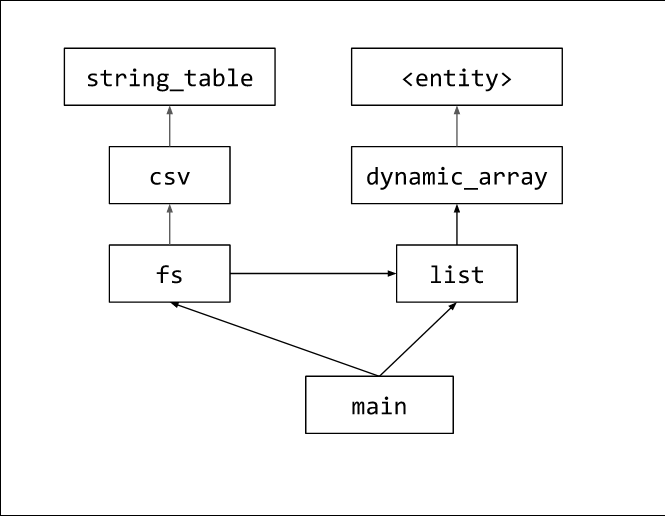
Ввести значення N.  
Після формування списку сутностей Знайти всі організації із країни N.  
Видалити зі списку сутностей зайві елементи.  
Вивести модифікований список сутностей у консоль.

Перетворити модифікований список сутностей у нову таблицю.  
Вивести нову таблицю у консоль.  
Зберегти нову таблицю у форматі CSV у вихідний файл out.csv

**Блок-схема алгоритму**



**Діаграма залежностей**



**Інтерфейси модулів**

|  |
| --- |
| **Dynamic\_array.h** |
| class DynamicArray {  Organisation \*items\_;  int capacity\_;  public:  DynamicArray(int capacity);  ~DynamicArray();  int size();  void resize(int capacity);  Organisation &operator[](int i);  }; |
| **csv.h** |
| #include "string\_table.h"  using namespace std;  // parse csv, init and fill string table  StringTable Csv\_parse(string& csvStr);  // create string and fill it with csv text  string Csv\_toString(StringTable& table); |
| **fs.h** |
| #include "csv.h"  #include <cstdio>  #include <cstring>  #include <fstream>  #include <iostream>  using namespace std;  void Check\_CountOfArgs( int arg, int num );  void Check\_isOpen( ifstream \* file );  void Check\_isClosed( ifstream \* file );  StringTable Table\_fromFile( string & file\_name );  void File\_fromTable(StringTable &table, string & file\_name); |

|  |
| --- |
| **list.h** |
| class List {  DynamicArray \_array;  int size\_;  public:  List();  ~List();  int size();  void insert(int index, Organisation value);  void remove\_at(int index);  void push\_back(Organisation value);  void remove(Organisation value);  int index\_of(Organisation value);  bool contains(Organisation value);  bool empty();  void clear();  void print();  Organisation &operator[](int i);  int getMaxWidth(int val = 0);  }; |
| **string\_table.h** |
| class StringTable {  std::string \* cells\_ = nullptr; // allocated memory for matrix  int nrows\_ = 0;  int ncols\_ = 0;  public:  StringTable(int rows, int cols);  ~StringTable();    int size\_rows();  int size\_columns();    string & at(int rowIndex, int colIndex);    void print();  }; |
| **Organisations.h** |
| struct Organisation {  string country;  string label;  string type;  string founders;  int founded\_date;  string NASDAQ;  }; |

**Вхідні/вихідні дані**

|  |
| --- |
| **data.csv** |
| country,label,type,founder(s),founded,NASDAQ  US,"Google inc.",Public Company,"Larry Page, Sergey Brin",1998,GOOG  UK,"EPAM Systems",Public Company,"Arkadiy Dobkin, Leo Lozner",1993,EPAM  US,"Microsoft Corporation",Public Company,"Bill Gates, Paul Allen",1975,MSFT  UK,"Apple Inc.",Public Company,"Steve Jobs, Steve Wozniak, Ronald Wayne",1976,AAPL |
| **Terminal** |
| *[country][label][type][founder(s)][founded][NASDAQ]*  *[US][Google inc.][Public Company][Larry Page, Sergey Brin][1998][GOOG]*  *[UK][EPAM Systems][Public Company][Arkadiy Dobkin, Leo Lozner][1993][EPAM]*  *[US][Microsoft Corporation][Public Company][Bill Gates, Paul Allen][1975][MSFT]*  *[UK][Apple Inc.][Public Company][Steve Jobs, Steve Wozniak, Ronald Wayne][1976][AAPL]*  *Country Label Type Founders Founded NASDAQ*  *US "Google inc." Public Company "Larry Page, Sergey Brin" 1998 GOOG*  *UK "EPAM Systems" Public Company "Arkadiy Dobkin, Leo Lozner" 1993 EPAM*  *US "Microsoft Corporation" Public Company "Bill Gates, Paul Allen" 1975 MSFT*  *UK "Apple Inc." Public Company "Steve Jobs, Steve Wozniak, Ronald Wayne" 1976 AAPL*  *SIZE: 4*  *CAPACITY: 5*  *Input country to filter:*  *UK*  *Country Label Type Founders Founded NASDAQ*  *UK "EPAM Systems" Public Company "Arkadiy Dobkin, Leo Lozner" 1993 EPAM*  *UK "Apple Inc." Public Company "Steve Jobs, Steve Wozniak, Ronald Wayne" 1976 AAPL*  *SIZE: 2*  *CAPACITY: 5*  *Writed to file successfully* |
| **out.csv** |
| country,label,type,founders,founded\_date,NASDAQ  UK,"EPAM Systems",Public Company,"Arkadiy Dobkin, Leo Lozner",1993,EPAM  UK,"Apple Inc.",Public Company,"Steve Jobs, Steve Wozniak, Ronald Wayne",1976,AAPL |

**Тексти коду програм**

|  |
| --- |
| **main.cpp** |
| //#include <unistd.h> //sleep|usleep  //#include <iomanip> //formatted output  #include "../HEADERS/csv.h"  #include "../HEADERS/dynamic\_array.h"  #include "../HEADERS/fs.h"  #include "../HEADERS/list.h"  #include "../HEADERS/string\_table.h"  #include <cstring>  #include <fstream>  #include <iostream>  using namespace std;  List createEntityListFromTable(StringTable &csvTable);  StringTable processEntities(List &items,  Organisation n); // T depends on your variant  int main(int argc, char \*argv[]) {  Check\_CountOfArgs(argc, 3);  string csv\_file = argv[1];  string out\_file = argv[2];  StringTable table = Table\_fromFile(csv\_file);  table.print();  List list = createEntityListFromTable(table);  list.print();  string input;  cout << "Input country to filter: " << endl;  cin >> input;  Organisation org;  org.country = input;  StringTable table2 = processEntities(list, org);  list.print();  File\_fromTable(table2, out\_file);  cout << "Writed to file successfully" << endl;  }  int checknum(string str) {  if (isdigit(str[0]))  return stoi(str);  else  cout << "\nDate is not num\n";  exit(1);  }  List createEntityListFromTable(StringTable &table) {  List list;  Organisation org;  for (int i = 1; i < table.size\_rows(); i++) {  org.country = table.at(i, 0);  org.label = table.at(i, 1);  org.type = table.at(i, 2);  org.founders = table.at(i, 3);  org.founded\_date = checknum(table.at(i, 4));  org.NASDAQ = table.at(i, 5);  list.push\_back(org);  }  return list;  }  StringTable processEntities(List &items, Organisation n) {  for (int i = items.size() - 1; i >= 0; i--)  if (items[i].country != n.country)  items.remove\_at(i);  StringTable table(items.size() + 1, 6);  for (size\_t i = 0; i < items.size(); i++) {  table.at(i + 1, 0) = items[i].country;  table.at(i + 1, 1) = items[i].label;  table.at(i + 1, 2) = items[i].type;  table.at(i + 1, 3) = items[i].founders;  table.at(i + 1, 4) = to\_string(items[i].founded\_date);  table.at(i + 1, 5) = items[i].NASDAQ;  }  table.at(0, 0) = "country";  table.at(0, 1) = "label";  table.at(0, 2) = "type";  table.at(0, 3) = "founders";  table.at(0, 4) = "founded\_date";  table.at(0, 5) = "NASDAQ";  return table;  } |
| **dynamic\_array.cpp** |
| #include "../HEADERS/dynamic\_array.h"  DynamicArray::DynamicArray(int capacity) {  items\_ = new Organisation[capacity];  capacity\_ = capacity;  }  DynamicArray::~DynamicArray() {  delete[] items\_;  }  Organisation & DynamicArray::operator[](int i) {  return items\_[i];  }  void DynamicArray::resize(int capacity) {  Organisation \* tmparr = new Organisation[capacity];  for(int i = 0; i < capacity\_; i++) tmparr[i] = items\_[i];  delete[] items\_;  items\_ = tmparr;  capacity\_ = capacity;  }  int DynamicArray::size() {  return capacity\_;  } |
| **list.cpp** |
| #include "../HEADERS/list.h"  #define color1 "\033[31m"  #define color2 "\033[32m"  #define color3 "\033[33m"  #define color4 "\033[34m"  #define color5 "\033[35m"  #define color6 "\033[36m"  #define color7 "\033[37m"  #define restt "\033[0m"  List::List(): \_array {5} {  size\_ = 0;  }  List::~List() {}  int List::size() {  return size\_;  }  Organisation& List::operator[](int i) {  return \_array[i];  }  void List::insert(int index, Organisation value) {  \_array.resize(\_array.size() + 1);  for (int i = 0; size\_ - i > index; i++)  \_array[size\_ - i] = \_array[size\_ - i - 1];  \_array[index] = value;  size\_++;  }  void List::remove\_at(int index) {  for (int i = index; i < size\_ - 1; i++) {  \_array[i] = \_array[i + 1];  }  size\_--;  }  void List::push\_back(Organisation value) {  \_array[size\_] = value;  size\_++;  if (size\_ == \_array.size())  \_array.resize(\_array.size() + 1);  }  void List::remove(Organisation value) {  for (int i = 0; i < size\_; i++) {  if (\_array[i] == value) {  remove\_at(i);  }  }  }  int List::index\_of(Organisation value) {  for (int i = 0; i < size\_; i++) {  if (\_array[i] == value)  return i;  }  return -1;  }  bool List::contains(Organisation value) {  for (int i = 0; i < size\_; i++) {  if (\_array[i] == value)  return true;  }  return false;  }  bool List::empty() {  return size\_ == 0;  }  void List::clear() {  size\_ = 0;  }  void List::print() {  int l1 = 5;  for (int i = 0; i < size\_; i++) {  if (\_array[i].label.length() > l1)  l1 = \_array[i].label.length();  }  int l2 = 8;  for (int i = 0; i < size\_; i++) {  if (\_array[i].founders.length() > l2)  l2 = \_array[i].founders.length();  }  cout << color1 << left << setw(9) << "Country" << restt << color2 << left  << setw(l1 + 3) << "Label" << restt << color4 << left << setw(16)  << "Type" << restt << color5 << setw(l2 + 3) << "Founders" << color6  << setw(9) << left << "Founded" << color7 << "NASDAQ" << restt << endl;  for (int i = 0; i < size\_; i++) {  cout << color1 << left << setw(9) << \_array[i].country << restt << color2  << left << setw(l1 + 3) << \_array[i].label << restt << color4 << left  << setw(16) << \_array[i].type << restt << color5 << setw(l2 + 3)  << \_array[i].founders << color6 << setw(9) << \_array[i].founded\_date  << color7 << \_array[i].NASDAQ << restt << endl;  }  cout << color3 << "SIZE: " << restt << size() << endl;  cout << color3 << "CAPACITY: " << restt << \_array.size() << endl;  }  bool operator==(Organisation o1, Organisation o2) {  return o1.country == o2.country;  } |
| **fs.cpp** |
| #include "../HEADERS/fs.h"  void Check\_CountOfArgs( int arg, int num ) {  if( arg < num ) {  perror( "\033[;;41mWRONG COUNT OF ARGUMENTS\033[0m\n" );  exit( EXIT\_FAILURE );  }  }  void Check\_isOpen( ifstream \* file ) {  if( !file->is\_open() ) {  perror( "\033[;;41mCAN'T OPEN FILE\033[0m\n" );  exit( EXIT\_FAILURE );  }  }  void Check\_isClosed( ifstream \* file ) {  if( file->is\_open() ) {  perror( "\033[;;41mCAN'T CLOSE FILE\033[0m\n" );  exit( EXIT\_FAILURE );  }  }  StringTable Table\_fromFile( string & file\_name ) {  ifstream file( file\_name );  Check\_isOpen( &file );  string from\_file, temp;  while( getline( file, temp ) ) {  from\_file += temp + "\n";  }  file.close();  Check\_isClosed( &file );  return Csv\_parse( from\_file );  }  void File\_fromTable(StringTable &table, string & file\_name) {  ofstream file( file\_name );  if(!file.is\_open()) {  perror( "\033[;;41mCAN'T OPEN FILE\033[0m\n" );  exit( EXIT\_FAILURE );  }  string out = Csv\_toString(table);  file << out <<endl;  return;  }  /\*void check\_csv( char \* arg ) {  std::string format = &arg[strlen( arg ) - 4];  if( format != ".csv" ) {  perror( "\033[;;41mFILE MUST BE .csv\033[0m\n" );  exit( EXIT\_FAILURE );  }  }\*/ |
| **csv.cpp** |
| #include "../HEADERS/csv.h"  StringTable Csv\_parse( string & csvStr ) {  const char \* begin = csvStr.c\_str();  bool quotes = 0;  int mcols = 0, rows = 0, cols = 0;  for( const char \* p = begin; \*p; p++ ) {  if( \*p == '"' ) quotes = !quotes;  else if( \*p == ',' and quotes == 0 ) {  cols++;  begin = p + 1;  } else if( \*p == '\n' ) {  if( cols > mcols ) mcols = cols;  cols = 0;  rows++;  begin = p + 1;  }  }  mcols++;  StringTable table{ rows, mcols };  begin = csvStr.c\_str();  quotes = 0, cols = 0, rows = 0;  for( const char \* p = begin; \*p; p++ ) {  if( \*p == '"' ) quotes = !quotes;  else if( \*p == ',' and quotes == 0 ) {  table.at( rows, cols ) = string( begin, p - begin );  cols++;  begin = p + 1;  } else if( \*p == '\n' ) {  table.at( rows, cols ) = string( begin, p - begin );  for( cols++; cols < mcols; cols++ ) {  table.at( rows, cols ) = "";  }  cols = 0;  rows++;  begin = p + 1;  }  }  return table;  }  string Csv\_toString( StringTable & table ) {  string tmp;  for( int i = 0; i < table.size\_rows(); i++ ) {  for( int j = 0; j < table.size\_columns(); j++ ) {  tmp += table.at( i, j ) + ",";  }  tmp[tmp.length() - 1] = '\n';  }  return tmp;  } |
| **String\_table.cpp** |
| #include "../HEADERS/string\_table.h"  StringTable::StringTable(int rows, int cols) {  cells\_ = new string[rows \* cols];  nrows\_ = rows;  ncols\_ = cols;  }  StringTable::~StringTable() {}  int StringTable::size\_rows() {  return nrows\_;  }  int StringTable::size\_columns() {  return ncols\_;  }  std::string& StringTable::at(int rowIndex, int colIndex) {  return cells\_[ncols\_ \* rowIndex + colIndex];  }  void StringTable::print() {  for (size\_t i = 0; i < size\_rows(); i++) {  for (size\_t j = 0; j < size\_columns(); j++) {  string to\_filter = at(i, j);  for (size\_t i = 0; i < to\_filter.length(); i++) {  if (to\_filter[i] == '"') {  to\_filter.erase(i, 1);  }  }  cout << '[' << to\_filter << ']';  usleep(10000);  }  cout << endl;  }  } |

**Висновки**

У лабораторній роботі були використані структури даних на основі динамічного масиву. Виділення пам’яті було виконано за допомогою оператора new та звільнена за допомогою оператор delete[].

Для виводу у консоль був використаний заголовочний файл <iostream>, об’єкти cout, endl з простору імен std та оператор вставки у потік <<.

Для зчитування файлу був використаний getline.

Компіляція коду відбувалася за допомогою CMake.