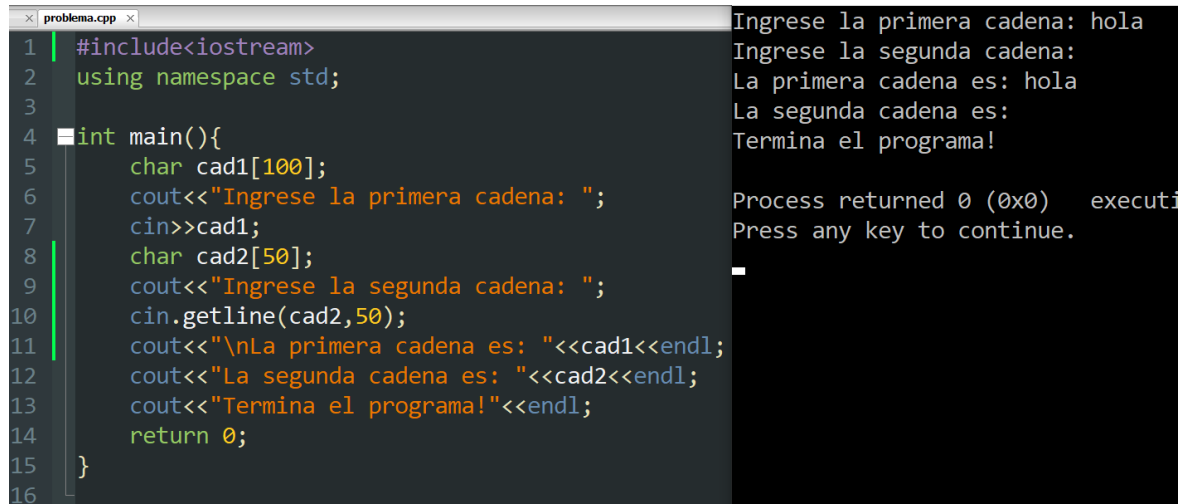


Universidad Nacional de Asunción - Facultad de Ingeniería

Cátedra de Computación

Descripción del problema – “Salto” de `cin.getline()`

Cuando trabajamos con cadenas, en ocasiones nos aparece el problema del "salto" de ciertas instrucciones de lecturas de cadenas. Esto ocurre cuando se utiliza el `cin.getline()` -el cual se "ignora"- luego de emplear `cin` (ya sea para cargar un número o una cadena). Un ejemplo de esta situación se visualiza aquí:



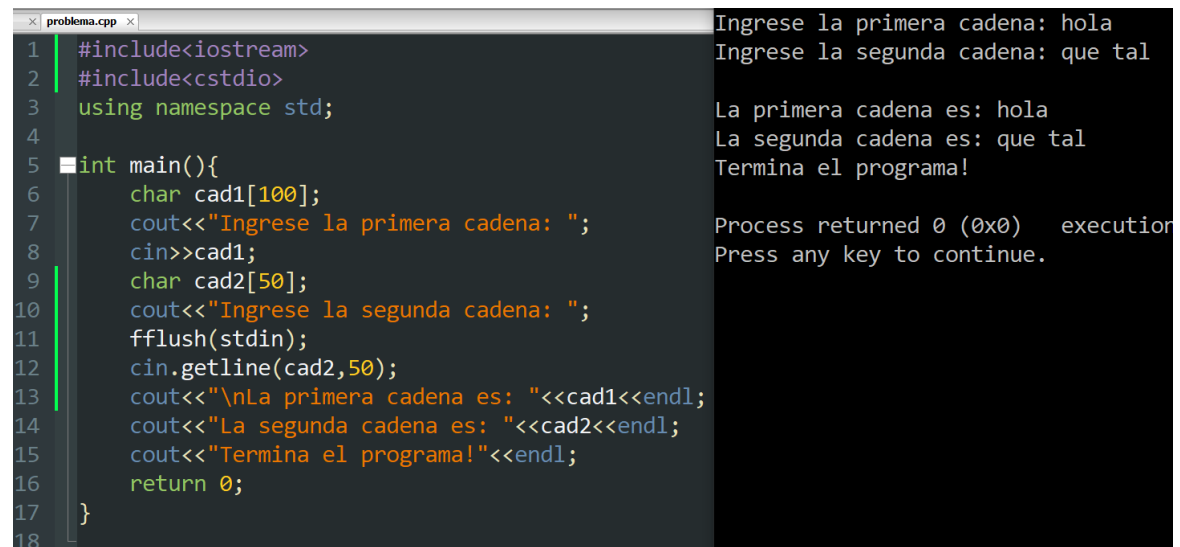
```
1 #include<iostream>
2 using namespace std;
3
4 int main(){
5     char cad1[100];
6     cout<<"Ingrese la primera cadena: ";
7     cin>>cad1;
8     char cad2[50];
9     cout<<"Ingrese la segunda cadena: ";
10    cin.getline(cad2,50);
11    cout<<"\nLa primera cadena es: "<<cad1<<endl;
12    cout<<"La segunda cadena es: "<<cad2<<endl;
13    cout<<"Termina el programa!"<<endl;
14    return 0;
15 }
16
```

Ingresa la primera cadena: hola
Ingresa la segunda cadena:
La primera cadena es: hola
La segunda cadena es:
Termina el programa!
Process returned 0 (0x0) execution terminated.
Press any key to continue.

Como podrá comprobarse, luego de ingresar la primera cadena, automáticamente muestra ambas cadenas en pantalla (en donde `cad2` está vacía) y finaliza el programa. En resumen, esto ocurre porque el `enter` que dimos al cargar “hola” quedó almacenado en el buffer, y es tomado por el `cin.getline()` para `cad2` (que resulta en una cadena vacía, ya que ese `enter` es el delimitador por defecto). Más detalles acerca de la aparición de esta situación pueden encontrarse en este enlace: <https://yosoy.dev/limpiar-el-buffer-en-c/>

Uso de `fflush()`

Generalmente damos solución a esto empleando la instrucción `fflush(stdin)` para limpiar el buffer de entrada, impidiendo que el `enter` se cargue en el `cin.getline()` siguiente. Esta función se encuentra en la librería `cstdio`.



```
1 #include<iostream>
2 #include<cstdio>
3 using namespace std;
4
5 int main(){
6     char cad1[100];
7     cout<<"Ingrese la primera cadena: ";
8     cin>>cad1;
9     char cad2[50];
10    cout<<"Ingrese la segunda cadena: ";
11    fflush(stdin);
12    cin.getline(cad2,50);
13    cout<<"\nLa primera cadena es: "<<cad1<<endl;
14    cout<<"La segunda cadena es: "<<cad2<<endl;
15    cout<<"Termina el programa!"<<endl;
16    return 0;
17 }
18
```

Ingresa la primera cadena: hola
Ingresa la segunda cadena: que tal
La primera cadena es: hola
La segunda cadena es: que tal
Termina el programa!
Process returned 0 (0x0) execution terminated.
Press any key to continue.

El problema está en que `fflush()` está definido para el buffer de salida (`stdout`), siendo su comportamiento indefinido para el buffer de entrada (`stdin`). Más detalles: <https://es.stackoverflow.com/questions/226494/para-que-sirve-flush-en-c>

Cuando programamos en nuestros equipos, generalmente no tenemos inconveniente al usar `fflush()` (la mayoría programa en Windows). Este no es el caso del VPL (que corre en Linux). De hecho, al emplearlo allí, `fflush()` no tiene el comportamiento esperado.

fflush.cpp

```
1 #include<iostream>
2 #include<cstdio>
3 using namespace std;
4
5 int main(){
6     char cad1[100];
7     cin>>cad1;
8     char cad2[50];
9     fflush(stdin);
10    cin.getline(cad2,50);
11    cout<<"La primera cadena es: "<<cad1<<endl;
12    cout<<"La segunda cadena es: "<<cad2<<endl;
13    cout<<"Termina el programa!"<<endl;
14    return 0;
15 }
```

Nota propuesta: 0 / 100

Comentarios

Test 1: Caso1
Incorrect program output
--- Input ---
hola
que tal

--- Program output ---
La primera cadena es: hola
La segunda cadena es:
Termina el programa!

--- Expected output (exact text)---
La primera cadena es: hola
La segunda cadena es: que tal
Termina el programa!

Summary of tests
+-----+
| 1 test run/ 0 tests passed |
+-----+

Solución mediante `cin.ignore()`

Para solucionar este inconveniente se propone el uso de la función `cin.ignore()`, la cual se encarga de ignorar el `enter` en el buffer de entrada. Más detalles sobre el empleo de esta función: <https://es.stackoverflow.com/questions/193829/porque-es-necesario-usar-el-comando-cin-ignore-despu%C3%A9s-de-usar-cin-y-lue>

fflush.cpp

```
1 #include<iostream>
2 #include<cstdio>
3 using namespace std;
4
5 int main(){
6     char cad1[100];
7     cin>>cad1;
8     char cad2[50];
9     //fflush(stdin);
10    cin.ignore(); /*alternativa a fflush() para solucionar
11    el problema descrito*/
12    cin.getline(cad2,50);
13    cout<<"La primera cadena es: "<<cad1<<endl;
14    cout<<"La segunda cadena es: "<<cad2<<endl;
15    cout<<"Termina el programa!"<<endl;
16    return 0;
17 }
```

Nota propuesta: 100 / 100

Comentarios

Summary of tests
+-----+
| 1 test run/ 1 test passed |
+-----+