# Лекція №3

## Потокові маніпулятори

- **Маніпулятори**
- Маніпуляторами називаються функції, які можна включати в ланцюжок операцій приміщення та вилучення для форматування даних. Маніпулятори діляться на прості, які не потребують вказівки аргументів, і параметризовані. Користуватися маніпуляторами більш зручно, ніж методами встановлення прапорів форматування
- Нижче перераховані маніпулятори, які не потребують вказівки аргументів.

| | |
|---|---|
| dec | — устанавливает при вводе и выводе флаг десятичной системы счисления; |
| oct | — устанавливает при вводе и выводе флаг восьмеричной системы счисления; |
| hex | — устанавливает при вводе и выводе флаг шестнадцатеричной системы счисления; |
| ws | — устанавливает при вводе извлечение пробельных символов; |
| endl | — при выводе включает в поток символ новой строки и выгружает буфер; |
| ends | — при выводе включает в поток нулевой символ; |
| flush | — при выводе выгружает буфер. |

- Зміни системи числення діють до наступної явної зміни.
- Приклад:
- cout << 13<< hex<< '  '<< 13 << oct<< ' '<< 13 << endl;
- Якщо інші значення прапорів встановлені за замовчуванням, буде виведено:
- 13 d 15
- **Параметризовані маніпулятори**
- Нижче перераховані маніпулятори, що вимагають вказівки аргументу. Для їх використання потрібно підключити до програми заголовний файл <iomanip>.

| | |
|---|---|
| setbase(int n) | — задает основание системы счисления (п = 8, 16, 10 или 0). 0 является основанием по умолчанию (десятичное, кроме случаев, когда вводятся 8- или 16-ричные числа); |
| resetiosflags(long) | — сбрасывает флаги состояния потока, биты которых установлены в параметре; |
| setiosflags(long) | — устанавливает флаги состояния потока, биты которых в параметре равны 1; |

| | |
|---|---|
| setfill (int) | — устанавливает символ-заполнитель с кодом, равным значению параметра; |
| setprecision(int) | — устанавливает максимальное количество цифр в дробной части для вещественных чисел в форме с фиксированной точкой (флаг fixed) или общее количество значащих цифр для чисел в форме с мантиссой и порядком (флаг scientific); |
| setw(int) | — устанавливает максимальную ширину поля вывода. |

- Приклад використання параметризованих маніпуляторів. Для нього наведемо приклад, який робить аналогічний форматований вивід інформації про студентів, тільки формат виведення чисел встанови у вигляді мантиси з порядком
- #include "stdafx.h"
- #include <iostream>
- #include <iomanip>
- using namespace std;
- const int n=2;
- //Створимо структуру для зберігання інформації про студентів з двома полями ПІБ та середній бал успішності

```
struct Tstudent
{
char FIO[20];
 float bal;
}StudetnPotok[n]={"Petrov",3.5,"Sidorov",5.0/3};
int main (int argc, char * const argv[])
{
 for(int i=0;i<n;i++)
 {
```

- cout<<resetiosflags(ios::right);// те ж що і
- cout.unsetf(ios::right);
- cout<<setiosflags(ios::left);  // те ж що і
- cout.setf(ios::left);
- cout<<setfill('.');  //те ж що і cout.fill('.');
- cout<<setw(15);  // те ж що і  cout.width(15);
- cout<<setiosflags(ios::fixed); //те ж що і
- cout.setf(ios::fixed);
- cout<<setiosflags(ios::scientific);// cout.setf(ios::scientific);
- cout<<resetiosflags(ios::fixed);// cout.unsetf(ios::scientific);

```
cout<<setprecision(3); //теж що і
 cout.setf(ios::scientific);
      cout <<StudetnPotok[i].FIO;//виводимо прізвище
      cout<<setw(10);         //те ж що і
 cout.width(10)
      cout<<setiosflags(ios::right);// те ж що і
cout.setf(ios::right);
      cout<<setfill('_');       //cout.fill('_')
      cout<<StudetnPotok[i].bal<<'\n';
}
 cin.get();
 return 0;
}
```

- Цей фрагмент виведе на екран наступний результат:
- Petrov....._3.500e+00
- Sidorov....._1.667e+00
- Використовуючи маніпулятори методи можна об'єднувати. Наступний фрагмент коду робить той же самий висновок за винятком того що змінений формат виведення чисел
- #include <iostream>
- #include <iomanip>
- using namespace std;

```
const int n=2;
struct Tstudent
{
 char FIO[20];
 float bal;
}StudetnPotok[n]={"Petrov",3.5,"Sidorov",5.0/3};
int main (int argc, char * const argv[])
{
 for(int i=0;i<n;i++)
 {
cout<<resetiosflags(ios::right)<<setiosflags(ios::left)<<setfill('.')
            <<setw(15)<<StudetnPotok[i].FIO;
```

- cout<<setprecision(3)<<setw(10)<<setiosflags(ios::right|ios::fixed)
- <<setfill('_')<<StudetnPotok[i].bal<<'\n';
- }
- cin.get();
- return 0;
- }
- Цей фрагмент виведе на екран наступний результат:
- Petrov....._____3.500
- Sidorov....._____1.667

- Наведемо приклад, як за допомогою прапорів, методів і маніпуляторів потоку можна створити власну функцію для форматованого виведення спеціалізованих даних
- Нехай у нас є інформація про студентів, яку необхідно виводити користувачеві в зручному вигляді
- #include <iostream>
- #include <fstream>
- #include <iomanip>
- using namespace std;
- const int n=2;

- //Створимо структуру для зберігання інформації про студентів з двома полями ПІБ та середнім балом успішності
- struct Tstudent
- {
- char FIO[20];
- float bal;
- }Stud[n]={"Petrov",3.5,"Sidorov",5.0/3};
- void myOut(ostream &out,Tstudent *StudetnPotok,int n)
- {
- for(int i=0;i<n;i++)
- {

```
cout<<resetiosflags(ios::right)<<setiosflags(ios::left)
            <<setfill('.')<<setw(15)<<StudetnPotok[i].FIO;
      cout<<setw(10)<<setiosflags(ios::right|ios::fixed)
            <<setfill('_')<<StudetnPotok[i].bal<<'\n';
  }
}
int main (int argc, char * const argv[])
{
```

- // Цей фрагмент виведе на екран наступний результат:
- myOut(cout,Stud,n);
- ofstream fileout("o.txt");
- myOut(fileout,Stud,n);
- return 0;
- }
- **Методи обміну з потоками**
- У потокових класах поряд з операціями вилучення >> і включення << визначені методи для неформатованого читання і запису в потік

- (при цьому перетворення даних не виконуються). Нижче наведені функції читання, визначені в класі istream.

| | |
|---|---|
| gcount() | — возвращает количество символов, считанных с помощью последней функции неформатированного ввода; |
| get() | — возвращает код извлеченного из потока символа или EOF; |
| get (c) | — возвращает ссылку на поток, из которого выполнялось чтение, и записывает извлеченный символ в с; |

| | |
|---|---|
| get(buf,num,lim='\n') | — считывает num-1 символов (или пока не встретится символ lim) и копирует их в символьную строку buf. Вместо символа lime строку записывается признак конца строки ('\0'). Символ lim остается в потоке. Возвращает ссылку на текущий поток; |
| getline(buf, num, lim='\n') | — аналогична функции get, но копирует в buf и символ lim; |
| ignore(num = 1, lim = EOF) | — считывает и пропускает символы до тех пор, пока не будет прочитано num символов или не встретится разделитель, заданный параметром lim. Возвращает ссылку на текущий поток; |

| | |
|---|---|
| peek() | — возвращает следующий символ без удаления его из потока или EOF, если достигнут конец файла; |
| putback(c) | — помещает в поток символ c, который становится текущим при извлечении из потока; |
| read(buf, num) | — считывает num символов (или все символы до конца файла, если их меньше num) в символьный массив buf и возвращает ссылку на текущий поток; |
| readsome(buf, num) | — считывает num символов (или все символы до конца файла, если их меньше num) в символьный массив buf и возвращает количество считанных символов; |
| seekg(pos) | — устанавливает текущую позицию чтения в значение pos; |

| | |
|---|---|
| seekg(offs, org) | — перемещает текущую позицию чтения на offs байтов, считая от одной из трех позиций, определяемых параметром org: ios::beg (от начала файла), ios::cur (от текущей позиции) или ios::end (от конца файла); |
| tellg() | — возвращает текущую позицию чтения потока; |
| unget() | — помещает последний прочитанный символ в поток и возвращает ссылку на текущий поток. |
| flush() | — записывает содержимое потока вывода на физическое устройство; |
| put(c) | — выводит в поток символ c и возвращает ссылку на поток; |
| seekg(pos) | — устанавливает текущую позицию записи в значение pos; |

| | |
|---|---|
| seekg (offs, org) | — перемещает текущую позицию записи на offs байтов, считая от одной из трех позиций, определяемых параметром org: ios::beg (от начала файла), ios::cur (от текущей позиции) или ios::end (от конца файла); |
| tellg() | — возвращает текущую позицию записи потока; |
| write(buf, num) | — записывает в поток num символов из массива buf и возвращает ссылку на поток. |

**Помилки потоків**

У базовому класі ios визначено поле state, яке являє собою стан потоку у вигляді сукупності бітів:

```
enum io_state    {
goodbit      = 0x00,// Немає помилок
eofbit   = 0x01,// Досягнуто кінця файлу
fail bit = 0x02,// Помилка форматування або перетворення
```

- badbit = 0x04, // Серйозна помилка, після якої
- // користуватися потоком неможливо
- hardfail = 0x08 // Несправність обладнання
- }:
- Станом потоку можна керувати за допомогою перерахованих нижче методів і операцій:

| | |
|---|---|
| int rdstate() | — возвращает текущее состояние потока; |
| int eof() | — возвращает ненулевое значение, если установлен флаг eofbit; |
| int fail() | — возвращает ненулевое значение, если установлен один из флагов failbit, badbit или hardfail; |
| int bad() | — возвращает ненулевое значение, если установлен один из флагов badbit или hardfail; |

| | |
|---|---|
| int good() | — возвращает ненулевое значение, если сброшены все флаги ошибок; |
| void clear(int = 0) | — параметр принимается в качестве состояния ошибки, при отсутствии параметра состояние ошибки устанавливается 0; |
| operator void*() | — возвращает нулевой указатель, если установлен хотя бы один бит ошибки; |
| operator !() | — возвращает ненулевой указатель, если установлен хотя бы один бит ошибки. |

Далі наведені часто використовувані операції з прапорами стану потоку.

// Перевірити, чи встановлений прапор flag;

if(stream_obj.rdstate() & ios::flag)

// Скинути прапор flag:

stream_obj.clear(rdstate() & ~ios::flag)

- // Встановити прапор flag:
- stream_obj.clear(rdstate() | ios::flag)
- // Встановити прапор.flag і скинути всі інші:
- stream_obj.clear(ios::flag)
- // Скинути всі прапори:
- stream_obj.clear()
- Операція vo1d * () неявно викликається всякий раз, коли потік порівнюється з 0. Це дозволяє записувати цикли виду:
- while (stream_obj){
- **// Все в порядку, можна проводити введення / виведення**
- **}**