**Лекція 6**

**Форматне введення виведення**

**План**

1. Порівняння старої й нової систем вводу-виводу
2. Потоки
3. Форматне уведення-виведення
4. Встановлення і скидання опції формату
5. Перевантажена форма функції setf()
6. Перевірка опцій форматування
7. Встановлення всіх опцій
8. Застосування функцій width(), precision() і fill()
9. Застосування маніпуляторів формату

Мова C++ підтримує дві повноцінні системи вводу-виводу: одна з них успадкована від мови С, а інша є об’єктно-орієнтованою і визначена в мові C++ (із цього моменту ми будемо називати її просто *системою вводу-виводу мови* C++). Системи вводу-виводу мови С и C++ є абсолютно сумісними. Різні компоненти кожної із систем, наприклад, засоби для роботи з консоллю або файлами, просто по-різному реалізують той самий механізм.

Оскільки система вводу-виводу, успадкована від мови С, дуже різноманітна й ефективна, виникає питання: "Навіщо знадобилося створювати ще одну систему вводу-виводу?". Відповідь проста: система вводу-виводу мови С нічого не знає про об'єкти.Отже, щоб система вводу-виводу повністю відповідала принципам об’єктно-орієнтованого програмування, необхідні засоби для роботи з об'єктами, певними користувачем.Крім цього, система вводу-виводу мови C++ володіє ще рядом переваг, навіть якщо програма не дуже широко використовує об'єкти.Чесно говорячи, всі нові програми повинні використовувати тільки систему вводу-виводу мови C++.Стара система, успадкована від мови С, застосовується тільки для підтримки сумісності програм.

**Порівняння старої й нової систем вводу-виводу**

У цей час використовуються дві бібліотеки об’єктно-орієнтованого виводу: стара, заснована на вихідних специфікаціях мови C++, і сучасним, певним стандартом мови C++. Стара бібліотека вводу-виводу підтримується за допомогою заголовка <iostream.h>. Нова система вводу-виводу забезпечується заголовком <iostream>. У багатьох відносинах ці системи нічим не відрізняються, оскільки сучасна бібліотека просто є модифікованою й поліпшеною версією старої бібліотеки.Основні розходження між ними сховані від спостерігачів, оскільки ставляться до механізмів реалізації, а не до способів застосування засобів вводу-виводу.

З погляду програміста, між старою й новою системами вводу-виводу мови C++ є два розходження. По-перше, нова бібліотека має додаткові властивості й визначає кілька нових типів даних. Отже, стара бібліотека є підмножиною нової. Практично всі програми, написані з використанням старої бібліотеки вводу-виводу, без проблем компілюються новими компіляторами. По-друге, стара бібліотека вводу-виводу перебувала в глобальному просторі імен. (Нагадаємо, що простір імен **std** використовується всіма стандартними бібліотеками мови C++.) Оскільки ця бібліотека морально застаріла, у книзі використається тільки нова система вводу-виводу, хоча всі програми можна скомпілювати й зі старою бібліотекою.

**Потоки**

Система вводу-виводу мови C++, який її аналогу мові С, оперує потоками. П*отік* (stream) — це логічний пристрій, що одержує або передає інформацію. Потік пов'язаний з фізичним пристроєм вводу-виводу. Всі потоки функціонують однаково, хоча фізичні пристрої, з якими вони зв'язані, можуть бути різними.Оскільки всі потоки однакові, одна функція вводу-виводу може працювати з різними типами фізичних пристроїв.Наприклад, за допомогою однієї функції можна виводити дані як на принтер, так і на екран.

**Класи потоків у мові C++**

Як відомо, для системи вводу-виводу необхідний заголовок <iostream>. У цьому заголовку визначена досить складна ієрархія класів, що підтримують операції вводу-виводу. Спочатку визначаються шаблонові класи вводу-виводу.Шаблоновий клас являє собою схему, у якій не уточнюється тип даних, якими вона оперує. Після визначення шаблонових класів можна створювати їхньої конкретизації. Стандарт мови C++ створює дві спеціалізації шаблонових класів вводу-виводу: одну для восьмибітових символів, а іншу - для розширених.

Система вводу-виводу мови C++ побудована на основі двох родинних, але різних ієрархій шаблонових класів. В основі першої ієрархії лежить клас Bаsic\_streambuf, призначений для низькорівневого вводу-виводу. У звичайних додатках частіше застосовується ієрархія класів, побудована на основі класу basic\_ios, що забезпечує високорівневі операції вводу-виводу, перевірку помилок й аналіз інформації про статус потоків

**Вбудовані потоки в мові C++**

На початку виконання програми мовою C++ автоматично відкриваються чотири потоки.

|  |  |  |
| --- | --- | --- |
| ***Потоки*** | ***Значення*** | ***Пристрій за замовчуванням*** |
| **сin** | Стандартне введення | Клавіатура |
| cout | Стандартний вивід | Екран |
| cerr | Стандартне виведення помилок | Екран |
| clog | Варіант потоку cerr, що буферизує | Екран |

Потоки **cin, cout** і **cerr** відповідають потокам **stdin, stdout** й **stderr.**

За замовчуванням стандартні потоки використовуються для взаємодії з консоллю.Однак в операційних системах, що підтримують перенаправлення потоків вводу-виводу (таких як DOS, Unix, OS/2 й Windows), стандартні потоки можна зв'язати з іншими пристроями або файлами.

Крім того, мова C++ визначає чотири додаткових потоки: **win, wout, werr** й **wlog.** Це версії потоків для вводу-виводу розширених символів. Для подання розширених символів використовується тип **wchar\_t** й 16-бітові значення. Як правило, розширені символи застосовуються для підтримки деяких природних мов.

**Форматне уведення-виведення**

Мова C++ дозволяє виконувати операції форматованого вводу-виводу. Наприклад, можна задати ширину поля, указати основу числення або визначити кількість цифр після десяткової крапки. Для форматування даних можна застосовувати два схожих, але різних способу. По-перше, можна прямо звернутися до членів класу **ios.** Зокрема,можна самостійно задавати різні опції форматування, визначені у середині класу **ios,** або викликати різноманітні функції-члени. По-друге, у виразах вводу-виводу можна використати спеціальні функції, які називаються *маніпуляторами* (manipulators).

Розглянемо спочатку засоби форматованого вводу-виводу за допомогою опцій і функцій — членів класу **ios.**

**Форматування за допомогою членів класу ios**

Кожен потік пов'язаний з набором опцій формату, керуючих поданням інформації. Клас **ios** повідомляє бітову маску за назвою **fmtflags,** у якій визначаються наступні значення. (З технічної точки зору ці значення визначені в класі **ios\_base,** що є базовим стосовно класу **ios.)**

adjustfield basefield boolalpha dec

fixed floatfield hex internal

left oct right scientific

showbase showpoint showpos skipws

unitbuf uppercase

Ці значення використовуються для встановлення й скидання опцій форматування. При роботі зі старими компіляторами неможливо визначити перерахування **fmtflags. У** такому випадку опції формату кодуються за допомогою значень, що мають тип **long.**

Якщо встановлено опцію **skipws**, при уведенні даних з потоку роздільники (пробіли, знаки табуляції й символи переходу на новий рядок) ігноруються. Якщо цей опцію скинутий, роздільники враховуються.

Якщо встановлено опцію **left**, рядки виводу вирівнюються по лівому краю. Якщо встановлено опцію **right** - по правому краю. Якщо встановлено опцію **internal**, між знаком числа і його першою цифрою пробіли уставляються так, щоб число заповнило собою все поле виводу.Якщо жодна із цих опцій не встановлена, за замовчуванням виконується вирівнювання по правому краї.

За замовчуванням числові значення виводяться в десятковому виді.Однак основу системи числення можна змінити. Для виводу вісімкових чисел призначена опція **oct**. Встановлення опції **hex** дозволяє виводити числа в шістнадцятковому форматі.Вивід чисел у десятковому форматі забезпечується опцією **dec**.

Встановлення опції **showbase** дозволяє вивести на екран основу системи числення. Наприклад, при виводі шістнадцяткових чисел значення 1F буде відображене як Ox1F.

При виводі чисел у науковому форматі буква “е” за замовчуванням виводиться як рядкова. Крім того, буква “х” у вісімковому поданні чисел також вважається рядковою.Якщо необхідно вивести ці букви як прописні, варто встановити опцію **uppercase**.

Встановлення опції **showpos** дозволяє вивести знак перед позитивними числами.

Встановлення опції **showpoint** дозволяє виводити десяткову крапку й незначущі нулі при відображенні десяткових чисел.

Якщо встановлено опцію **scintiefic**, число виводиться в науковому форматі. Якщо встановлено опцію **fixed**, десяткове число виводиться у звичайному виді.Якщо жодна із цих опцій не встановлена, компілятор сам вибирає підходяще подання чисел.

Якщо встановлено опцію **unitbuf**, то після кожної операції вставки буфер очищається.

Опцію **boolalpha** дозволяє вводити й виводити булеві значення true й false.

Оскільки числа звичайно виводяться в десятковому, вісімковому й шістнадцятковому форматі, поля dec, oct й hex називають загальним ім'ям **basefield**. Аналогічно поля left, right й internal називають **adjustfield**. Крім того, поля scientific й fixed поєднують загальним ім'ям **floatfield**.

**Встановленняопції формату**

Для встановлення опції використовується функція **setf ()**. Ця функція є членом класу **ios**. Вона має такий вигляд.

fmt flagssetf(fmt flags *flags);*

Дана функція повертає поточний стан опцій формату, відзначених параметром *flags* і встановлює їх. Наприклад, щоб установити опцію **showpos**, можна застосувати наступний оператор.

*stream.*setf (ios : : showpos) ;

Тут ім'я ***stream*** означає потік, на який ви хочете вплинути. Зверніть увагу на префікс ios:: перед опцією **showpos**. Він необхідний, оскільки опцію **showpos** є перелічувальною константою, визначеною в класі **ios**.

Наступна програма виводить на екран число 100, установлюючи опції **showpos** і **showpoint.**

#include<iostream>

using namespace std;

int main()

{

cout.setf(ios::showpoint);

cout.setf(ios::showpos);

cout « 100.0; // Виводимо число +100.0

return 0;

}

Варто пам'ятати, що функція **setf ()** є членом класу **ios** і впливає на потоки, створені цим класом. Отже, будь-який виклик функції **setf ()** пов'язаний з конкретним потоком.Сама по собі функція **setf ()** ніколи не викликається. Інакше кажучи, у мові C++ немає концепції глобального статусу формату. Кожен потік підтримує свій власний статус.

Хоча попередня програма є синтаксично правильною, її можна переписати й зробити більше ефективною.Замість декількох викликів функції setf() можна застосувати до її аргументів логічну операцію "АБО".Наприклад, що попередні виклики можна замінити одним.

cout.setf (ios : :showpoint | ios :: showpos);

**Скидання опцій формату**

Антиподом функції **setf ()** є функція **unsetf ().**Ця функція - член класу **ios**- використовується для скидання одного або декількох параметрів формату й має такий вигляд.

void unsetf (fmtflags *flags);*

Дана функція скидає опції, задані своїм параметром.(Всі інші опції зберігають свій колишній стан.)

Функція **unsetf ()** ілюструється наступною програмою. Спочатку вона встановлює опції uppercase й scientific, потім виводить число 100.12 у науковому форматі.У цьому випадку науковий формат числа містить прописну букву "Е".Після цього програма скидає опцію uppercase і знову виводить число 100.12 у науковому форматі, цього разу використовуючи малу літеру "е".

#include<iostream>

using namespace std;

int main()

{

cout.setf (ios::uppercase | ios::scientific);

cout « 100.12; // Виводимо число 1.0012Е+02

cout.unsetf (ios::uppercase); // Скидаємо опцію uppercase

cout « " \n" « 100.12; // Виводимо число 1.0012е+02

return 0;

}

**Перевантажена форма функції setf()**

Функція **setf**() має перевантажену форму.

fmt flagssetf (fmtflags *опції 1,* fmtflags *опції 2)* ;

У цій версії змінюються тільки опції, задані параметром *опції2.* Спочатку вони скидаються, а потім встановлюються у відповідності зі статусами опцій, заданих параметром *опції 1.* Зверніть увагу на те, що навіть якщо параметри *опції 1* і *опції 2* ставляться до різних опцій, змінюються тільки опції, задані параметром *опції 2.* Функція повертає попередній стан опцій.

#include<iostream>

using namespace std;

int main( )

{

cout.setf(ios::showpoint | ios::showpos, ios::showpoint);

cout « 100.0; // Виводить на екран число 100.0, а не +100.0

return 0;

}

Програма встановлює опцію **showpoint,** а не **showpos,** оскільки опцію **showpos** незазначений у другому параметрі.

#include<iostream>

using namespace std;

int main()

{

cout.setf(ios::hex, ios::basefield);

cout « 100; // Виводить на екран число 64

return 0;

}

У цій програмі спочатку скидаються опції **basefield** (тобто опції **dec, oct** й **hex),** а потім установлюється опцію **hex.**

Стан опцій, заданих параметром *опції 1,* впливає тільки на опцій, заданих параметром *опції 2.* Наприклад, у наступній програмі перша спроба встановити опцію *showpos* виявляється невдалою.

// Ця програма не працює,

#include<iostream>

using namespace std;

int main()

{

cout.setf(ios::showpos, ios::hex); // Помилка, опцію

// showpos не встановлений.

cout « 100 « '\n'; // Виводить на екран число 100, а не +100

cout.setf(ios::showpos, ios::showpos); // Правильно

cout « 100; // Тепер на екран виводиться число +100.

return 0;

}

В більшості випадків для скидання опцій використовується функція unsetf(), а для встановлення опції викликається функція setf() з одним параметром. У більше складних випадках, наприклад, при виводі основи системи числення варто викликати функцію setf (fmtflags, fmtflags). Крім того, іноді програмісти створюють шаблон всіх опцій і в формату, що описує подання чисел, а потім намагаються змінити одну або дві опції. У цьом увипадку шаблон задають за допомогою параметра *опції 1,* а змінювані опції перераховують у параметрі *опції 2.*

**Перевірка опцій форматування**

Іноді необхідно точно визначити поточний стан формату, не змінюючи його опції. Для цього в класі ios використається функція-член flags(). Яка повертає поточний стан кожноїопції формату.Її прототип має такий вигляд.

fmt flagsflags();

У наступній програмі ця функція застосовується для встановленняопцій форматування, пов'язаних з потоком cout. Зверніть особливу увагу на функцію showflags (). Вона може виявитися корисною.

#include<iostream>

using namespace std;

void showflags ;

intmain()

{

// Показати стан опцій формату,

// передбачити за замовчуванням.

showflags();

cout.setf(ios::right | ios::showpoint | ios::fixed);

showflags();

return 0;

}

// Функція виводить на екран стан опцій формату.

void showflags()

{

ios::fmtflags f;

long i;

f = (long) cout.flags(); // Одержати стан опцій.

// Перевірити коженої опції

for(i=0x4000; і; і = i>>l)

if(і & f) cout « "1 "

else cout « "0 ";

cout « " \n";

}

Нижче показані результати роботи цієї програми.(Ці установки залежать від компілятора.)

1000001000000001 010001010010001

**Встановлення всіх опцій**

Функція **flags ()** має другу форму, що дозволяє встановлювати всі опції формату, пов'язані з потоком.Прототип цієї версії функції **flags ()** показаний нижче.

fmtflagsflags(fmtflags*f) ;*

У цій версії параметр *f* задає бітовий шаблон опцій формату, пов'язаних з конкретним потоком.Таким чином, змінюються відразу всі опції форматування.Функція повертає попередній стан опцій.

Ця версія функції **flags()**ілюструється наступною програмою.Спочатку вона створює бітову маску, у якій установлені опції **showpos**, **showbase**, **oct** й **right**. Всі інші опції скинуті. Потім для установки опцій форматування, пов'язаних з потоком **cout**, використовується функція **flags()**. Функція **show-flags()** виводить на екран стан опцій форматування.(Це та ж функція, що була визначена в попередньому прикладі.)

#include<iostream>

using namespace std;

void showflags();

int main()

{

// Виводимо верб екран стан опцій формату,

// передбачене за замовчуванням.

showflags();

// Опції showpos, showbase, oct й right установлені,

// інші скинуті.

long f = ios::showpos | ios::showbase | ios::oct | ios::right;

cout.flags(f); // Задаємо стан всіх опцій.

showflags();

return 0;

}

**Застосування функцій width(), precision() і fill()**

У класі **ios** передбачені три функції-члени, що дозволяють змінювати ширину поля виводу, точність і символ-заповнювач. Вони називаються **width()**, **precision()** і **fill ()** відповідно.Розглянемо кожну з них окремо.

За замовчуванням кількість позицій, які займає число при виводі, дорівнює кількості символів, з яких воно складається. Однак існує можливість змінити мінімальну ширину поля виводу, використовуючи функцію **width()**. Її прототип показаний нижче.

streamsize width(streamsize w);

Функція повертає попередню ширину поля, а її нове значення задається параметром *w.* У деяких реалізаціях ширина поля повинна задаватися при кожному виводі. Якщо цього не зробити, використовується її значення, передбачене за замовчуванням. Тип **streamsize** являє собою один з варіантів типу **int,** причому конкретний вибір залежить від компілятора.

Якщо число не цілком заповнює поле, воно доповнюється символом-заповнювачем (за замовчуванням - пробілом), щоб довжина числа збігалася із шириною поля.Якщо число не міститься в поле виводу, воно однаково виводиться повністю.Усікання чисел не виробляється.

При виводі дійсних чисел можна задавати кількість цифр після десяткової крапки (точність числа), використовуючи функцію **precision**(). Її прототип представлений нижче.

| streamsize precision(streamsize *р) ;*

Функція повертає старе значення, а нова кількість цифр після десяткової крапки задається параметром *р.* За замовчуванням після десяткової крапки виводиться 6 цифр. У деяких реалізаціях це значення варто задавати перед кожним виводом, інакше буде використовуватися точність, передбачена за замовчуванням.

Крімтого, якщо поле виводу незаповнено, воно автоматично доповнюється пробілами. Символ-заповнювач можна змінити, використавши функцію **fill().** Її прототип має такий вигляд.

char fill (char*ch) ;*

Функція **fill**() повертає старий символ-заповнювач, а новим заповнювачем стає символ *ch.*

Розглянемо програму, що ілюструє застосування цих функцій.

#include<iostream>

using namespace std;

int main ()

{

cout.precision(4) ;

cout.width(l0);

cout « 10.12345 « "\n” // потрібно cout<<fixed<<10.12345 <<endl;

cout.fill(' \*' ) ;

cout.width(10);

cout « 10.12345 « "\n"; // Виводить на екран \*\*\*\*\*10.12

// Ширина поля виводу поширюється й на рядки

cout.width(l);

cout « "Hi!" « "\n"; // Виводить \*\*\*\*\*\*\*ні!

cout.width(10);

cout.setf(ios::left); // Вирівнює по лівому краї

cout « 10.12345; // Виводить на екран 10.12\*\*\*\*\*

return 0;

}

Результат роботи цієї програми виглядає так.

10.12

\*\*\*\*\*10.12

\*\*\*\*\*\*\*ні!

10.12\*\*\*\*\*

Функції **width(), precision()** і**fill()** мають перевантажені версії, що одержують, але не відповідні параметри, що змінюють, потоку.

char fill () ;

streamsize width;

streamsize precision();

**Застосування маніпуляторів формату**

Інший спосіб зміни формату заснований на застосуванні спеціальних функцій, називаних *маніпуляторами* (manipulators). Їх можна включати в оператори вводу-виводу.Стандартні маніпулятори перераховані в табл. 20.1.

**Таблиця 20.1. Маніпулятори**

|  |  |  |
| --- | --- | --- |
| *Маніпулятори* | *Призначення* | *Ввід-вивід* |
| boolalpha  dec  endl  ends  fixed  flush  hex  left  noboolalpha  noshowbase  noshowpoint  noshowpos  noskipws  nounitbuf  nouppercase  oct  resetiosflags (fmtflags*f)*  right  scientific  setbase(іnt*base)*  setfill(int*ch)*  setiosflags (fmtflags f)  setprecision(int р)  setw(int*w)*  showbase  showpoint  showpos  skipws  unitbuf  uppercase  ws | Установлює опцію boolalpha  Установлює опцію dec  Виводить символ переходу на новий рядок й очищає буфер  Виводить нульовий байт  Установлює опцію fixed  Очищає буфер  Установлює опцію hex  Установлює опцію left Скидає опцію boolalpha  Скидає опцію showbase  Скидає опцію showpointСкидає опцію showpos  Скидає опцію skipws  Скидає опцію unitbuf  Скидає опцію uppercase  Установлює опцію oct  Скидає опції, зазначені параметром *f*  Установлює опцію right  Установлює опцію scientific  Задає основи системи числення, зазначені параметром *base*  Задає символ-заповнювачїв  Установлює опції, зазначені параметром *f*  Задає кількість цифр після десяткової крапки  Задає ширину поля, зазначену параметром *w*  Установлює опцію showbase  Установлює опцію showpoint  Установлює опцію showpos  Установлює опцію skipws  Установлює опцію unitbuf  Установлює опцію uppercase  Ігнорує провідні роздільники | Ввід-вивід  Ввід-вивід  Вивід  Вивід  Вивід  Ввід-вивід  Вивід  Ввід-вивід  Вивід  Вивід  Вивід  Вивід  Вивід  Вихід  Ввід-вивід  Ввід-вивід  Вихід  Вихід  Ввід-вивід  Вихід  Ввід-вивід  Вихід  Вихід  Вихід  Вихід  Вихід  Вхід  Вихід  Вихід  Вхід |

Для доступу до маніпуляторів, що одержують параметри (наприклад, до функції setw()), необхідно включити в програму заголовок <iomanip>. Розглянемо приклад, що ілюструє застосування маніпуляторів.

#include<iostream>

#include<iomanip>

using namespace std;

int main() {

cout<<hex << 100 <<endl;

cout « setfill('?') << setw(10) « 2343.0;

return 0; }

Результат роботи програми наведений нижче.

164

??????2343

Зверніть увагу на те, як маніпулятори включаються в оператори виводу Крім того, якщо маніпулятор не має аргументів (наприклад, endl), дужки після нього не ставляться, оскільки його ім'я є адресою, переданою перевантаженому операторові "<<".

Для порівняння приведемо функціонально еквівалентну версію попередньої програми, у якій використаються функції - члени класу ios.

#include<iostream>

#include<iomanip>

using namespace std;

int main() {

cout.setf(ios::hex, ios::basefield);

cout<< 100 « "\n"; // Число 100 у шістнадцятковому форматі

cout.fill('?'); cout.width(10); cout « 2343.0;

return 0; }

Як бачимо, основна перевага маніпуляторів над функціями - членами класу ios полягає в компактності коду.

Використовуючи маніпулятор **setiosflags()**, можна встановити відразу всі опції формату, пов'язані з потоком.Наприклад, у наступній програмі маніпулятор **setiosflags ()** застосовується для установки опціюів**showbase** й **showpos**.

#include<iostream>

#include<iomanip>

using namespace std;

intmain

{

cout<<setiosflags(ios::showpos);

cout<<setiosflags(ios::showbase);

cout<< 123 << " " <<hex<< 123;

return 0;

}

Маніпулятор **setiosflags**() і функція **setf**() виконують однакові операції. Маніпулятор **boolalpha** цікавіший. Він дозволяє вводити й виводити логічні значення, використовуючи ключові слова **true** й **false,** а не числа.

#include<iostream>

using namespace std;

int main()

{

bool b;

b = true;

cout<< b << " " <<boolalpha<< b <<endl;

cout<< "Введіть булеве значення: ";

сіn>>boolalpha>> b;

cout<< "Ви ввели наступне значення: " << b;

return 0;

}

Розглянемо приклад її роботи.

1 true

Введіть булеве значення: false

Ви ввели наступне значення: false

**Використана література**

1. Г. Шилдт, Полный справочник по С++, 4-е видання, в-во «Вильямс», 2006
2. Х.М.Дейтел, Какпрограмировать на С++, 4-е видання, в-во «Бином-Пресс» 2009.
3. Р. Лафоре, Объектно-ориентированное программирование в С++, в-во «Питер», 2004, с 924.