Input Streams
“Flujos de Entrada”

Agustín J. González
Versión original de Kip Irvine
ELO-326: Seminario II

Streams de Entrada

- istream y ostream son clases estándares de C++, y están definidas en el encabezado <iostream>.
- cin es el objeto istream predefinido.
- EL operador >> extrae caracteres y números desde un stream de entrada.
- Si el strean de entrada está vacío, al función >> espera por entrada (decimos que se bloquea)

Banderas o Flags de Estatus

- Los flags de estatus en un objeto istream indican el estado del stream. Se emplean las siguientes funciones:
  - cin.good() - stream de entrada está OK
  - cin.bad() - stream de entrada está corrupto
  - cin.fail() - operación más reciente falló

```
cin >> n;
if(cin.good())...      // stream is ok
if(cin.fail())...      // corrupted
```

Reiniciación de un stream de entrada

- La función clear() reinicia un stream corrupto a un estado normal.
- Ejercicio: Corra el siguiente ejemplo y digite varios caracteres aleatoriamente:

```
double salary;
cin >> salary;
if( cin.fail() )
{
  cout << ”Entrada invalida!\n";
  cin.clear();
}
```

Removiendo Entrada en exceso

- La función ignore() remueve toda la entrada de un objeto istream hasta un carácter definido como delimitador.
- El carácter delimitador por defecto es el EOF, pero se puede sustituir (por ejemplo por '\n' ,end of line). Ejemplo:

```
cin.ignore( 1024, '\n' );
```

Saltándo espacios en blanco

- Por defecto, el operador >> salta espacios en blanco (espacios, tab, end-of-line) en el stream de entrada. Supongamos que el stream de entrada contenga:

  "26<eoln>32<tab>45 22"

- La siguiente sentencia lee los números exitosamente los cuatro números:

```
cin >> A >> B >> C >> D;
```

Modificador de salto de espacios

- Si el modificador noskipws es usado, espacios en blanco subsecuentes son tratados como carateres de entrada..
- El modificador skipws retorna la operación al valor por defecto.

Leyendo carateres de entrada

- El operador >> lee caracteres individuales y automáticamente se salta espacios en blanco.
- Supongamos que el stream de entrada contiene:

  " X<\n>Y<\t>Z"

- La siguinete sentencia asignará 'X' a c1, 'Y' a c2, y 'Z' a c3:

```
char c1, c2, c3;
cin >> c1 >> c2 >> c3;
```

Leyendo todos los caracteres

- Si usamos noskipws, podemos leer caracteres espacios, tabs, y eoln.
- Supongamos que la entrada contiene:

```
" X<\n>Y<\t>Z"
```

- y se ejecuta la siguiente sentencia:

```
char c1,c2,c3,c4,c5;
cin >> c1 >> c2 >> c3 >> c4 >> c5;
```

- Los siguinetes valores son asignados:

```
// c1=' ',c2='X',c3='\n',c4='Y',
// c5='\t'
```

La función get()

- La función get() lee un único carácter de la entrada sin saltar espacios.
- Supongamos que la entrada contiene::

```
"X<\t>A B"
```

La siguientes sentencias leerán estos datos:

```
char c1,c2,c3,c4,c5;
cin.get( c1 );    // 'X'
cin.get( c2 );    // '\t'
cin.get( c3 );    // 'A'
cin.get( c4 );    // ' '
cin.get( c5 );    // 'B'
```

Limitando el stream de entrada

- Podemos usar el manipulador istream::setw(int n) para limitar el número de caracteres que serán leidos desde una entrada hacia un string.
- OJO hacer el argumento (n) una unidad mayor que el número de caracteres a leer. :

```
string temp;
cout << "Enter a string. The first 5 "
        "characters will be read: ";
cin >> setw(6) >> temp;
cout << "The string is: " << temp << endl;
cin.ignore(255,'\n');
```

Leyendo Valores Booleanos

- El manipulador boolalpha nos permite leer valores booleanos "true" o "false" desde la entrada. Ejemplo:

```
cin >> boolalpha;

bool IsChilean;
cout << "Is the student chilean "
        "[true/false]? ";
// (the user types true or false)
cin >> IsChilean;
```

Fin