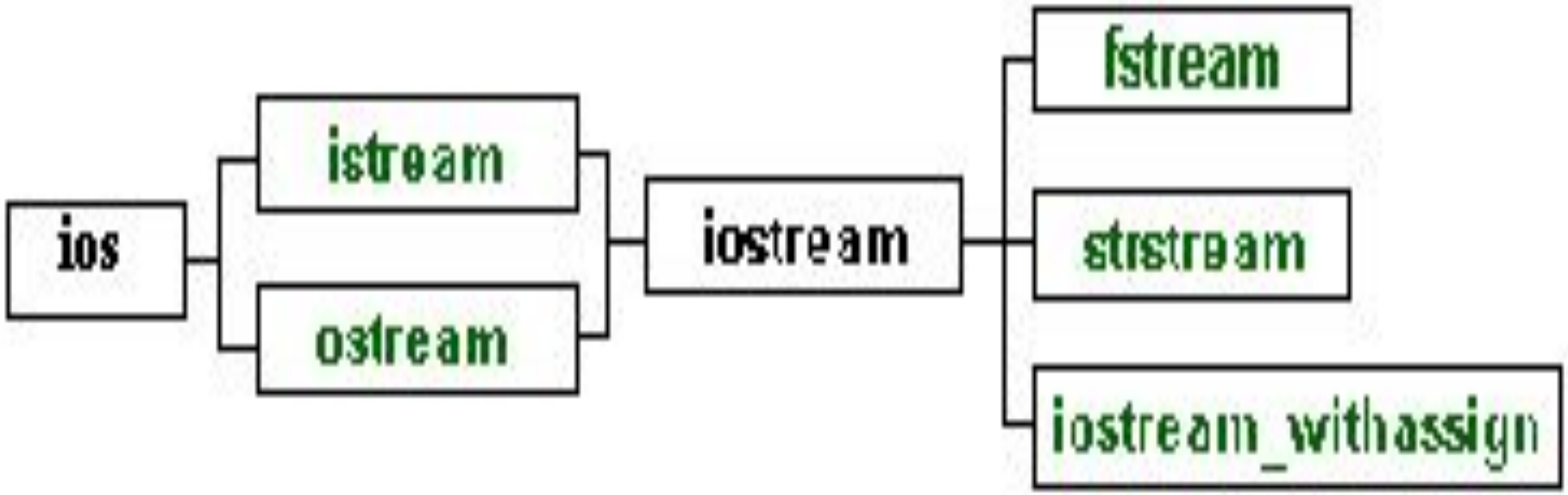
Об’єктно-орієнтоване проектування та моделювання

Лектор
Доцент НТУУ «КПІ»
Ковалюк Тетяна Володимирівна
tkovalyuk@ukr.net

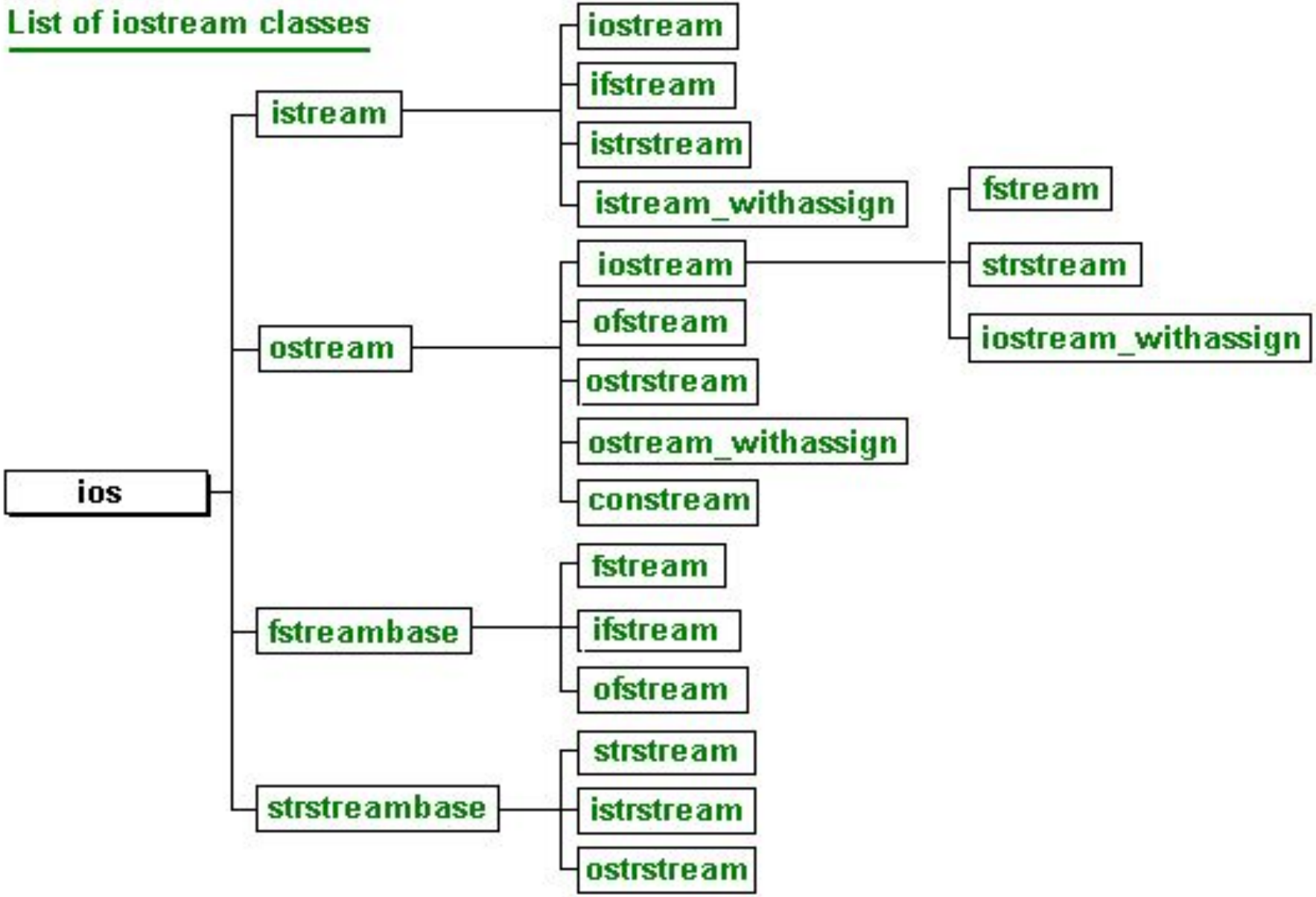
Лекція 2 Класи потоків

Ієрархія класів потоків

- Класи потоків утворюють бібліотеку для реалізації операцій введення та виведення даних потоком.
- Для реалізації введення даних використовується клас **istream**, для виведення − клас **ostream.**
- Для файлового введення-виведення використовується клас **fstream.**



Повна схема ієрархії класів потоків



Ієрархія класів потоків

Клас **istream** визначає об’єкт **cin,**
клас **ostream** визначає об’єкт **cout.**
Об’єкт **cerr** визначає вихідний потік помилок
Для виконання операцій запису та зчитування даних використовуються перевантажені за допомогою операторних функцій **operator** операції:
<< для виведення,
>> для введення
різних типів даних.

Приклад:

```
char name[10]; int number=10;
cout<<“HELLO ”<<number;
cin>>name>>number;
```

Різновиди методів класів потоків

Класи потоків містять різні методи, за допомогою яких здійснюється операції:

- **Форматування** даних під час виведення,
- **Доступ** до файлів під час їх відкриття,
- **Читання та запис** даних у файл
- **Введення** даних з клавіатури
- **Виведення** даних на екран
- та інші операції.

Функції –члени класу ios

Формат функції	Призначення функції
Стан потоку введення та виведення	
int bad();	Повертається ненульове значення, якщо встановлено один з прапорів: **badbit, hardfail**
int good();	Повертається ненульове значення, якщо не встановлено біти помилок
void clear(int);	Біти обнулюються, якщо параметр дорівнює 0, інакше параметр визначається як стан помилок

Функції –члени класу ios

int rdstate();	Повертається поточний стан потоку
int eof();	Повертається ненульове значення, якщо встановлено біт кінця файлу
int fail();	Повертається ненульове значення, якщо операція завершилася невдало під час перевірки бітів прапорів: **badbit, hardfail, failbit**

Управління прапорами форматування

long flags(); **long flags(long);**	Прапори форматування визначають формат потоку даних, які вводяться та виводяться. Прапори є бітовими полями змінної типу **long,** їх значення повертаються і встановлюються функціями згідно з параметрами
long setf(long) ;	Встановлюються прапори у відповідності до параметру
long unsetf(long);	Прапори формату ігноруються
long setf (long _setbits, long _field);	Встановлюються прапори формату, присвоює значення бітів першого параметра бітам другого параметра

Функції, що форматують

char fill(); **char fill(char);**	Повертається поточний **символ заповнення**, якщо ширина поля менша за задану або встановлює новий символ заповнення. Якщо не вказано заповнювач, то використовується символ **пробіл**.
int precision(int); **int precision();**	Встановлюється **точність** (кількість значущих цифр) дійсних чисел або повертається її поточне значення. Якщо точність не вказано, то використовується значення шість.
int width(); **int width(int);**	При введенні встановлюють максимальне число символів, які читаються, при виведенні задають **мінімальну ширину поля** виведення. Якщо ширину не вказано, то приймається значення 0.

Прапори форматування

skipws	Ігноруються пробіли, що передують символам
left	Вирівнюються дані зліва під час виведення
right	Вирівнюються дані справа під час виведення
internal	Знак числа виводиться зліва, число вирівнюється справа
dec	Число у десятковій системі числення
oct	Число у восьмиричній системі числення
hex	Число у шістнадцятковій системі числення
showbase	Під час виведення додається індикатор основи системи числення

Прапори форматування

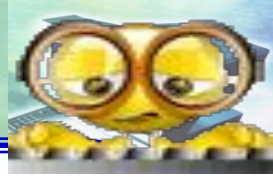
showpoint	Виводиться десяткова точка (floating output)
uppercase	Виводяться букви у верхньому регістрі
showpos	Виводиться знак "+" для додатних значень
scientific	Виводяться числа у форматі X.XXXXEXX
fixed	Виводяться числа у форматі XXX.XX
unitbuf	Очищається буфер потоку після кожної операції розміщення даних у пам’яті
stdio	Очищаються потоки stdout, stderr після кожного розміщення даних у пам’яті

Прапори помилкових станів

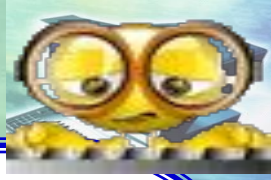
goodbit	Немає помилок (no bit set: all is ok)
eofbit	Кінець прапорця
failbit	Помилка форматування
badbit	Поважна помилка
hardfail	Помилка, що не виправляється

Біти відкриття файлів

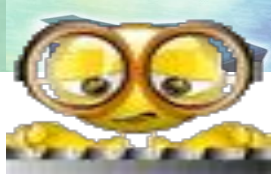
app	Доповнення даних. Дані записуються завжди в кінець файлу
ate	Перехід до кінця файлу під час його відкриття
in	Відкриття файлу для введення даних (default for ifstreams)
out	Відкриття файлу для виведення даних (default for ofstreams)
binary	Відкриття файлу в бінарному режимі
trunc	Вилучення даних, якщо файл існує
nocreate	Якщо файл не існує, не відкривати його
noreplace	Якщо файл існує, не можливо відкрити його для виведення, доки не встановлено значення *ate* чи *app*

```
//Пример использования форматирования
#include<iostream.h>
#include<conio.h>
#include<iomanip.h>    //для манипуляторов, например, endl
void main()
{ float f;  int x;char c;  double y;
  cout<<"Ввод и вывод встроенных типов.Сцепленные потоки
\endl";
  cout<<"ввод чисел: f-float,x-integer : ";
  cin>>f>>x;            //сцепленные потоки
  cin.get(c);   //извлечение (ввод) символа <enter>, метод класса
istream
  cout<<"Исходные числа: f="<<f<<"  x="<<x<<"\n";
  cout<<"press any key to continue...";
  getch();
  cout<<"\n              Форматирование \n";
  cout<<"              Вывод целых чисел\n";
```

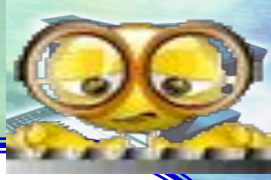



```
//Пример использования форматирования
 cout<<"Установлены форматы:width(10),fill('0')\n";
 cout<<"Результат форматированиЯ:\n";
 cout.width(10); //установить ширину поля вывода=10,
//метод класса ios
  cout.fill('0')  ;   //заполнить недостающие символы нулями,
//метод класса ios
  cout<<x<<"-введеннаЯ переменнаЯ\n";
 //установки действуют только на вывод значения,
//следующего за установками, надо повторить
```



```
//установки для вывода следующего значения
  cout.width(10);
  cout.fill('0');
  cout<<64<<"-заданнаЯ константа\n";    //вывод в 10-й с.с.
  cout<<"Установлены форматы вывода констант в 16-й с.с.:\n";
  cout.setf(ios::hex,ios::basefield);          //работает при //выводе
констант, метод класса ios
  cout<<"число 64->"<<64<<", число 128 ->"<<128<<"\n";
  cout<<"Установлен флаг отображениЯ системы счислениЯ:\n";
  cout.setf(ios::hex|ios::showbase);
  cout<<"число 64->"<<64<<", число 128 ->"<<128<<"\n";
  cout<<"Установлен флаг showpos - отображениЯ знака + \n";
  cout.setf(ios::showpos);                                  //показать знак
“+”
  cout.setf(ios::dec,ios::basefield);   //10-я с.с.
  cout<<"  x="<<x<<"\n";
```

```
cout<<"сброшен флаг skipws-игнорирования ведущих пробелов\n";
 cout<<"ПроверЯетсЯ состояние потока: если нет пробелов, то вывод\n";
 cin.unsetf(ios::skipws); //сбросить флаг игнорирования
                          // пробелов в начале числа
 cout<<"input y= ";  cin>>y;
 if (cin.good()) //проверка ошибочных состояний при
                 // вводе
   {
    cout<<" y="<<y<<"\n";
    cout<<"установлен флаг scientific-вывод в экспоненциальной форме\n";
    cout.setf(ios::scientific);
    cout<<"y="<<y<<"\n";
   }
```

```
//если в начале числа пробелы, то сообщение //"неправильный "
   else cout<<"неправильный ввод-перед числом пробелы\n";
 cout<<"       Вывод вещественных чисел\n";
 cout<<"Установлены форматы: precision(4), scientific, width(10),
fill('0')\n";
 cout.precision(4);
 cout<<"f="<<f<<"\n";
 cout.fill('0');
 cout.width(10);
 cout.setf(ios::fixed);
 cout.unsetf(ios::showpos);
 cout<<f<<" -fixed format with fill('0')\n";
 }
```

Маніпулятори

Маніпулятори − це функції, які використовуються для управління прапорами потоку та включаються у ланцюг операцій << або >>.

Визначення маніпуляторів подано в файлі ***iomanip***

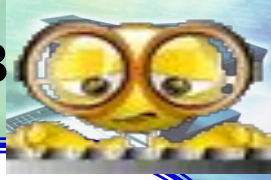
Специфікація маніпуляторів

Прості маніпулятори	Дії маніпуляторів
endl	Символ '\n' розмістити у вихідний потік і викликати маніпулятор flush
ends	Символ '\0' розмістити у вихідний потік
flush	Очистити потік
dec	Встановити основу системи числення 10
hex	Встановити основу системи числення 16
oct	Встановити основу системи числення 8
ws	Ігнорувати пробіли, що йдуть попереду при введенні даних

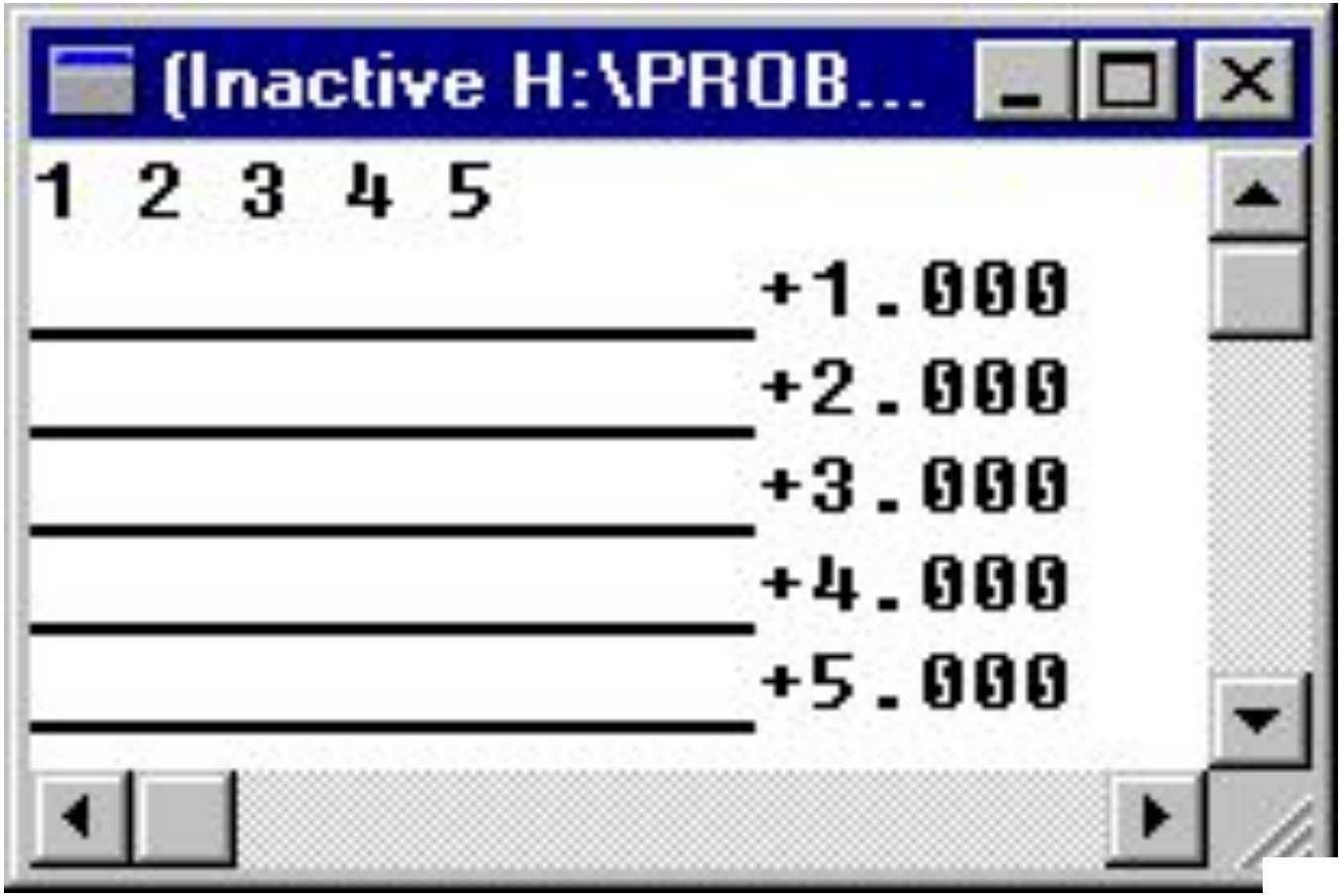
Специфікація маніпуляторів

Параметризовані маніпулятори	Дії маніпуляторів
setbase(int)	Задається основа для перетворення в іншу систему
setfill(int)	Задається символ, що заповнює
setprecision(int)	Задається точність дійсних чисел
setw(int)	Задається ширина поля (максимальна при введенні, мінімальна при виведенні)
setiosflags(long)	Встановлюються прапори у відповідності до параметра
resetiosflags(long)	Ігноруються прапори у відповідності до параметра

Приклад використання маніпуляторів



```
//Использование манипуляторов
#include<iostream>
#include<iomanip>
using namespace std;
void main()
  { double a[5];
    for(int i=0;i<5;i++)
       cin>>a[i];
  cout<<setprecision(10)<<
          setiosflags(ios::showpoint |
          ios::showpos)<<setfill('_');
 for( i=0;i<5;i++)
   cout<<setw(20)<<a[i]<<endl;
  }
```


(Inactive H:\PROB...
1 2 3 4 5
+1.000
+2.000
+3.000
+4.000
+5.000

Методи класів ISTREAM,OSTREAM

Клас ISTREAM	
Метод	Призначення
int get()	зчитує черговий символ
int gcount()	повертає кількість зчитаних символів
istream& get(char&);	читати символ з потоку і помістити за адресою
istream& get(char*, int len, char = '\n');	читати задану кількість символів у буфер, припинити операцію в разі зчитування **enter**
istream& read(char*, int);	читати задану кількість бінарних даних в буфер
istream& getline(char*, int, char = '\n');	зчитати задану кількість символів текстового рядка в буфер

Методи класів ISTREAM,OSTREAM

istream& seekg (streamoff offset, seek_dir dir)	встановити покажчик потоку get зі зміщенням offset від позиції ios:: beg, ios:: end, ios:: cur
long tellg();	визначити поточне положення покажчика потоку get
istream &operator>>(istream&)	витягти з потоку об'єкти зазначених типів

Методи класів ISTREAM,OSTREAM

Класс OSTREAM	
ostream& flush();	зкинути буфер streambuf
ostream& put(char ch);	помістити символ в потік
ostream& write(const char*, int n);	запис заданої кількості символів з буфера в потік
ostream& seekp(streampos);	помістити покажчик потоку put на задану позицію
ostream &operator <<(ostream&)	помістити в потік задані об'єкти

Ввод-вивід типів, що визначені користувачем

Перевантаживши функції **operator <<** та **operator>>,** визначені в класах **istream, ostream**, користувач може вводити й виводити дані власних типів.

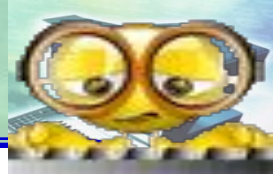
При цьому не вносяться зміни в визначення класів **istream, ostream**.

Досить визначити функції читання з потоку

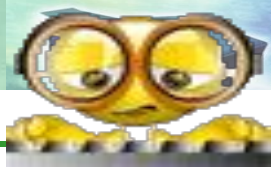
istream operator>>

і запису даних у потік

ostream operator <<.

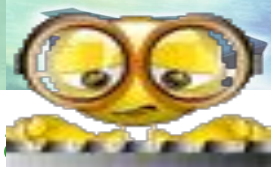



```
//lec16_3.cpp Ввод -вывод типов, определенных
//пользователями (комплексных чисел в виде(re,im))
#include<iostream.h>
class complex
 {
public : double re,im;
complex(double r=0,double i=0):re(r),im(i){}
// конструктор инициализации
// дружественные методы переопределения бинарных
//операций (имеют 2 параметра)
   friend complex operator +(complex &c1,complex &c2)
            {return complex(c1.re+c2.re,c1.im+c2.im);}
   friend double real(complex& a)  {return a.re;}
   friend double imag(complex& a) {return a.im;}
  };        //class complex
```

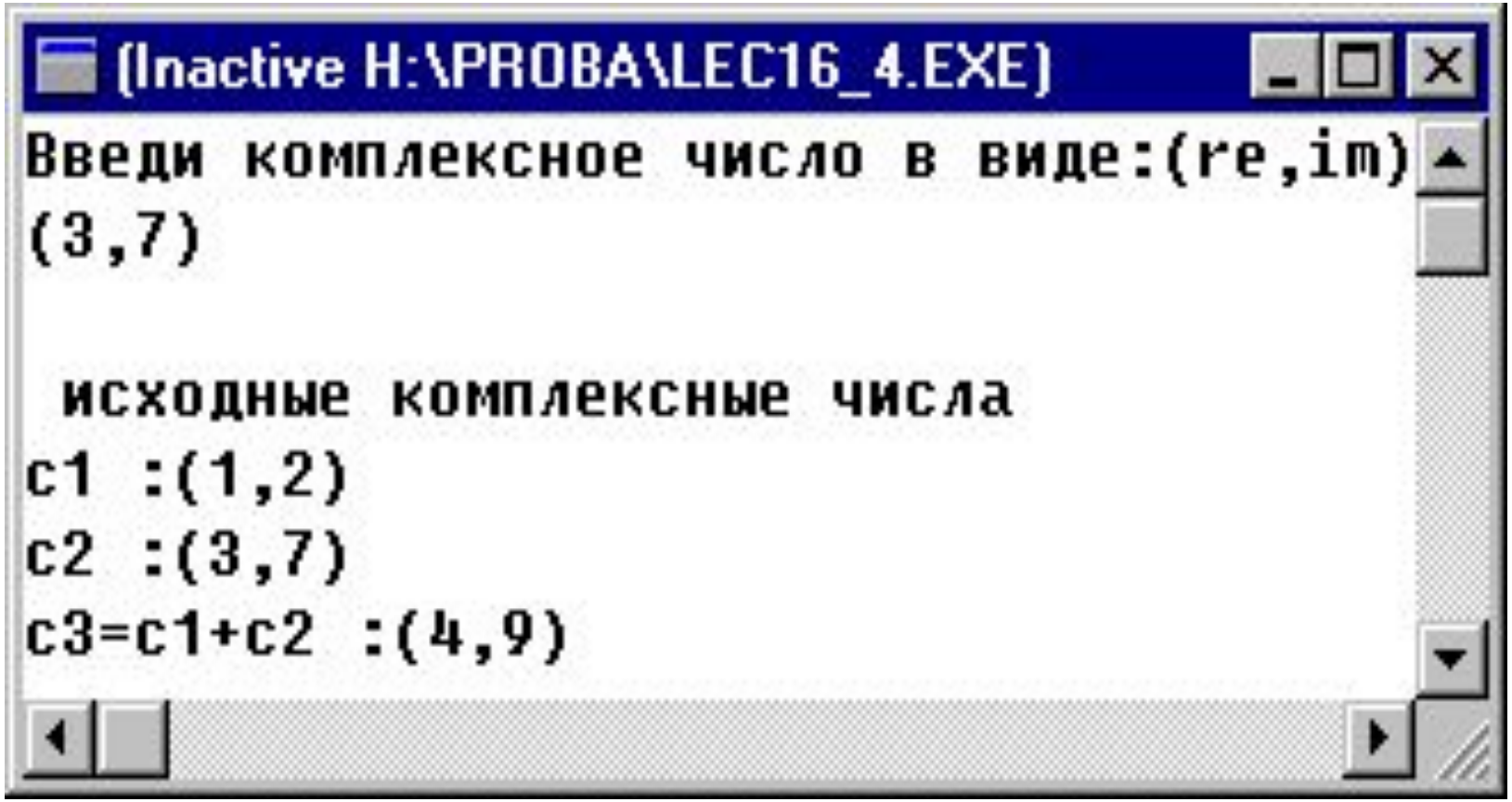


```
//----операторная функция ввода комплексных чисел----
 istream& operator >> (istream& is, complex& a)
//перегруженная операторная ф-ция >>
{ double re=0, im=0;              //параметры ф-ции - ссылки на
                                  //вводимое значение и объект типа complex.
  char c=0;                       // ввод в виде (re,im)
   is>>c;          //ввод символа и проверка наличия символа ‘(’
  if( c=='(' )                    //если введен ‘(’,
    { is>>re>>c;                  //то ввод действительной
                                  //части и разделительного символа
      if(c==',')  is>>im>>c;     // если введен символ ‘, ‘, то
                                  //ввод мнимой части
      if(c!=')')  is.clear(ios::badbit); //проверка наличия
                                  // ошибок ввода и установка флага badbit
    }
```

```
else { is.putback(c);                    //если первый символ не
                                  // ‘(’,то поместить символ обратно в поток
  is>>re; }                      // и ввести действительную частьчисла
 if(is) a=complex(re,im);          // если введены значения, то
                                    // определить комплексное число
 return is;                        // и возвратить его из функции
  };
```



```
//------операторная функция вывода комплексного числ
 ostream& operator << (ostream& os, complex& a)
 {   os<<'('<<real(a)<<','<<imag(a)<<')';
   return os; }                          //ostream
//--------------------основная программа-------------
void main ()
 {complex c1(1,2);                  // объекты с инициализацией в
                                    //точке объявления
complex c2,c3;     //объекты без параметров - вызов
                   //конструктора и инициализация значениями 0
cout<<"Введи комплексное число в виде:(re,im)\n";
cin>> c2;   c3=c1+c2;
cout<<"\n исходные комплексные числа " <<endl;
cout<<"c1 :" <<c1 << "\n";     cout<<"c2 :" <<c2 << " \n";
cout<<"c3=c1+c2 :" <<c3 <<"\n";
}
```


(Inactive H:\PROBA\LEC16_4.EXE)
Введи комплексное число в виде:(re,im)
(3,7)
исходные комплексные числа
c1 :(1,2)
c2 :(3,7)
c3=c1+c2 :(4,9)

Файлове введення-виведення

- Використовують класи **ifstream, ofstream, fstream** для операцій з вхідними і вихідними файлами.
- Ці класи є похідними від класів **istream, ostream**, значить, успадковують операції <<,>>, маніпулятори, прапори формату, стану потоків і пр.
- Класи потоків оголошені в заголовному файлі **fstream.**

Конструктори класів

Функції конструктора

1. Конструювання об'єкта, відкриття файлу для читання, прикріплення об'єкта до файлу.
2. Існуючий файл зберігається.

- Новий файл створюється шляхом додавання записів у кінець.
- За замовчуванням файл не створюється, якщо не існував раніше.

  ifstream:: ifstream (const char * name, int mode = ios:: in, int = filebuf:: openprot);

  Параметри конструктора: **ім'я файлу, прапор стану файлу, тип захисту**.
- За замовчуванням тип захисту: S_IREAD | S_IWRITE.

  Приклад: ifstream fileobj(‘name.txt’,ios::in|ios::out);

Конструктори класів

2. Конструювання об'єкта і прикріплення до вже відкритого файлу по дескриптору (fd).

- Об'єкт використовує буфер, специфікований char * buf довжини buf_len.

ifstream:: ifstream (int fd);

3.Конструювання об'єкта і прикріплення до вже відкритого файлу, специфікованому дескриптором (fd).

- Об'єкт використовує буфер, специфікований char * buf довжини buf_len.

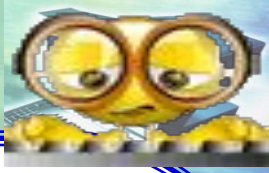
ifstream:: ifstream (int fd, char * buf, int buf_len);

4. Конструювання об'єкта без прив'язки до нього файлу.

ifstream:: ifstream ()

Аналогічно для класів ofstream, fstream.

Методи відкриття файлів



За допомогою конструктора

```
//ввести ім'я файлу, прочитати файл і роздрукувати
// посимвольно через кому
#include<iostream.h>
#include<fstream.h>
#include<stdio.h>
void main()
{
  cout<<"name: ";                    //вивід повідомлення
  char fname[10], c;                 // ім'я файлу
  cin>>fname;                        //ввід імені файла
  ifstream ifs(fname);               //оголошення об’єкта та
    //відкриття файлу з ім'ям, заданим в fname
```

```
if (!ifs)                                  //якщо файл не відкрито
{
 cout<<"file is not opened"<<endl;       }
 while(ifs)                              // поки не кінець файла
 {
   ifs.get(c);                           // читати символ з файла
  putc (c,stdout);      //писати його в стандартний потік виводу
 cout<<',' ;                             //писати символ ,
  }
}
```

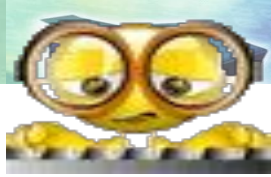
Відкрити файл за допомогою метода open()

```
void open(const char *name, int mode,
int prot=filebuf::openprot);
```

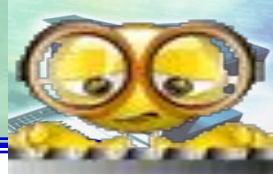
Метод є членом класів **fstream, fstreambase, ifstream, ofstream**

Параметр **mode** описує режими доступу, які визначені в класі ios:

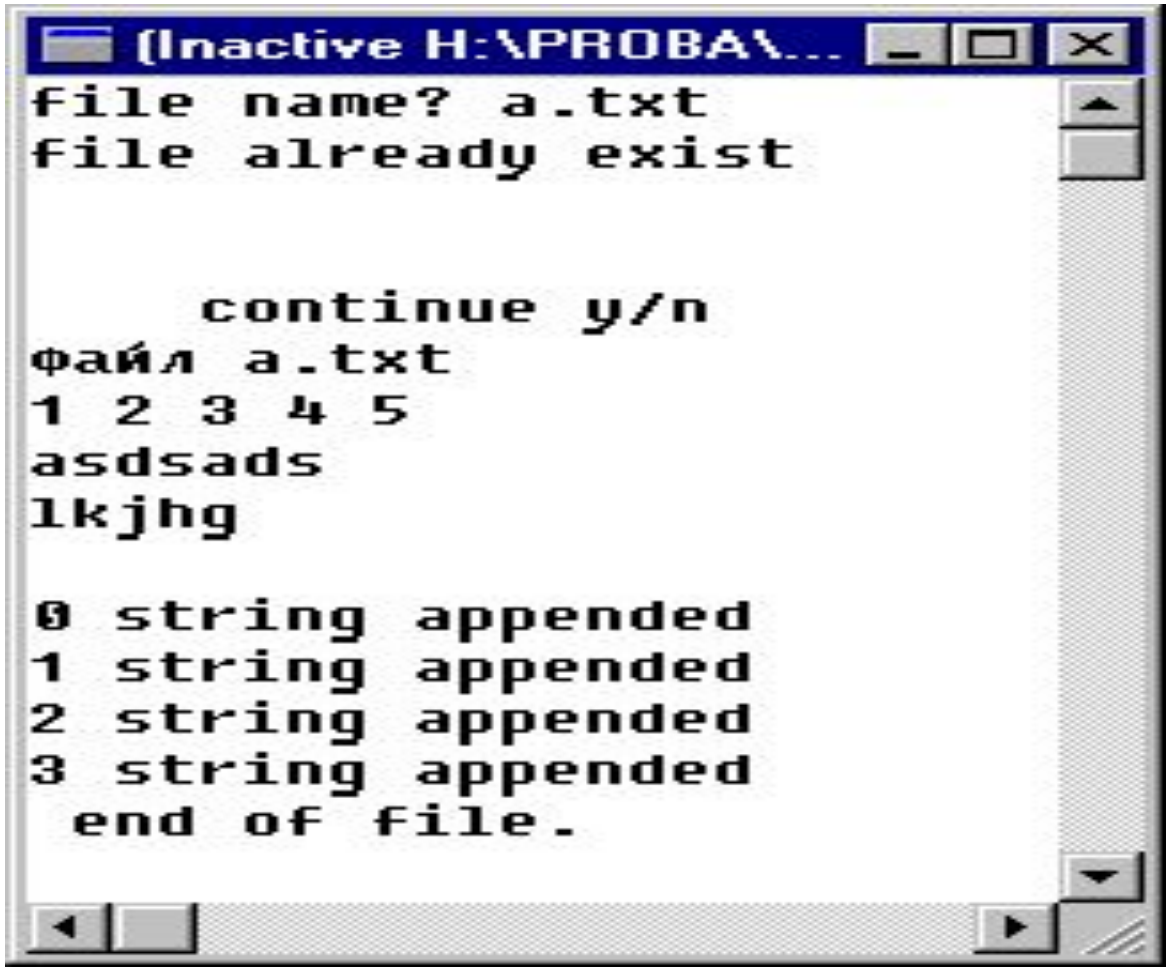
```
enum open_mode  {
 app,     /*     Append data--always write at end of file.*/
 binary,     /*Open file in binary mode.*/
 ate,          /*  Seek to end of file upon original open. */
 trunc,   /*Discard contents if file exists.*/
 in,           /* Open for input (default for ifstreams).   */
 nocreate,   /*If file does not exist, open fails.*/
 out,       /* Open for output (default for ofstreams).*/
 noreplace, /*If file exists, open for output fails unless ate
or app is set.*/
};
```



```
#include<iostream.h>
#include<fstream.h>
#include<stdio.h>
#include<conio.h>
 void main()
{ cout<<"name";
  char fname[10], ch;  int i=0;
  cin>>fname;              //ввод имени файла
 ofstream  ofs(fname,ios::out);   //определить  объект  //класса  ifstream,
открыть файл для записи, не заменять, если файл //существует
if (!ofs)                    //если файл уже существует
     cout<<"file already exist"<<endl;  //вывод сообщений
    else                          //иначе
      ofs<<"new string";       // запись в файл строки
   ofs.close();                  //закрыть файл
   fstream fs;                   //определить новый объект
   fs.open(fname,ios::out|ios::ate);     //открыть файл для
                                 //записи в режиме дополнения
```

```
do{
     fs<<i<<" string appended\n";    //дополнить новой строкой
      i++;
     gotoxy(5,3);  cout<<”continue y/n” ; ch=getch();
   } while(ch!=’n’);
fs.close();                          //закрыть файл
fs.open(fname,ios::in);    //открыть для чтения
while(fs)                          //читать символы, писать их на экран
  {
    char c;
    fs.get(c);                    //читать из файла в память
   fs>>c;
      putc  (c,stdout);   }  //вывод  в  стандартный  поток,  т.е.  на
//экран
   cout<<'.';
 }
```


(Inactive H:\PROBA\...
file name? a.txt
file already exist
continue y/n
файл a.txt
1 2 3 4 5
asdsads
lkjhg
0 string appended
1 string appended
2 string appended
3 string appended
end of file.

Методи читання та запису

Функція-член read () класів istream, ostream витягує задану (int) кількість символів у буфер (char *). Використовуються для бінарних файлів

istream& read(char*, int);
istream& read(signed char*, int);
istream& read(unsigned char*, int);

Функція-член write () класу поміщає в потік вказану (n) кількість символів з буфера (char *). Використовуються для бінарних файлів.

ostream& write(const char*, int n);
ostream& write(const signed char*, int n);
ostream& write(const unsigned char*, int n);

Методи читання та запису

Функція-член get () класу витягує задану кількість символів в масив. Використовуючи gcount (), визначають кількість витягнутих символів, якщо виявлена помилка.

int get();

Функція витягує задану кількість символів по заданому вказівником до роздільника, кінця файлу.

istream& get(char*, int len, char = '\n');

istream& get(signed char*, int len, char = '\n');

istream& get(unsigned char*, int len, char = '\n')

Функція витягує один символ в задане посилання.

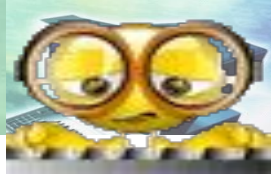
istream& get(char&);

istream& get(signed char&);

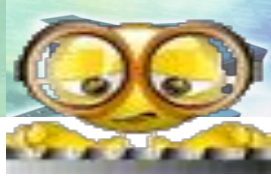
istream& get(unsigned char&);

Функція витягує символи в буфер до першого роздільника.

istream& get(streambuf&, char = '\n');

```
//Использование read(),write()
#include<iostream.h>
#include<fstream.h>
#include<stdio.h>
void main()
{cout<<" function read, write"<<endl;
  char fname[10];
  char c;
  char msg[80];
  cout<<"file name? ";
  cin>>fname;
 fstream fs, ofs;  //объявление объектов файлового потока для записи и чтения
 ofs.open(fname,ios::out|ios::binary|ios::ate);   //откр
для записи как бинарный
```

```
if (ofs)
 for(int i=0;i<3;i++)
    {  cout<<"string? "; cin>>msg;
       ofs.write(msg,sizeof(msg));     //запись в файл
     }
  ofs.close();
  fs.open(fname,ios::in|ios::binary);            //открыть файл
                                                 // для чтения как бинарный
  while(!fs.eof())
     {    fs.read(msg,sizeof(msg)); //чтение из бинарного файла
          if (fs.eof()==0)                  //чтобы последнюю
                                            // запись не печатать дважды
              puts(msg);          //если eof()==0, то конец
                                  //файла не достигнут, если  eof()==1, то
          }                                 //достигнут конец файла
fs.close();
}
```

Кінець лекції 2