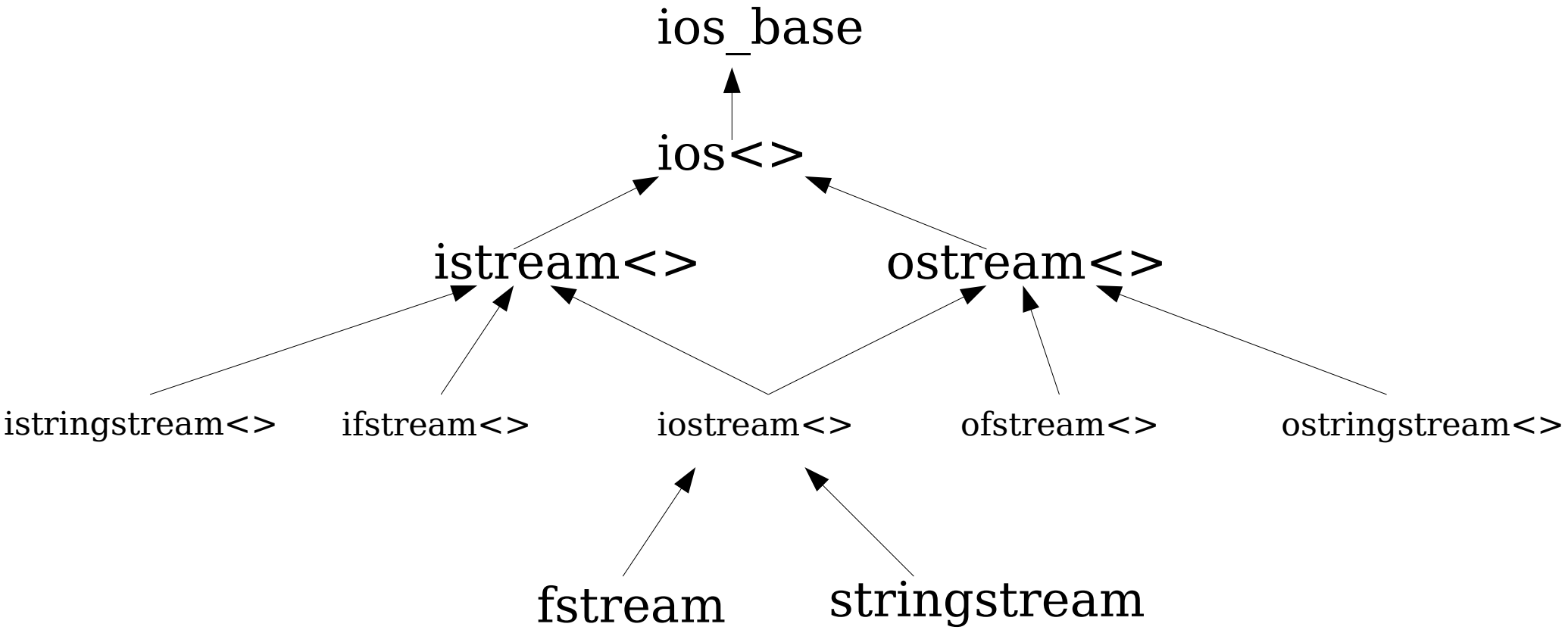
Introducción a los flujos de caracteres: Archivos, cadenas.

Prof. Andrés Arcia

Gerarquía de Clases

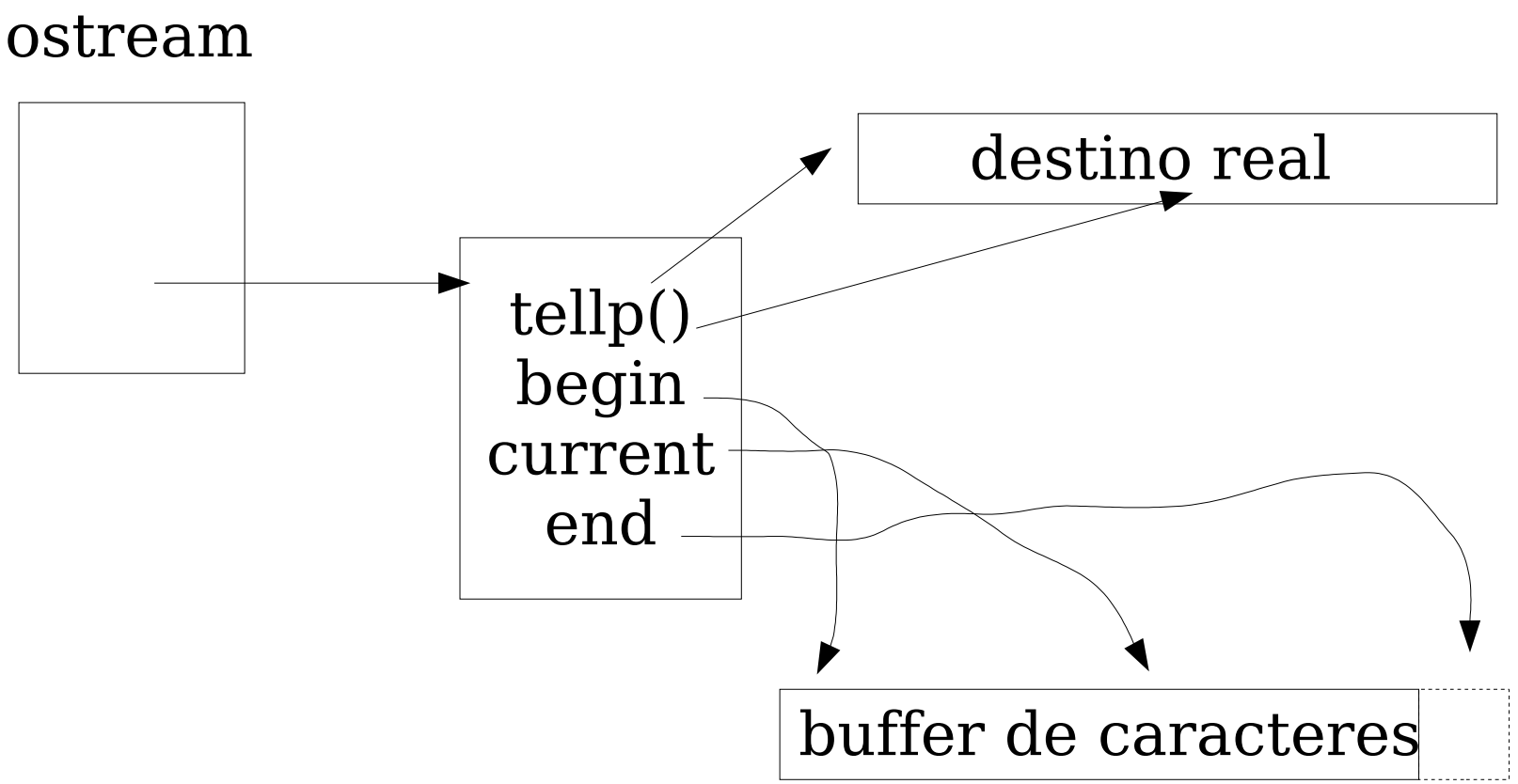


- El sufijo <> indica clases plantillas.
- Tanto los archivos como las cadenas son flujos de datos de los que se puede leer y escribir.

Buffering

- Antes de guardar datos al disco, se aplica una técnica denominada: buffering.
- El buffering consiste en almacenar datos en un arreglo hasta desbordarlo, que es cuando se procede a vaciarlo en el disco.
- Esto da un mejor desempeño, pues se trabaja sobre la memoria RAM que es más rápida.

Cómo se representa un Buffer



Cómo trabaja I/O streams

- Debe incluirse el archivo de cabecera iostream.
- Incluye varios prototipos de las funciones operator<<() y operator>>(). Bien conocidos como inserción (>>) y extractor (<<).

```
istream & operator >>(istream& s, char *pDest)

istream & operator >>(istream& s, int &dest)

//salida

ostream& operator<<(ostream& dest, char *pSource)
```

Cómo trabaja I/O Stream

- Observe que tanto las funciones de entrada como las de salida devuelven una referencia al manejador base. Se utiliza para hacer secuencias en cadena.
  - cout << a << b; => ((cout << a) << b);
- El primer parámetro es también una referencia al manejador base. Se utiliza para conocer al manejador.
  - cout << a; => operator<<(cout, a);

Algunas consideraciones

- Al compilar una llamada a funciones derivadas de ios, primero se determina de qué tipo (ent/sal) y luego el tipo de dato a imprimir.
  - Ej:
  - cout << “Introducción a las cadenas”;
- Vé cual es el tipo del parametro?
- Los objetos cin, cout y cerr, son creados justo antes de comenzar cada programa.

Algunas consideraciones

- Los objetos istream y ostream contienen datos miembros necesarios para mantener cuenta de la salida.

Subclases de fstream

- Las clases ofstream, ifstream y fstream están definidas en archivos con sus mismos nombres.
- La firma del constructor para fstream es:
  - fstream::fstream(const char *p, openmode m)
  - Para ofstream m==out.
  - Para ifstream m==in.

Modos de Apertura

app	agrega al final del archivo
ate	Abre y se posiciona al final de un archivo.
binary	I/O hecha en modo binario
in	abre para lectura
out	abre para escritura

Ejemplo

```
void fn()
{
    ofstream a1(“primero”);
    ofstream a2(“segundo”, ios::binary | ios::ate);

    a1 << “texto” << endl;
    a2 << 2.5 << 3 << endl;
}
```

Errores

- bool good() // pregunta si la próxima operación va a tener éxito.
- bool eof() // pregunta si se ve el fin de archivo.
- bool fail() // lógica invertida de good
- bool bad() // si el flujo está corrompido

Ejemplo

```
void fn()
{
   ifstream a(“enteros.in”);
   ofstream b(“otro.out”);
   int buf;
   while (!a.eof())
        { a>>buf;
          b<<buf;}
}
```

Otro ejemplo

```
ifstream transaccion(“TRANS”);
if (transaccion.bad())
{ cerr << “No se pudo abrir el archivo de transacciones”;
   return;}
while (!transaccion.eof())
{ transaccion >> numeroCuenta >> tipoTrans >> monto; }
```

StringStreams

- Permite que todos los métodos de fstream sean aplicados a cadenas.
- Las clases son:
  - istringstream, ostringstream, stringstream.

Ejemplo

```
#include <sstream>
#include <iostream>

using namespace std;
int main()
{ char * test = “23 34 2 34 43 5”;
   istringstream  a(test);
   int b;
   while (!a.eof())
   { a >> b;
      cout << b; }  }
```

Modificadores de Formato (Manipulators)

- Existen varias funciones miembros de los objetos stream que permiten modificar el formato de la salida.
- Los modificadores pueden ser utilizados desde dos perspectivas: como funciones miembro y como cadenas modificadoras.

Modificadores como Funciones Miembro

- Modificador de formato: precision (en ingles c antes de s).
- Fija un número de digitos máximo a mostrar durante una impresión numérica.
- Sintaxis:
  - int ostream::precision(int)
  - El valor de retorno corresponde al valor previo. Se guarda en caso de querer regresar al valor antiguo.

Ejemplo

```
 #include <iostream>
using namespace std;
void h(float interes, float cantidad)
{
  int pres_backup;
  cout << "Cantidad en bolivares: ";
  pres_backup = cout.precision(2);
  cout << cantidad;
  cout.precision(4);
  cout << " al " << interes << endl;
  // vuelta a la normalidad
  cout.precision(pres_backup);
}
```

```
int main()
{
  h(0.23455, 342);
  h(0.6, 43);
  h(0.12, 2);
}
```

```
Cantidad en bolivares: 3.4e+02 al 0.2345
Cantidad en bolivares: 43 al 0.6
Cantidad en bolivares: 2 al 0.12
```

Cadenas Modificadoras

- Las cadenas modificadoras se utilizan como si fuesen parte de la salida. Luego ellas cumplen una función de cambio.
- La sintaxis varia de acuerdo a la función utilizada. En general es una llamada a función.

Ejemplo

```
#include <iostream>
#include <iomanip>
using namespace std;
void h(float interes, float cantidad)
{
  cout << "Cantidad en bolivares: ";
  cout << setprecision(2);
  cout << cantidad;
  cout << setprecision(4);
  cout << " al " << interes << endl;
}
```

```
int main()
{
  h(0.23455, 342);
  h(0.6, 43);
  h(0.12, 2);
}
```

```
Cantidad en bolivares: 3.4e+02 al 0.2345
Cantidad en bolivares: 43 al 0.6
Cantidad en bolivares: 2 al 0.12
```

Manejadores de Formato más comunes

Función De Flujo	Función Miembro	Descripción
dec	flags(10)	Fija la base en 10
hex	flags(16)	Fija la base en 16
oct	flags(8)	Fija la base en 8
char setfill(char)	fill(char)	Fika el caracter de relleno
int setprecision(int	int precision(int)	Fija la presición de la salida
setw(int)	width(int)	Fija el ancho de un campo y retorna a normalidad luego

Ejemplo

```
  #include <iostream>
#include <iomanip>
using namespace std;
void fn()
{
  cout << hex
       << setfill('0')
       << setw(8)
       << 15
       << endl
       << dec
       << 20
       << endl;
}
```

salida:
0000000f
20

Redefinición de los operadores de flujo

Siempre que se quiera reutilizar una función de impresión para la salida estandar (iostream), cadenas de flujos (string streams) y archivos (fstream) es ideal utilizar las clases base ostream e istream.

Ejemplo

```
#include <iostream>
#include <iomanip>
using namespace std;

class A {
  int e1;
  float e2;
public:
  A() : e1(0), e2(0.0)
  {
    // vacia
  }
  void display(ostream & out)
  {
    out << "Primer Elemento: " << e1 << "Segundo Elemento: " << e2
        << endl;
  }
};

ostream & operator << (ostream & o, A & _a)
{
  _a.display(o);
  return o;
}
```

Notas

- Recuerde que una llamada sobre un operador de impresión tiene la siguiente forma:
  - (((cout << “cantidad”) << c) << “otra: “)
  - Los parentesis denotan llamadas diferente
- Recuerde siempre devolver una referencia a la cadena de salida ostream y no a una cadena derivada como cout, fstream, etc.

Recordatorios

- Los caracteres representables a trav'es de un char pertenecen al cojunto ASCII.
- Un conjunto de caracteres extendido es el Unicode. Proporciona una codificaci'on para cada caracter en cualquier lengua.