МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний технічний університет
«Харківський політехнічний інститут»

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторної роботи

«Використання класу vector у програмах мовою С++»

з дисципліни «Алгоритмізація та програмування»
для студентів спеціальності 124 «Системний аналіз»
і дисципліни «Інформатика і програмування» для студентів
спеціальності 186 «Видавництво та поліграфія»

Затверджено
редакційно-видавничою
радою університету,
протокол № 1 від 15.02.2024 р.

Харків
НТУ «ХПІ»
2024

Методичні вказівки до виконання лабораторної роботи «Використання класу vector у програмах мовою C++» з дисципліни «Алгоритмізація та програмування» для студентів спеціальності 124 «Системний аналіз» і дисципліни «Інформатика і програмування» для студентів спеціальності 186 «Видавництво та поліграфія» / уклад. М. І. Безменов. – Харків : НТУ «ХПІ», 2024. – 20 с.

Укладач М. І. Безменов

Рецензент С. В. Шевченко

Кафедра системного аналізу та інформаційно-аналітичних технологій

ВСТУП

Велика кількість задач передбачає обробку масивів, одно- або двовимірних, а в багатьох випадках багатовимірних. Засоби мов С і С++ дозволяють писати програми обробки масивів, у тому числі з використанням динамічної пам'яті для розміщення масивів. При цьому функції обробки масивів є загальними функціями, призначеними для загального їх використання. У С++, крім звичайних динамічних масивів, є можливість використання векторів, що є об'єктами класу `vector`, який дозволяє не тільки створювати аналоги динамічних масивів, але й має у своєму визначенні сукупність функції для створення і обробки таких об'єктів.

Мета роботи – освоєння методики використання об'єктів класу `vector` у програмах мовою С++.

1. ТЕОРЕТИЧНІ ОСНОВИ

1.1. Загальні положення

Як альтернатива динамічного масиву, у С++ визначено вектори – об'єкти класу `vector`. Будучи стандартним компонентом, клас `vector` знаходиться в просторі імен `std`, а визначення його шаблону – у заголовному файлі `<vector>`, який повинен бути підключений для забезпечення можливості оперування векторами:

```
#include <vector>
```

Як і у випадку масиву, усі елементи вектора мають той самий тип. Визначення вектора виконується аналогічно визначенню звичайних змінних – вказівкою типу і імені. При цьому як тип зазначається слово `vector`, слідом за яким у кутових дужках зазначається тип елементів вектора:

```
vector <тип> ім'я_змінної;
```

Наприклад,

```
vector <int> v;
```

При цьому створюється аналог динамічного масиву, розмір якого дорівнює 0.

Одночасно з оголошенням вектор може бути ініціалізований аналогічно тому, як ініціалізується масив, – переліченням значень його

елементів після знаку рівності (який не є обов'язковим) у фігурних дужках через кому; наприклад,

```
vector <int> v1 = {2, -3, 4}, v2 {5, 6};
```

У такому разі створюється вектор, кількість елементів у якому дорівнює кількості значень, перелічених у фігурних дужках, а самі елементи – указаним значенням.

Клас `vector` постачений вбудованими функціями, які дозволяють виконувати дії над векторами-об'єктами. Такими діями є, наприклад, додавання нових елементів, вставка елементів, їх видалення тощо. Звертання до таких функцій здійснюється застосуванням операції «крапка», першим операндом якої є ім'я об'єкта, а другим – ім'я функції із звичайним переліченням фактичних параметрів:

```
ім'я_об'єкта-вектора.ім'я_функції(фактичні_параметри)
```

Наприклад, для визначеного вище вектора `v1` можна записати:

```
v1.push_back(-1);
```

Заголовний файл `vector` разом з визначенням класу `vector` містить визначення типів, що використовуються у визначенні вбудованих у клас функцій. Для забезпечення доступу до таких типів використовується бінарна операція `::`, лівим операндом якої є ім'я класу `vector`, а правим – ім'я типу; наприклад, `vector::pointer`.

Задля пришвидшення роботи програми у разі видалення елементів вектора перерозподіл пам'яті не виконується – елементи переписуються всередині ділянки пам'яті, яка була відведена під вектор, а в кінці цієї ділянки залишається вільне місце. При додаванні нових елементів до вектора, якщо це можливо, здійснюється перезапис у вже відведеній ділянці пам'яті. Якщо її обсягу недостатньо, то в динамічній пам'яті автоматично відводиться безперервна ділянка пам'яті, розмір якої відповідає новому розміру вектора, і в неї переписується старий його вміст разом з новим, а стара ділянка пам'яті переходить у розряд вільної. Такий підхід значно прискорює роботу програми, якщо збільшення розміру вектора виконується більш менш активно.

1.2. Конструктори

Клас `vector`, які і будь-який інший клас, має у своєму визначенні декілька конструкторів – функцій (методів), які автоматично викликаються при створенні об'єктів для їх початкової ініціалізації. Конструктори

мають одне й те ж ім’я, яке збігається з ім’ям класу (у даному випадку `vector`). Виклик конструктора здійснюється за рахунок зазначення в дужках його фактичних параметрів одразу за ім’ям оголошуваного об’єкта. При цьому в динамічній пам’яті створюється об’єкт, який ініціалізується у відповідності до переліку вказаних параметрів. Конструктор може бути не вказаний. У такому разі викликається так званий конструктор за умовчанням. Конструктор за умовчанням класу `vector` створює вектор нульової довжини.

У класі `vector` визначені, наприклад, такі конструктори:

```
explicit vector(size_type кількість);
```

– створює вектор, кількість елементів якого дорівнює параметру кількість із значеннями за умовчанням (0 у базовому типі вектора);

```
vector(size_type кількість, тип& значення);
```

– створює вектор, кількість елементів якого дорівнює параметру кількість, причому всі елементи мають те саме значення, що дорівнює другому параметру;

```
vector(const vector& джерело);
```

– створює вектор, що є копією вектора, заданого параметром джерело, з використанням пам’яті, відведеної під параметр.

1.3. Індексування

Як і елементи масиву, елементи вектора індексуються з початковим значенням індексу 0. Індекс елементу зазначається у квадратних дужках.

Є ще один варіант звертання до елементу вектора – використання функції `at`, що має такий формат:

```
at(size_type позиція);
```

Наприклад, звертання `v1[1]` і `v1.at(1)` є ідентичними.

Ще приклади:

```
v1[0] = -27;
v1.at(1) += v1.at(0);
std::cout << v1.at(1);
```

1.4. Ітератори

Опрацювання векторів можливе за допомогою покажчиків, на які орієнтовані функції класу `vector`. У класі `vector` визначені спеціальні

покажчики, що забезпечують доступ до елементів вектора з можливістю послідовного їх перебору за рахунок автоматичного змінення значення такого покажчика в циклі. Такі покажчики мають назву «***ітератори***». За допомогою ітераторів можливий різний доступ до елементів вектора – для читання, довільний доступ (читання і змінення) з переходом від першого елемента вектора в напрямку останнього або у зворотному порядку. При цьому тип ітератора визначається типом елементів вектора.

Визначені такі ітератори:

`iterator` – ітератор довільного доступу для читання і змінення;

`const_iterator` – ітератор довільного доступу для читання;

`const_reverse_iterator` – ітератор, що забезпечує читання і перехід від останнього елементу в напрямку першого;

`reverse_iterator` – ітератор для читання та змінення з забезпеченням переходу від останнього елементу в напрямку першого.

Крім ітераторів, у класі `vector` визначені й деякі інші типи.

1.5. Деякі функції класу vector

Клас `vector` містить близько 30 функцій різного призначення. Якщо ж урахувати те, що більшість з них має декілька однойменних перевантажень, то функцій значно більше. Деякими з них є такі:

`size_type capacity() const;` – повертає кількість елементів, які можуть бути поміщені у вектор без перерозподілу пам'яті.

`size_type size() const;` – повертає кількість елементів у векторі;

`void clear();` – очищує вектор;

`void resize(size_type кількість);` – визначає новий розмір вектора, додаючи нові елементи в кінець або видаляючи їх в кінці вектора залежно від співвідношення між запитаним і наявним розмірами;

`const_pointer data() const;` – повертає покажчик на перший елемент непорожнього вектора (тип `const_pointer` – визначений у класі `vector` тип, що забезпечує доступ до елементів вектора);

`bool empty() const;` – перевіряє, чи є вектор порожнім, повертаючи **`true`**, якщо вектор порожній, і **`false`** у противному випадку;

`const_iterator begin() const;` – повертає ітератор на константу, що вказує на перший елемент **непорожнього** вектора;

`const_iterator begin();` – повертає ітератор довільного доступу, що вказує на перший елемент **непорожнього** вектора;

`iterator end();` – повертає ітератор, що вказує на елемент, що знаходиться слідом за останнім елементом вектора;

`const_iterator end() const;` – повертає ітератор, що вказує на елемент, що знаходиться слідом за останнім елементом вектора;

`const_reference front() const;` – повертає посилання на перший елемент об'єкта-вектора (тип `const_reference` визначений у класі `vector` як у деякому сенсі універсальний тип);

`const_reference back() const;` – повертає посилання на останній елемент вектора;

`void pop_back();` – видаляє останній елемент вектора;

`void push_back(тип значення);` – дописує елемент у кінець вектора;

`iterator erase(const_iterator позиція);` – видаляє елемент вектора, на який вказує параметр-ітератор;

`iterator erase(const_iterator перший, const_iterator останній);` – видаляє діапазон елементів, починаючи з позиції `перший` і закінчуючи позицією `останній`, повертаючи позицію вставки;

`iterator insert(const_iterator позиція, тип значення);` – вставляє задане другим параметром `значення` в `позицію`, що задається першим параметром-покажчиком, повертаючи позицію вставки;

`void insert(const_iterator позиція, size_type кількість, const тип значення);` – вставляє вказане третім параметром `значення` в задану першим параметром `позицію` з повторенням цього значення в `кількості`, що задається другим параметром-покажчиком.

1.6. Імітація багатовимірних масивів

Об'єкт-вектор є аналогом одновимірного динамічного масиву. Оскільки тип елементів вектора може бути довільним, можна створювати об'єкти-вектори, елементи яких у свою чергу є векторами, що відкриває можливість для імітування двовимірних масивів і масивів більшої розмірності. Наприклад, двовимірний масив імітує об'єкт що має такий опис:

```
vector<vector<тип>> ім'я_об'єкта;
```

Наприклад,

```
vector<vector<double>> figureArea;
```

Такий об’єкт індексується звичайним чином, а саме, зазначенням двох індексів, кожного у своїх квадратних дужках. Тим самим забезпечується доступ до елемента «двовимірного масиву». Наприклад, для описаного вище об’єкта `figureArea` виконуваними є такі оператори:

```
area.push_back(vector<double>()); // Створення рядка
// довжиною 0 (використаний конструктор по умовчанню).
area[0].push_back(3.1415);  // Додано один елемент.
area[0].push_back(25.1234); // Додано другий елемент.
std::cout << area[0][0] << ' ' << area[0][1];
int n;
std::cin >> n;
area.push_back(vector<double>(n)); // Створення другого
// рядка з n елементів типу double.
for (vector::size_type j = 0; j < area[1].size(); j++)
    std::cin >> area[1][j];
```

У даному випадку створено «масив» з рядками, що мають різну довжину, причому ітератори не використовувалися.

Якщо необхідно створити прямокутний масив, то це можна зробити так:

```
int m, n; // Змінні, що відповідатимуть за кількість
// рядків та кількість стовпців «масиву».
std::cin >> m >> n;
vector<vector<double>> area(m, vector<double> (n)));
```

Тут оголошено вектор `area` з векторів з елементами типу **`double`**. При цьому створення вектора `area` виконується конструктором, що створює `m` елементів, кожен з яких є вектором, який у свою чергу створюється конструктором, що відводить пам’ять під `n` елементів типу **`double`**.

***

Слід пам’ятати, що в разі доступу до елементів, які знаходяться поза діапазону вектора, виникає помилка часу виконання. Тому необхідно контролювати порожність вектор. Якщо вектор порожній, то в такому разі `vector::begin() == vector::end()`. Крім того, якщо ітератор є покажчиком на константу, змінення вектора з доступом за таким покажчиком є неможливим і теж призводить до помилки.

2. ПРИКЛАДИ ПРОГРАМ

Приклад 1. Дано натуральне число n і послідовність дійсних чисел a_1, a_2, …, a_n. Вивести члени послідовності в порядку, зворотному порядку введення.

Розв'язок

```
#include <iostream>
#include <vector>
#include <conio.h>

using namespace std;

int main()
{
    int n;
    cout << " Enter the number of sequence members (n): ";
    cin >> n;
    // Створення вектора з n елементів типу double.
    vector<double> a(n);
    // Цикл уведення вмісту вектора з використанням ітератора.
    for (vector<double>::iterator iter = a.begin();
         iter != a.end(); iter++)
    {
        cin >> *iter;
    }
    // Цикл зсуву ітератора від останнього елементу вектора до
    // першого. Насправді починати перегляд треба з a.end() – 1
    // й закінчувати позицією a.begin() з доступом *iter. Але
    // тоді після останньої ітерації за рахунок iter--
    // виконається спроба доступу до елемента перед першим
    // елементом вектора.
    for (vector<double>::iterator iter = a.end();
         iter > a.begin(); iter--)
    {
        cout << *(iter - 1) << '\t';
    }
    cout << "\nPress any key to exit.\n";
    int ch = _getch();
    return 0;
}
```

Значно простішою буде програма, що не орієнтована на використання ітераторів:

```
#include <iostream>
#include <vector>
#include <conio.h>

using namespace std;

int main()
{
    int n;
    cout << " Enter the number of sequence members (n): ";
    cin >> n;
    // Створення вектора з n елементів типу double.
    vector<double> a(n);
    // Цикл уведення вмісту вектора з використанням індексу.
    for (int i = 0; i < n; i++)
        cin >> a[i];
    for (int i = n – 1; i >= 0; i--)
        cout << a[i] << '\t';
    cout << "\nPress any key to exit.\n";
    int ch = _getch();
    return 0;
}
```

Приклад 2. Дано натуральні числа *m*, *n* і двовимірний масив дійсних чисел, що має *m* рядків і *n* стовпців. Вивести цей масив на екран за стовпцями, поданими у вигляді екранних рядків.

Розв’язок

```
#include <iostream>
#include <vector>
#include <conio.h>

using namespace std;
int main(void)
{
    int m, n;
    cout << "Enter rows count: ";
    cin >> m;
    cout << "Enter columns count: ";
```

```
    cin >> n;
    // Створення вектора, що має m елементів, кожен з яких є
    // вектором з n елементами типу double.
    vector<vector<double>> a(m, vector<double> (n));
    // Аналогом цього оператору є такий код:
    // vector<vector<double>> a(m, vector<double>));
    // for (vector<double>::size_type i = 0; i < a.size(); i++)
    //    a[i] = vector<double>(n);
    for (int i = 0; i < m; i++)
        for (int j = 0; j < n; j++)
        {
            cout << "Enter a[" << i << "][" << j << "]: ";
            cin >> a[i][j];
        }
    cout << "Source:";
    for (int i = 0; i < m; i++)
    {
        cout << endl;
        for (int j = 0; j < n; j++)
            cout << a[i][j] << '\t';
    }
    cout << "\nResult:";
    for (int j = 0; j < n; j++)
    {
        cout << endl;
        for (int i = 0; i < m; i++)
            cout << a[i][j] << '\t';
    }
    cout << "\nPress any key to exit.\n";
    int ch = _getch();
    return 0;
}
```

Приклад 2. Дано натуральне число n, дійсний n-вимірний вектор $\boldsymbol{x}$ і n-вимірну симетричну відносно головної діагоналі квадратну матрицю $\boldsymbol{A}$. Обчислити добуток матриці $\boldsymbol{A}$ і вектора $\boldsymbol{x}$: $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$. Координати вектора $\boldsymbol{y}$ обчислюються за формулою $y_i = \sum_{j=1}^{n} a_{ij} x_j,\ i = 1,2,\ldots,n$. Вважати, що матриця $\boldsymbol{A}$ задана елементами нижнього трикутника під головною діагоналлю і на ній.

Розв'язок

```
#include <iostream>
#include <vector>
#include <conio.h>

using namespace std;
int main(void)
{
    int n;
    cout << "Enter dimension: ";
    cin >> n;
    // Створення двох векторів, що мають по n елементів
    // типу double.
    vector<double> x(n), y(n);
    cout << "Enter " << n << " vector x coordinates:\n";
    // Уведення з використанням ітератора.
    for (vector<double>::iterator iter = x.begin();
         iter < x.end(); iter++)
        cin >> *iter;
    // Створення і введення трикутної матриці A.
    // Спочатку створюємо вектор a, що має n елементів, кожен
    // з яких є вектором з елементами типу double.
    vector<vector<double>> a(n, vector<double>());
    // Далі в циклі створюємо вектори з елементами типу double.
    // Кількість елементів створюваних векторів залежить від
    // значення параметру циклу і збільшується від 1 до n з
    // кроком 1, що забезпечує створення структури, яка імітує
    // нижній трикутник симетричної матриці A. Тут же
    // виконується введення елементів.
    cout << "Enter the elements of the lower triangle of a:\n";
    for (int i = 0; i < n; i++)
    {
        a[i] = vector<double>(i + 1);
        for (int j = 0; j <= i; j++)
        {
           cout << "Enter A[" << i + 1 << "][" << j + 1 << "]:";
           cin >> a[i][j];
        }
    }
    cout << "Matrix A:";
```

```
    for (int i = 0; i < n; i++)
    {
        cout << endl;
        for (int j = 0; j < n; j++)
            if (i <= j)
                cout << a[j][i] << '\t';
            else
                cout << a[i][j] << '\t';
    }
    for (int i = 0; i < n; i++)
    {
        y[i] = 0;
        for (int j = 0; j < n; j++)
            if (i <= j)
                y[i] += a[j][i] * x[j];
            else
                y[i] += a[i][j] * x[j];
    }
    cout << "\nResult:\n";
    for (int i = 0; i < n; i++)
            cout << y[i] << '\t';
    cout << "\nPress any key to exit.\n";
    int ch = _getch();
    return 0;
}
```

Зазначимо наступне. У наведеній програмі продемонстровано використання ітератора для опрацювання об’єкта-вектора (для введення координат вектора $\boldsymbol{x}$) і доступ за допомогою індексування. Показано також, що двовимірний масив не обов’язково повинен бути прямокутним. У такому масиві рядки можуть мати різну кількість елементів. У програмі «матриця» $\boldsymbol{A}$ має рядки, довжина яких дорівнює його номеру. У загальному випадку можна зберігати довжини рядків у окремому векторі.

3. ЗАВДАННЯ НА ЛАБОРАТОРНУ РОБОТУ

За час, відведений для виконання лабораторної роботи (2 академічні години), студент повинен:

1. Написати програми для розв’язання наведених нижче задач.

2. Здійснити налаштування програми, виправивши синтаксичні та логічні помилки.
3. Оформити звіт до лабораторної роботи.
4. Відповісти на контрольні запитання.

Примітка. Процес прийому лабораторної роботи передбачає можливість деякої модифікації умов для забезпечення деякої варіативності і перевірки самостійності виконання роботи.

4. ВАРІАНТИ ЗАДАЧ

1. Дано натуральні числа m, n і цілі числа $a_1, a_2, \ldots, a_m, b_1, b_2, \ldots, b_n, k$. Якщо в послідовності $a_1, a_2, \ldots, a_m$ нема жодного члена зі значенням k, то перший за порядком член цієї послідовності, що є не меншим за усіх інших членів, замінити на значення k. За таким же правилом перетворити послідовність $b_1, b_2, \ldots, b_n$ відносно значення 10. Визначити функції перевірки наявності в послідовності цілих чисел елемента з деяким значенням і перетворення послідовності за описаним правилом.
2. Дано натуральне число n і дійсні числа $r_1, r_2, \ldots, r_n$. Розвернути в послідовності $r_1, r_2, \ldots, r_n$ найдовшу підпослідовність, що містить тільки додатні числа. Визначити функцію формування нової послідовності.
3. Дано натуральні числа m, n і дійсні числа $a_1, a_2, \ldots, a_m, b_1, b_2, \ldots, b_n$. У заданих послідовностях усі члени, що йдуть за членом з найбільшим значенням (за останнім по порядку, якщо їх декілька) замінити на 0,5. Визначити дві функції – пошуку останнього за порядком найбільшого елемента послідовності й заміни деяким значенням усіх елементів послідовності, починаючи з елемента із заданим номером.
4. Дано натуральне число n і цілі числа $a_1, a_2, \ldots, a_n$. Отримати дві послідовності, перша з яких повинна містити всі парні члени вихідної послідовності, а друга – усі непарні. Визначити функцію переформування однієї послідовності у дві інші. У нових послідовностях члени повинні йти в порядку, протилежному початковому.
5. Дано натуральне число n й цілі числа $a_1, a_2, \ldots, a_n$. Розглянути у послідовності $a_1, a_2, \ldots, a_n$ усі відрізки (послідовності членів, що йдуть підряд), які містять тільки:

а) парні числа;

б) степені п’ятірки;

в) прості числа.

У кожному випадку отримати найдовший підмасив. Визначити функції формування масивів, які містять вказані підмасиви.

6. Дано натуральне число n й дійсні числа a_1, a_2, …, a_n. Сформувати два масиви, у перший з яких переписати за неспаданням усі додатні члени цієї послідовності, а у другий – теж за неспаданням усі від’ємні члени. Визначити функції формування вказаних масивів.
7. Дано натуральне число n і послідовність з n цілих чисел. Сформувати масив з усіх простих чисел, що містяться в заданій послідовності між її максимальним і мінімальним елементами, записавши елементи в тому ж порядку, що й у початковій послідовності. Визначити функцію формування шуканого масиву.
8. Дано натуральне число n і послідовність цілих чисел a_1, a_2, …, a_n. Отримати нову послідовність, вставивши в первинну після кожного парного числа його половину і видаливши з початкової послідовності всі непарні значення. Визначити функцію формування нової послідовності.
9. Дано натуральне число n і масив з n цілих чисел. Сформувати новий масив, що містить тільки ті елементи первинного масиву, які не є простими числами. Визначити функцію формування нового масиву.
10. Дано натуральне числа n і послідовність цілих чисел a_1, a_2, …, a_n. Видалити в цій послідовності найбільший і найменший члени. Якщо в послідовності декілька найбільших або найменших елементів, то видалити всі такі члени. Визначити функцію видалення.
11. Дано натуральне число n і цілі числа b_1, b_2, …, b_n. З’ясувати, чи є серед чисел a_1, a_2, …, a_n співпадаючі, і видалити всі такі елементи, залишивши тільки неповторювані значення. Визначити функцію видалення.
12. Дано натуральне число n і послідовність цілих чисел a_1, a_2, …, a_n. Видалити з послідовності всі нульові члени і впорядкувати нову послідовність так, щоб у ній спочатку йшли за неубуванням усі від’ємні елементи, а потім за незростанням усі додатні елементи. Визначити функцію формування нової послідовності.

13. Дано натуральні числа m, n і цілі числа $a_1, a_2, \ldots, a_m, b_1, b_2, \ldots, b_n$. Сформувати нову послідовність із членів послідовності $a_1, a_2, \ldots, a_m$, що не входять у послідовність $b_1, b_2, \ldots, b_n$. Визначити функцію формування нової послідовності.
14. Дано натуральне число n і послідовність дійсних чисел $a_1, a_2, \ldots, a_n$. Залишити без зміни послідовність $a_1, a_2, \ldots, a_n$, якщо вона впорядкована за неубуванням або незростанням; у противному разі видалити з неї ті члени, порядкові номери яких кратні трьом, зберігши порядок членів, що залишаться. Визначити функцію формування нової послідовності.
15. Дано натуральне число n і послідовність відмінних від нуля цілих чисел $a_1, a_2, \ldots, a_n$. Якщо в послідовності числа з різними знаками чергуються, видалити з неї усі додатні члени; у противному разі змінити на протилежний знаки усіх членів послідовності. Визначити дві функції – видалення з послідовності додатних членів і зміни знаків її членів.
16. Дано натуральне число n і цілі числа $a_1, a_2, \ldots, a_n$. *Досконалим* зветься таке натуральне число, яке дорівнює сумі всіх своїх простих дільників. Знайти в послідовності $a_1, a_2, \ldots, a_n$ усі відрізки елементів, що йдуть підряд і містять тільки досконалі числа. Визначити функцію формування двовимірного масиву з рядками різної довжини, які містять указані вище відрізки (підпослідовності) натуральних чисел.
17. Дано натуральні числа m, n, p, а також дві матриці розміру $m \times n$ і одна матриця розміру $n \times p$. Отримати добуток першої і третьої матриць, а також суму перших двох. Визначити функції множення і підсумовування двох прямокутних матриць.
18. Дано натуральні числа m, n і квадратну матрицю $\boldsymbol{A}$ порядку m. Отримати матрицю $\boldsymbol{E} + \boldsymbol{A} + \boldsymbol{A}^2 + \ldots + \boldsymbol{A}^n$, де $\boldsymbol{E}$ – одинична матриця порядку m. Визначити дві функції – множення двох квадратних матриць і підсумовування двох квадратних матриць одного порядку.

    Одинична матриця – це квадратна матриця, усі елементи головної діагоналі якої дорівнюють одиниці, а всі інші – нулю.
19. Дано прямокутну матрицю. Помножити цю матрицю на транспоновану і навпаки. Визначити функції транспонування матриці й множення двох матриць.

20. Дано двовимірний числовий масив з рядками різної довжини. Упорядкувати цей масив за зростанням довжин рядків, визначивши функцію впорядкування.
21. Дано числовий масив з n елементів. Переписати у двовимірний масив з рядками різної довжини всі впорядковані за зростанням підпослідовності з елементів одновимірного масиву. Визначити функцію формування масиву з вказаними властивостями.
22. Дано натуральне число n і дві трикутні матриці $\boldsymbol{A}$ і $\boldsymbol{B}$ порядку n, у першій з яких нульові елементи містяться під головною діагоналлю (рис. 1), а в другій – під побічною (рис. 2). Обидві матриці задані послідовностями рядків, довжина першого з яких дорівнює n, другого – $(n-1)$ і т. д. Сформувати дві такі матриці й отримати їх суму.

    Визначити функції формування вказаних трикутних матриць як масивів з рядками різної довжини.

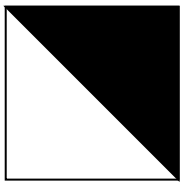

Рис. 1. Права верхня трикутна матриця $\boldsymbol{A}$

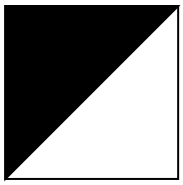

Рис. 2. Ліва верхня трикутна матриця $\boldsymbol{B}$

23. Дано натуральні числа m, n_1, n_2, …, n_m, де m – кількість рядків, а n_j – кількість елементів у j-му рядку ($j = 1, 2, \ldots, m$) двовимірного масиву дійсних чисел з рядками різної довжини. Дано масив із вказаними характеристиками. Упорядкувати рядки масиву за неспаданням їх максимальних елементів. Визначити функцію впорядкування вмісту масиву.

5. КОНТРОЛЬНІ ЗАПИТАННЯ

1. Що таке «об’єкт-вектор» і як він визначається?
2. Яке призначення векторів?
3. Як створюється об’єкт-вектор?
4. Які функції виконують конструктори?
5. Як здійснюється звертання до визначених у класі `vector` типів?
6. Як виконується виклик функцій класу `vector`?

7. Як іменуються конструктори?
8. Як здійснюється доступ до елементів вектора?
9. Охарактеризуйте поняття ітератора.
10. Чи можете Ви назвати деякі з ітераторів класу `vector` і для чого вони використовуються?
11. Як дізнатися кількість елементів вектора і його місткість?
12. Для чого призначена функція `clear`?
13. Як можна дізнатися про наявність елементів у векторі?
14. Яка функція забезпечує додавання нового елементу в кінець вектора?
15. Як видаляється останній елемент вектора?
16. Як здійснюється видалення елементів вектора в загальному випадку?
17. Опишіть призначення функцій `insert`.
18. Опишіть методику створення багатовимірного масиву за допомогою класу `vector`. У чому особливість такої структури даних?

СПИСОК ЛІТЕРАТУРИ

1. Безменов М. І., Безменова О. М., Калінін Д. В. Лабораторний практикум із програмування : навч. посіб. Харків: НТУ «ХПІ», 2015. 368 с. ISBN 978-617-7294-53-4. URL: https://repository.kpi.kharkov.ua/server/api/core/bitstreams/0aac9932-2972-4f67-8ded-ca8bc558a4c1/content (дата звернення: 10.02.2024).
2. Рудий Т. В., Паранчук Я. С., Сеник В. В. Алгоритмізація та програмування. Частина 1. Структурне програмування: навчальний посібник. Львів : Львівський державний університет внутрішніх справ, 2023. 240 с. URL: https://dspace.lvduvs.edu.ua/bitstream/1234567890/5515/1/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D1%96%D0%B7%D0%B0%D1%86%D1%96%D1%8F...--%D1%87.%201----%D0%92%D0%95%D0%A0%D0%A1%D0%A2%D0%9A%D0%90.pdf (дата звернення: 05.10.2023).
3. Ришковець Ю. В., Висоцька В. А. Алгоритмізація та програмування. Частина 1: навчальний посіб-ник. Львів : Видавництво «Новий Світ-2000», 2020. 337 с. URL: http://library.kpi.kharkov.ua/files/new_postupleniya/atprc1.pdf (дата звернення: 05.10.2023).

4. Ришковець Ю. В., Висоцька В. А. Алгоритмізація та програмування. Частина 2: навчальний посіб-ник. Львів : Видавництво «Новий Світ-2000», 2020. 314 с. URL: http://library.kpi.kharkov.ua/files/new_postupleniya/atprc2.pdf (дата звернення: 05.10.2023).
5. Довгунь А. Я., Ватаманіца Е. В., Ушенко Ю. О. Алгоритмізація та програмування: навч. посіб. Чернівці: Чернівецький нац. ун-т, 2022. 293 с. URL: https://archer.chnu.edu.ua/bitstream/handle/123456789/6739/%d0%90%d0%bb%d0%b3%d0%be%d1%80%d0%b8%d1%82%d0%bc%d1%96%d0%b7%d0%b0%d1%86%d1%96%d1%8f_%d1%82%d0%b0_%d0%bf%d1%80%d0%be%d0%b3%d1%80%d0%b0%d0%bc%d1%83%d0%b2%d0%b0%d0%bd%d0%bd%d1%8f.pdf?sequence=1 (дата звернення: 05.10.2023).
6. Гришанович Т. О., Глинчук Л. Я. Основи об’єктно-орієнтованого програмування : навч. посібник. Луцьк : ВНУ імені Лесі Українки, 2022. 120 с. URL: https://evnuir.vnu.edu.ua/bitstream/123456789/20320/1/oop.pdf (дата звернення: 05.10.2023).
7. Schildt H. C++: The Complete Reference. McGraw Hill, 2002. 1056 p. URL: https://ia601805.us.archive.org/15/items/cplusplus-books/C%2B%2B%20The%20Complete%20Reference%2C%204th%20Edition.pdf (дата звернення: 05.10.2023).
8. Prata S. C Primer Plus. Addison-Wesley Professional, 2013. 1072 p. URL: https://www.cl72.org/070documents/C/c-primer.pdf (дата звернення: 05.10.2023).

Навчальне видання

Методичні вказівки
до виконання лабораторної роботи
«Використання класу vector у програмах мовою C++»
з дисципліни «Алгоритмізація та програмування» для студентів спеціальності 124 «Системний аналіз» і дисципліни «Інформатика і програмування» для студентів спеціальності 186 «Видавництво і поліграфія»

Укладач:
БЕЗМЕНОВ Микола Іванович

Відповідальний за випуск Ю. І. Дорофєєв
Роботу до видання рекомендував І. П. Гамаюн
Комп’ютерна верстка М. І. Безменов

У авторській редакції

План 2024 р., поз. 180

Підп. до друку 20.02.2024 р. Формат 60×84 1/16. Папір офсетний.
Друк – цифровий. Гарнітура Times New Roman. Ум. друк. арк. 1,58.
Наклад 50 пр. Зам. № 44. Ціна договірна

Видавничий центр НТУ «ХПІ»
Свідоцтво про державну реєстрацію ДК № 5478 від 21.08.2017 р.
61002, Харків, вул. Кирпичова, 2