Міністерство освіти і науки України

Національний Технічний Університет

«Харківський Політехнічний Інститут»

Кафедра «Стратегічного Керування»

Лабораторна робота № 3

«Використання засобів Стандартної бібліотеки C++»

Перевірила: ас. кафедри СК

Вільхівская О. В.

Виконав:

Харків, 2018

Лабораторна робота № 3

Тема: Використання засобів Стандартної бібліотеки C++

Завдання

1. Вектор векторів для представлення двовимірного масиву

Розв'язати завдання 1.2 [першої лабораторної роботи](http://iwanoff.inf.ua/algorithmization_2/LabTraining01.html) (умови нижче) з використанням вектора векторів Стандартної бібліотеки. Для ініціалізації рядків використати ініціалізатори як фактичні параметри конструкторів векторів. Для всіх циклів, у яких не потрібне явне значення індексу, використовувати циклічну конструкцію **for**, побудовану на діапазоні.

*Завдання 1.2 першої лабораторної роботи.*

Розробити клас для представлення двовимірного масиву (матриці) цілих чисел довільних розмірів. Створити конструктори та деструктор, перевантажити операції додавання, віднімання і множення (згідно з правилами роботи з матрицями), звертання за індексом, введення з потоку та виведення в потік. Створити власні класи винятків та генерувати відповідні об'єкти-винятки, якщо неможливо виконати ту чи іншу операцію.

Створити окрему функцію, яка отримує посилання на матрицю і виконує над масивом дії, вказані в таблиці. Функція не повинна бути методом класу або дружньою функцією.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Номер варіанту  (номер студента у списку)** | | **Правило перетворення елементів масиву** | **Кількість рядків m** | **Кількість стовпців n** |
| 2 |  | Усі елементи з парними значеннями повинні бути замінені їх квадратами | 3 | 5 |

1. Представлення й обробка даних про студентів з використанням засобів Стандартної бібліотеки

Створити клас для представлення даних про студента. Клас повинен містити такі елементи даних:

1. номер залікової книжки (**unsigned int**);
2. прізвище (рядок типу std::string);
3. оцінки за останню сесію (std::vector).

Всередині класу реалізувати функції доступу, а також об'явити конструктор, який ініціалізує елементи даних, та функції, які здійснюють:

* обчислення показника, за величиною якого здійснюється сортування відповідно до індивідуального завдання;
* перевірку умови, яка використовується для пошуку даних відповідно до індивідуального завдання.

Здійснити опис класу для представлення групи студентів. В об'єкті такого класу повинні зберігатись дані про студентів у вигляді вектора об'єктів класу, який представляє студента. Клас повинен містити перевантажені операції введення-виведення, а також функції-елементи, які здійснюють

* сортування масиву за ознакою, яка наведена в індивідуальному завданні (з використанням алгоритму sort());
* пошуку даних про студентів, які відповідають умові, наведеній в індивідуальному завданні (з використанням алгоритму for\_each()).

Розмістити об'єкти класу "Студент" в черзі з пріоритетом, з якої вилучати об'єкти в порядку зменшення середнього балу.

Індивідуальні завдання:

|  |  |  |
| --- | --- | --- |
| **№ студента у списку** | **Умова сортування** | **Умова вибору даних** |
| 2 | За збільшенням середнього балу | З довжиною прізвища більше 7 літер |

Хід роботи

1. Вектор векторів для представлення двовимірного масиву

|  |
| --- |
| #include <iostream>  #include <vector>  using std::cout;  using std::vector;  using std::cin;  using std::endl;  using std::ostream;  using std::istream;  const int ROWS = 3; //Personal value of rows  const int COLUMNS = 5; //Personal value of columns  vector<vector<int>> arrCreation(int row, int col) {  vector<vector<int>> arr(row);  for (vector<int> &i : arr) i.resize(col); //Fils elements of a subvector with zeros creating a column  return arr;  }  class Matrix {  friend Matrix operator\*(Matrix& a, Matrix& b) {  if (a.col != b.row) {  throw NotMatch(a, b);  }  Matrix c(a.row, b.col);  for (int i = 0; i < a.row; i++) {  for (int k = 0; k < b.col; k++) {  int sum = 0;  for (int j = 0; j < a.col; j++) {  sum += a[i][j] \* b[j][k];  }  c.addElem(sum);  }  }  return c;  }  friend Matrix operator+(Matrix& a, Matrix& b) {  if ((a.row == b.row) && (a.col == b.col)) {  Matrix c(a.row, a.col);  for (int i = 0; i < a.row; i++) {  for (int j = 0; j < a.col; j++) {  c[i][j] = a[i][j] + b[i][j];  }  }  return c;  }  throw NotMatch(a, b);  }  friend Matrix operator-(Matrix& a, Matrix& b) {  if ((a.row == b.row) && (a.col == b.col)) {  Matrix c(a.row, a.col);  for (int i = 0; i < a.row; i++) {  for (int j = 0; j < a.col; j++) {  c[i][j] = a[i][j] - b[i][j];  }  }  return c;  }  throw NotMatch(a, b);  }  friend ostream& operator<<(ostream& out, Matrix& a) {  for (vector<int> &row : a.arr) {  for (int &x : row) out << x << "\t";  out << "\n";  }  return out;  }  friend Matrix operator>>(istream &in, Matrix& a) {  cout << "Enter parameters (rows, columns): ";  in >> a.row >> a.col;  a.arr=arrCreation(a.row,a.col);  a.size=a.row\*a.col;  cout << "Enter " << a.size << " elements in a row" << endl;  int x;  for (int i = 0; i < a.size; i++) {  cin >> x;  a.addElem(x);  }  return a;  }  private:  int row, col;  vector<vector<int>> arr;  int size;  int count = 0;  void addElem(int value) {  arr[count / col][count%col] = value;  count++;  }  public:  vector<vector<int>>& getArr() { return arr; }  int getRow() { return row; }  int getCol() { return col; }  int getSize() { return size; }  Matrix() {}  Matrix(int row, int col) : arr(arrCreation(row,col)), row(row), col(col), size (row\*col) {}    class NotMatch {  int row1, row2, col1, col2;  public:  NotMatch(Matrix& a, Matrix& b) : row1(a.row), row2(b.row), col1(a.col), col2(b.col) {}  int getRow1() { return row1; }  int getRow2() { return row2; }  int getCol1() { return col1; }  int getCol2() { return col2; }  };    class Proxy{  private:  vector<int>& \_arr;  public:  Proxy(vector<int>& \_arr) : \_arr(\_arr){}  int& operator[](int index2){  return \_arr[index2];  }  ~Proxy(){}  };  Proxy operator[](int index){  return Proxy(arr[index]);  }    ~Matrix() {  }  };  void squares(Matrix& a) {  for (int i = 0; i < a.getRow(); i++) {  for (int j = 0; j < a.getCol(); j++) {  if ((i + j) % 2 == 0) a.getArr()[i][j] \*= a.getArr()[i][j];  }  }  }  int main() {  Matrix a(ROWS,COLUMNS); // Rows and columns are constant values  a[0][0]=77; // Adding elements  a[0][1]=77;  cout << "Matrix a: " << endl << a << endl; // Displays matrix  squares(a); // Modifies matrix according to the personal task  cout << "Matrix a changed: " << endl << a << endl; // Displays modified matrix  Matrix b(2,2);  b[0][1]=7;  b[1][0]=7;  b[1][1]=7;  cout << "Matrix b: " << endl << b;    cout << "Matrix c=a+b: " << endl;  try{  Matrix c = a+b;  cout << c;  }  catch(Matrix::NotMatch ex){  cout << "Not Match " << ex.getRow1() << "x" << ex.getCol1() << " & " << ex.getRow2() << "x" << ex.getCol2() <<endl;  }  Matrix e(2, 4);  e[0][0] = 1;  e[0][1] = 2;  e[0][2] = 3;  e[0][3] = 7;  cout << "Matrix e: " << endl << e << endl;  Matrix f(4, 3);  f[0][0] = 1;  f[1][0] = 2;  f[0][1] = 5;  f[2][1] = 6;  f[2][0] = 3;  cout << "Matrix f: " << endl << f << endl;  system("pause");  try  {  Matrix d = e \* f;  cout << "Matrix d = e\*f: " << endl << d << endl;  }  catch (Matrix::NotMatch ex)  {  cout << "Not Match " << ex.getCol1() << "!=" << ex.getRow2() << endl;  }  system("pause");  } |

Приклад виконання програми:

|  |
| --- |
| Matrix a:  77 77 0 0 0  0 0 0 0 0  0 0 0 0 0  Matrix a changed:  5929 77 0 0 0  0 0 0 0 0  0 0 0 0 0  Matrix b:  0 7  7 7  Matrix c=a+b:  Not Match 3x5 & 2x2  Matrix e:  1 2 3 7  0 0 0 0  Matrix f:  1 5 0  2 0 0  3 6 0  0 0 0  Press any key to continue . . . |

1. Представлення й обробка даних про студентів з використанням засобів Стандартної бібліотеки

|  |
| --- |
| #include <iostream>  #include <vector>  #include <string>  #include <algorithm>  using namespace std;  class Student{  private:  unsigned int rbook;  string lname;  vector<int> marks;  public:  Student(){}  Student(unsigned int rbook, string lname, vector<int> marks) : rbook(rbook), lname(lname), marks(marks){}    unsigned int getRbook() const{return rbook;}  string getName() const{return lname;}  vector<int> getMarks() const{return marks;}    double Average() const{  int sum=0;  for (int i : marks) sum+=i;  return (double)sum/marks.size();  }  void search (){  if (lname.length()>7){  cout << lname << " / " << rbook << " / ";  for (int x : marks) cout << x << " ";  cout << endl;  }  }  bool operator()(const Student& s1, const Student& s2) const{  return s1.Average() < s2.Average();  }  void setRbook(int value){rbook=value;}  void setName(string value){lname=value;}  void setRbook(vector<int> value){marks=value;}  };    class Group{  private:  friend ostream& operator<<(ostream& out, Group& gr){  for (Student i : gr.a){  out << i.getName() << " / " << i.getRbook() << " / ";  for (int x : i.getMarks()) cout << x << " ";  cout << " / Average grade: " << i.Average() << endl;  }  return out;  }  friend istream& operator>>(istream& in, Group& gr){  cout << "Enter number of record book, last name, and 6 marks: " << endl;  int rbook;  string lname;  vector<int> marks;  int mark;  cin >> rbook >> lname;  for (int i=0; i<6; i++){  cin >> mark;  marks.push\_back(mark);  }  gr.a.push\_back(Student(rbook, lname, marks));  }  vector<Student> a;  public:  Group(){}  Group(vector<Student> a) : a(a) {};  void search(){  for\_each(a.begin(),a.end(),mem\_fun\_ref(Student::search));  }  void sorting(){  sort(a.begin(), a.end(), Student());  }  };  int main() {  Student Zar(123,"Zarchenko",{4,5,4,5,4,4});  Student Ser(451,"Sergeev",{4,5,4,5,4,4});  Student Bez(270,"Bezkrovniy",{5,5,4,5,4,5});  Student Sin(54,"Sinetutov",{2,2,2,2,2,2});    Group gr(vector<Student>{Zar,Ser,Bez,Sin});  cout << "Using for\_each with condition determined in class Student (last name contains more than 7 letters): " << endl;  gr.search();  cout << endl << "Printing all information that object of class Group contains: " << endl;  cout << gr;  gr.sorting();  cout << endl << "Sorted array(vector) by the average grade of objects of the Group class: " << endl;  cout << gr;    cout << endl;  cin >> gr; //Adding a new student  gr.sorting();  cout << gr;  system("pause");  } |

Приклад виконання програми:

|  |
| --- |
| Using for\_each with condition determined in class Student (last name contains more than 7 letters):  Zarchenko / 123 / 4 5 4 5 4 4  Bezkrovniy / 270 / 5 5 4 5 4 5  Sinetutov / 54 / 2 2 2 2 2 2  Printing all information that object of class Group contains:  Zarchenko / 123 / 4 5 4 5 4 4 / Average grade: 4.33333  Sergeev / 451 / 4 5 4 5 4 4 / Average grade: 4.33333  Bezkrovniy / 270 / 5 5 4 5 4 5 / Average grade: 4.66667  Sinetutov / 54 / 2 2 2 2 2 2 / Average grade: 2  Sorted array(vector) by the average grade of objects of the Group class:  Sinetutov / 54 / 2 2 2 2 2 2 / Average grade: 2  Zarchenko / 123 / 4 5 4 5 4 4 / Average grade: 4.33333  Sergeev / 451 / 4 5 4 5 4 4 / Average grade: 4.33333  Bezkrovniy / 270 / 5 5 4 5 4 5 / Average grade: 4.66667  Enter number of record book, last name, and 6 marks:  777 Litov4enko 5 5 5 5 5 5  Sinetutov / 54 / 2 2 2 2 2 2 / Average grade: 2  Zarchenko / 123 / 4 5 4 5 4 4 / Average grade: 4.33333  Sergeev / 451 / 4 5 4 5 4 4 / Average grade: 4.33333  Bezkrovniy / 270 / 5 5 4 5 4 5 / Average grade: 4.66667  Litov4enko / 777 / 5 5 5 5 5 5 / Average grade: 5  Press any key to continue . . . |

Висновок

Під час виконання цієї лабораторної роботи я ознайомився з визначення контейнерів – класи, що дозволяють створювати колекції інших об’єктів з набором функцій для здійснення стандартних алгоритмів. Краще використовувати контейнери ніж масиви, бо вони представляють собою перевірені класи, та їх використання дозволяє уникнути помилок.

За допомогою ітераторів ми можемо проходитися по контейнеру. Ітераторам присвоюється вказіник на елемент послідовності.

У другому завданні для сортування було використано шаблонну функцію sort, перевантаження оператору дужок і функціональний об’єкт класу з перевантаженим оператором.