Міністерство освіти і науки України

Національний університет “Львівська Політехніка”

Кафедра ЕОМ



**ЗВІТ**

З лабораторної роботи № 2

З дисципліни: “Програмування, частина 2 (Об'єктно-орієнтоване програмування)”

На тему: “Потоковий ввід-вивід”

Варіант 15

Виконав: ст. гр. КІ-15

Марков Л.Ю.

Прийняв:

Козак Н.Б.

Львів – 2020

Мета: познайомитися із потоковим вводом-виводом.

**ТЕОРЕТИЧНІ ВІДОМОСТІ**

Простори імен

У мові програмування С++ так само як і у багатьох об‘єктноорієнтованих мовах програмування існує поняття просторів імен, що розділяються, на відміну від мови С в якій існує один глобальний простір імен. Простори імен служать для об‘єднання класів, що написані різними програмістами або мають схоже призначення в групи. Завдяки цьому з‘являється можливість усувати неоднозначності, що пов‘язані з використанням класів, що мають однакові імена, але різне призначення. Також простори імен дозволяють безпечно підключати бібліотеки класів не боячись співпадінь у назвах існуючих і підключених класів. Все, що оголошене в межах простору імен є видимим лише в його межах. Для оголошення простору імен слід вжити ключове слово namespace після якого слід вказати назву простору імен. Після цього в фігурних дужках визначається його тіло. Тіло простору імен може містити як класи і структури, так і функції зі змінними та константами, а також вкладені простори імен.

Простори імен є відкритими. Це означає, що можна в будь-якому місці програми оголосити простір імен з існуючою назвою, додати в нього нові елементи і при компіляції ці простори імен об‘єднаються в один, що міститиме існуючі і нововведені елементи. Простір імен може бути безіменним, якщо його ім‘я при оголошенні не вказується. Безіменні простори імен можна використовувати для оголошення глобальних статичних змінних з внутрішнім зв‘язуванням, оскільки згідно стандарту ключове слово static є застарілим.

Щоб постійно не вказувати простори імен при зверненні до їх елементів, їх можна підключити до програми за допомогою ключового слова using частково або повністю. Для підключення окремих елементів простору імен використовується using оголошення. Для підключення всього простору імен з використовується using директива.

При підключені всього простору імен ми можемо звичним чином звертатися до всіх його елементів. Проте, при підключенні кількох просторів імен, що мають елементи з однаковою назвою до яких відбувається звертання, відбудеться конфлікт імен. Для його усунення слід явно вказувати до якого простору імен належить елемент до якого іде звертання. Простори імен можуть бути вкладеними. Оголошення вкладених просторів імен відбувається шляхом визначення простору імен в просторі імен.

Потоки вводу-виводу в С++

Система вводу-виводу в стандартній бібліотеці С++ реалізована у вигляді потоків. Потік вводу-виводу – це логічний пристрій, який приймає та видає інформацію користувача. Кожен потік зв‘язаний з фізичним пристроєм (клавіатура, монітор) або з файлом. Бібліотека потоків iostream реалізована як ієрархія класів та забезпечує широкі можливості для виконання операцій вводу-виводу. Далі наведено призначення деяких класів потокового вводу-виводу:

* istream – підтримує операції по вводу;
* ostream – підтримує операції по виводу;
* iostream – підтримує операції по вводу-виводу;
* іfstream – підтримує операції по вводу з файлу;
* ofstream – підтримує операції по виводу у файл;
* fstream – підтримує операції з файлами по вводу-виводу.

Стандартні потоки

Коли запускається програма на С++, автоматично створюються чотири об‘єкти, що реалізують стандартні потоки.

Таблиця 2.1. Об‘єкти стандартних потоків

|  |  |  |
| --- | --- | --- |
| **Потік** | **Призначення** | **Пристрій по замовчуванню** |
| cin | Стандартний ввід | Клавіатура |
| cout | Стандартний вивід | Монітор |
| cerr | Стандартний вивід повідомлень про помилки | Монітор |
| clog | Стандартний вивід повідомлень про помилки (буферизований) | Монітор |

Щоб мати можливість використовувати стандартні потоки необхідно підключити заголовочний файл iostream або iostream.h. Загалом різниця між стандартними заголовочними файлами з розширенням \*.h і без нього полягає в тому, що файли з розширенням \*.h відносяться до мови С, а без розширення – до С++. Таким чином програмуючи на мові С++ безпечніше використовувати заголовочні файли без розширення \*.h, які орієнтовані на мову С++. Проте в цьому випадку може бути необхідним підключати додатково простори імен. При використанні стандартних бібліотек вводувиводу таким простором імен є std. Об‘єкт стандартного потоку вводу cin класу istream, зв‘язаний із стандартним пристроєм вводу, за звичай клавіатурою. Об‘єкт стандартного потоку виводу cout класу ostream, зв‘язаний із стандартним пристроєм виводу, за звичай монітором. Об‘єкт cerr класу ostream, зв‘язаний із стандартним пристроєм виводу повідомлень про помилки. Потоки даних, що виводяться, для об‘єкту cerr являються небуферизованими. Тобто кожна операція помістити в cerr приводить до миттєвої появи повідомлень про помилки. Об‘єкт clog класу ostream, зв‘язаний із стандартним пристроєм виводу повідомлень про помилки. Потоки даних, що виводяться, для об‘єкту clog являються буферизованими. Тобто кожна операція помістити в clog може привести до того, що вивід буде зберігатися в буфері до тих пір, поки буфер повністю не заповниться або ж поки вмістиме буферу не буде виведене примусово. Вивід в потік виконується за допомогою операції «помістити в потік», а саме перевантаженої операції <<. Дана операція перевантажена для виводу елементів даних стандартних типів, для виводу рядків та значень вказівників. Операція << повертає посилання на об'єкт типу ostream, для якого вона викликана. Це дозволяє будувати ланцюжок викликів операції «помістити в потік», що виконуються зліва направо.

Ввід потоку виконується за допомогою операції «взяти із потоку», а саме перевантаженої операції >>. Дана операція зазвичай ігнорує у вхідному потоці символи розділювачі та пробільні символи. Операція «взяти із потоку» повертає нульове значення (false), якщо зустрічає в потоці признак кінця файлу або виникає помилка при спробі читання із потоку.

Засоби форматування потоку

Система вводу-виводу дозволяє виконувати форматування даних та змінювати визначені параметри вводу інформації. Дані операції реалізовані за допомогою функцій форматування, прапорців та маніпуляторів. Функції форматування та їх призначення приведені у табл. 2.2.

Таблиця 2.2. Функції форматування

|  |  |
| --- | --- |
| **Функція-член** | **Призначення** |
| width(int wide) | Дозволяє задати мінімальну ширину поля для виведення значення. При вводі задає максимальне число символів, що читаються. Якщо значення, що виводиться, має менше символів, ніж задана ширина поля, то воно доповнюється символами-  заповнювачами до заданої ширини (за замовчуванням - пробілами). Якщо ж значення, що виводиться має більше символів, ніж ширина відведеного йому поля, то поле буде розширене. |
| precision(int prec) | Дозволяє прочитати або встановити точність (число цифр після десяткової крапки), з якою виводяться числа з рухомою крапкою. По замовчуванню числа з рухомою крапкою виводяться з точністю, рівною шести цифрам. |
| fill(char ch) | Дозволяє прочитати або встановити символ-заповнювач. |

З кожним потоком зв'язаний набір прапорців, що керують форматуванням потоку. Вони являють собою бітові маски. Прапорці форматування і їхнє призначення приведені в табл. 2.3. Встановити значення одного або кількох прапорців можна за допомогою функції-члену setf(long mask).

Таблиця 2.3. Прапорці форматування і їхнє призначення

|  |  |
| --- | --- |
| **Прапорець** | **Призначення** |
| dec | Встановлюється десяткова система числення |
| hex | Встановлюється шістнадцяткова система числення |
| oct | Встановлюється вісімкова система числення |
| scientific | Числа з рухомою крапкою, виводяться в науковому записі (тобто n.хххЕуу) |
| showbase | Виводиться основа системи числення у виді префікса до цілого числового значення (наприклад, шістнадцяткове число 1FE виводиться як 0x1FE) |
| showpos | При виводі позитивних числових значень виводиться знак плюс |
| uppercase | Замінюються визначені символи нижнього регістра на символи верхнього регістра (символ "е" при виведенні чисел в науковому записі – на "Е" і символ "х" при виведенні чисел в шістнадцятковій системі – на "X") |
| left | Дані при виведенні вирівнюються по лівому краю поля виводу |
| right | Дані при виведенні вирівнюються по правому краю поля виводу |
| internal | Додаються символи-заповнювачі між усіма цифрами і знаками числа для заповнення поля виводу |
| skipws | Ведучі символи-заповнювачі (знаки пробілу, табуляції і переходу на новий рядок) відкидаються |

Список маніпуляторів та їхнє призначення приведені в табл. 2.4. Маніпулятори вводу-виводу являють собою вид функцій-членів класу іos, що, на відміну від звичайних функцій-членів, можуть розташовуватися усередині операцій вводу-виводу. За винятком функції-члену setw(int n), усі зміни в потоці, внесені маніпулятором, зберігаються до наступної установки. Для доступу до маніпуляторів з параметрами необхідно включити в програму стандартний заголовний файл іomanіp.

Таблиця 2.4. Маніпулятори вводу-виводу

|  |  |
| --- | --- |
| **Маніпулятор** | **Призначення** |
| endl | Виводить символ нового рядка та очищує потік |
| flush | Видає вмістиме буфера потоку у пристрій |
| dec | Встановлює десяткову систему числення |
| hex | Встановлює шістнадцяткову систему числення |
| oct | Встановлює вісімкову систему числення |
| setbase (int base) | Задає основу системи числення для цілих чисел (8, 10, 16) |
| setfill (int c) | Встановлює символ-заповнювач |
| setprecision  (int n) | Встановлює точність чисел з рухомою крапкою |
| setw(int n) | Встановлює мінімальну ширину поля виводу |
| setiosflags (iosbase::long mask) | Встановлює ios-прапорці згідно з mask |

Часто застосовувані функції

Крім вже описаних функцій, бібліотека вводу-виводу C++ містить широкий набір інших функцій. Тут ми приведемо лише деякі часто вживані з них. Для читання символу з потоку можна використовувати функцію-член get() потоку іstream. Функція get() повертає код прочитаного символу або -1, якщо зустрівся кінець файлу вводу (ctrl/z). Функція get(char\* str, іnt len, char delіm) може також використовуватися для читання рядка символів. У цьому випадку використовується її варіант, у якому ця функція читає з вхідного потоку символи в буфер str, поки не зустрінеться символ-обмежувач delіm (за замовчуванням – \n) або не буде прочитано (len-1) символів чи ознаку кінця файлу. Сам символ-обмежувач не читається з вхідного потоку. Для вставки символу в потік виведення використовується функція-член put(char ch). Через те, що функція get() не читає з вхідного потоку символобмежувач, вона використовується рідко. Набагато частіше використовується функція getlіne(char\* str, іnt len, char delіm), що читає з вхідного потоку символ-обмежувач, але не поміщає його в буфер.

Функція gcount() повертає число символів, прочитаних з потоку останньою операцією неформатуючого вводу (тобто функцією get(), getlіne() або read()).

Для того, щоб пропустити при введенні кілька символів, використовується функція іgnore(іnt n = l, іnt delіm = EOF). Ця функція ігнорує n символів у вхідному потоці. Пропуск символів припиняється, якщо вона зустрічає символ-обмежувач, яким по замовчуванню є символом кінця файлу. Символ-обмежувач читається з вхідного потоку. Функція peek() дозволяє "заглянути" у вхідний потік і довідатися наступний символ, що вводиться. При цьому сам символ з потоку не читається. За допомогою функції putback(char ch) можна повернути символ ch у потік вводу.

Файловий ввід-вивід

Робота з файлами в мові С++ як і у мові С передбачає 3 етапи: відкривання файлу (файлового потоку), обмін даними з файловим потоком, закривання файлового потоку. Для виконання операцій з файлами в мові С++ передбачено три класи: іfstream, ofstream і fstream. Ці класи є похідними від класів іstream, ostream і іostream. Всі функціональні можливості (перевантажені операції << та >> для вбудованих типів, функції і прапорці форматування, маніпулятори й ін.), що застосовуються до стандартного вводу та виводу, можуть застосовуватися і до файлів. Існує деяка відмінність між використанням стандартних та файлових потоків. Стандартні потоки можуть використовуватися відразу після запуску програми, тоді як файловий потік спочатку слід зв'язати з файлом. Для реалізації файлового вводу-виводу потрібно підключити заголовочний файл fstream, що знаходиться в просторі імен std.

Режими відкриття файлів та їхнє призначення наведені у табл. 2.5.

Таблиця 2.5. Режими відкриття файлів

|  |  |
| --- | --- |
| **Режим відкриття** | **Призначення** |
| іos::іn | Відкрити файл для читання |
| іos::out | Відкрити файл для запису |
| іos::ate | Відкрити файл для додавання в кінець |
| іos::app | Відкрити файл для додавання в кінець |
| іos::trunc | Усікти файл, тобто видалити його вміст |
| іos::bіnary | Відкрити файл у двійковому режимі |

Режими відкриття файлу являють собою бітові маски, тому можна задавати два або більш режими, поєднуючи їх побітовою операцією АБО. Слід звернути увагу, що по замовчуванню режим відкриття файлу відповідає типові файлового потоку. У потоці вводу або виводу прапорець режиму завжди встановлений неявно. Між режимами відкриття файлу іos::ate та іos::app існує певна відмінність. Якщо файл відкривається в режимі додавання (іos::app), весь вивід у файл буде здійснюватися в позицію, що починається з поточного кінця файлу, безвідносно до операцій позиціонування у файлі. У режимі відкриття іos::ate (від англійського "at end") можна змінити позицію виводу у файл і здійснювати запис, починаючи з неї. Файли, які відкриваються для виводу, створюються, якщо вони ще не існують. Якщо при відкритті файлу не зазначений режим іos::bіnary, файл відкривається в текстовому режимі. Якщо відкриття файлу завершилося невдачею, об'єкт, що відповідає потокові, буде повертати нуль. Перевірити успішність відкриття файлу можна також за допомогою функції-члена іs\_open(). Дана функція повертає 1, якщо потік вдалося зв'язати з відкритим файлом. Для перевірки, чи досягнутий кінець файлу, можна використовувати функцію eof(). Завершивши операції вводу-виводу, необхідно закрити файл, викликавши функцію-член close().

Меню

Більшість прикладних програм передбачає інтерактивну взаємодію користувача з комп‘ютером. Для її реалізації з метою керування процесом роботи консольної програми часто використовується меню. Воно передбачає вивід на екран варіантів функціонування програми (включно з варіантом, що передбачає вихід з програми) з подальшим вибором одного з них користувачем. Один з алгоритмів реалізації меню має наступний вигляд:

while (true)

{

вивід <- вивести варіанти роботи

ввід -> варіант роботи

if (варіант роботи 1)

{

Функціонування програми згідно варіанту роботи 1 } …

if (варіант роботи N)

{

Функціонування програми згідно варіанту роботи N

}

if (вихід)

{

break; //Переривання вічного циклу

}

}

**ІНДИВІДУАЛЬНЕ ЗАВДАННЯ**

(Варіант 15)

З клавіатури вводиться найменування жорсткого диску, його об‘єм і відсоток зайнятого простору, а у файл записується найменування жорсткого диску його об‘єм і кількість байт вільного простору

**Код виконання програми:**

#include <iostream>

#include <fstream>

#include<iomanip>

using namespace std;

void write(string Disk\_\_Name, int Disk\_\_Size, int Disk\_\_Percent, int size)

{

ofstream file("laba.txt", ios\_base::app);

file << endl << Disk\_\_Name << " " << Disk\_\_Size << " " << Disk\_\_Percent << " " << size;

file.close();

}

void read()

{

string Buf;

ifstream file("laba.txt");

cout << setw(10) << "Disk name:" << setw(15) << "Disk size" << setw(20) << "Disk percent" << setw(20) << "Disk occupation" << endl;

while (!file.eof())

{

file >> Buf;

cout << setw(10) << Buf;

file >> Buf;

cout << setw(15) << Buf;

file >> Buf;

cout << setw(20) << Buf;

file >> Buf;

cout << setw(20) << Buf << endl;

}

cout << endl;

file.close();

}

int main()

{

string Disk\_\_Name;

int Disk\_\_Size, choose, size;

double Disk\_\_Percent;

do

{

cout << "Add information(0)\nRead information(1)\nExit(3)\n";

cin >> choose;

switch (choose)

{

case 0:

system("cls");

cout << "Enter disk name: ";

cin >> Disk\_\_Name;

cout << "Enter disk size MB: ";

cin >> Disk\_\_Size;

cout << "Enter disk percent occupation: ";

cin >> Disk\_\_Percent;

size = Disk\_\_Size \* 8192 \* ((100 - Disk\_\_Percent) / 100);

write(Disk\_\_Name, Disk\_\_Size, Disk\_\_Percent, size);

cout << endl;

break;

case 1:

system("cls");

read();

break;

case 3:

system("cls");

return 0;

default:

system("cls");

cout << "Enter corect!!!" << endl;

continue;

}

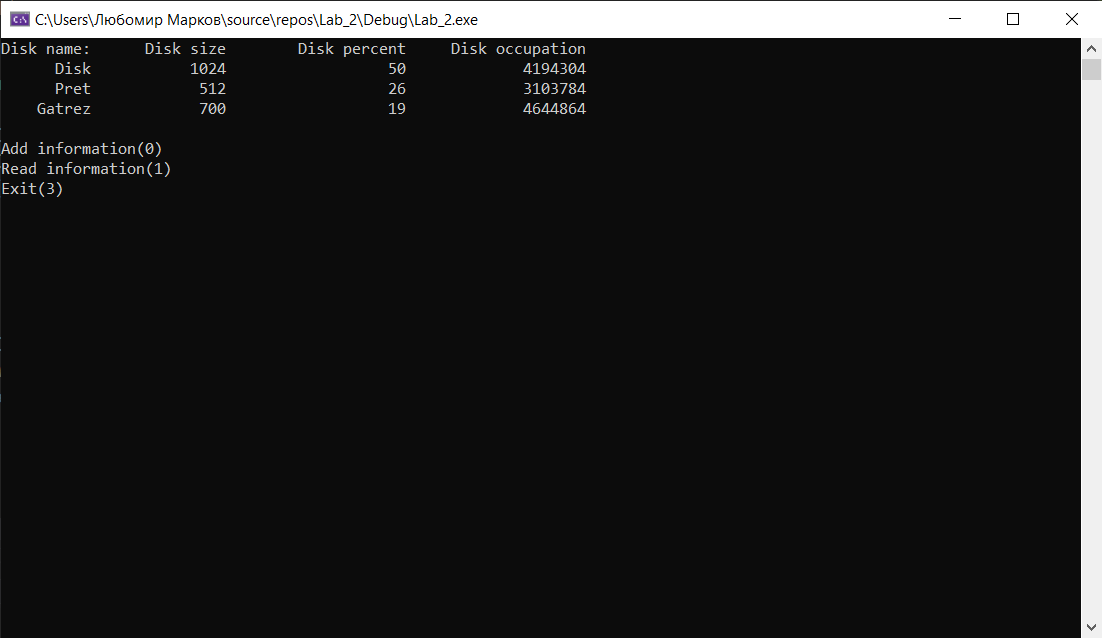
} while (true);

system("pause");

return 0;

}

**Виконання програми:**

****

**Висновок**: На цій лабораторній роботі я ознайомився з потоковим вводом-виводом.

Контрольні питання:

1. Що таке простір імен?

Простори імен служать для об‘єднання класів, що написані різними

програмістами або мають схоже призначення в групи. Для оголошення простору імен слід вжити ключове слово namespace

після якого слід вказати назву простору імен.

1. Потоки вводу-виводу в С++?

Об‘єкт стандартного потоку вводу cin класу istream, зв‘язаний із стандартним пристроєм вводу, за звичай клавіатурою. Об‘єкт стандартного потоку виводу cout класу ostream, зв‘язаний із стандартним пристроєм виводу, за звичай монітором.

1. Що таке маніпулятор?

Маніпулятори вводу-виводу являють собою вид функцій-членів класу іos, що, на відміну від звичайних функцій-членів, можуть розташовуватися усередині операцій вводу-виводу.