

Primer p1.cpp

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

using namespace std;

const char* intToString( int n )
{
    stringstream ss {};
    ss << n;
    return ss.str().c_str();
}

int main()
{
    cout << 5 << '\n';
    cout << intToString( 5 ) << '\n';
    cout << intToString( 6 ) << '\n';
    cout << 6 << '\n';
    return 0;
}
```

Korak 1

Program ne radi, ne ispisuje konvertovane brojeve:

```

1  #include <iostream>
2  #include <sstream>
3
4  using namespace std;
5
6  const char* intToString( int n )
7  {
8      stringstream ss {};
9      ss << n;
10     return ss.str().c_str();
11 }
12
13 int main()
14 {
15     cout << 5 << '\n';
16     cout << intToString( 5 ) << '\n';
17     cout << intToString( 6 ) << '\n';
18     cout << 6 << '\n';
19     return 0;
20 }
21

```

Terminal output:

```

5
6
==>

```

Ako se ide korak po korak vidi se da je broj zapisan u privremeni tok, ali da se ne vraća.

```

1  #include <iostream>
2  #include <sstream>
3
4  using namespace std;
5
6  const char* intToString( int n )
7  {
8      stringstream ss {};
9      ss << n;
10     return ss.str().c_str();
11 }
12
13 int main()
14 {
15     cout << 5 << '\n';
16     cout << intToString( 5 ) << '\n';
17     cout << intToString( 6 ) << '\n';
18     cout << 6 << '\n';
19     return 0;
20 }
21

```

Debugger Variables:

- Locals:
 - std::basic_ostream<char, std::char_traits<char>>: n: 5, ss: 0x00ec5278 "5"
- Registers: (empty)

CALL STACK:

- Paused on step
- p1.exe!intToString(int n) Line 10 p1...
- p1.exe!main() Line 16 p1.cpp 16:1
- [Inline Frame] p1.exe!invoke_main() Line 7
- p1.exe!__scrt_common_main_seh() Line 288
- kernel32.dll!76fc9c9() Unknown Source
- [Frames below may be incorrect and/or miss]
- ntdll.dll!177b57c6e() Unknown Source 0
- ntdll.dll!177b57c3e() Unknown Source 0

Terminal output:

```

5

```

Pokušamo da napravimo lokalnu promenljivu, da vidimo da li taj deo radi dobro:

```
string s = ss.str();
```

Proverimo njenu vrednost i ustanovimo da dobija ispravnu vrednost.

Pokušamo da dobijemo iz nje odgovarajuću vrednost:

```
string s = ss.str();
const char* cs = s.c_str();
```

Proverimo kako se ona ponaša i ustanovimo da i `cs` dobija ispravnu vrednost.

Pokušamo da vratimo tu vrednost:

```
string s = ss.str();
return s.c_str();
```

Korak 2

I dalje se ne vraća ispravna vrednost.

The screenshot shows a Visual Studio Code editor with a C++ file named p1.cpp. The code is as follows:

```

1 #include <iostream>
2 #include <sstream>
3
4 using namespace std;
5
6 const char* intToString( int n )
7 {
8     stringstream ss {};
9     ss << n;
10    string s = ss.str();
11    return s.c_str();
12 }
13
14 int main()
15 {
16     cout << 5 << '\n';
17     cout << intToString( 5 ) << '\n';
18     cout << intToString( 6 ) << '\n';
19     cout << 6 << '\n';
20     return 0;
21 }
22

```

The terminal output shows:

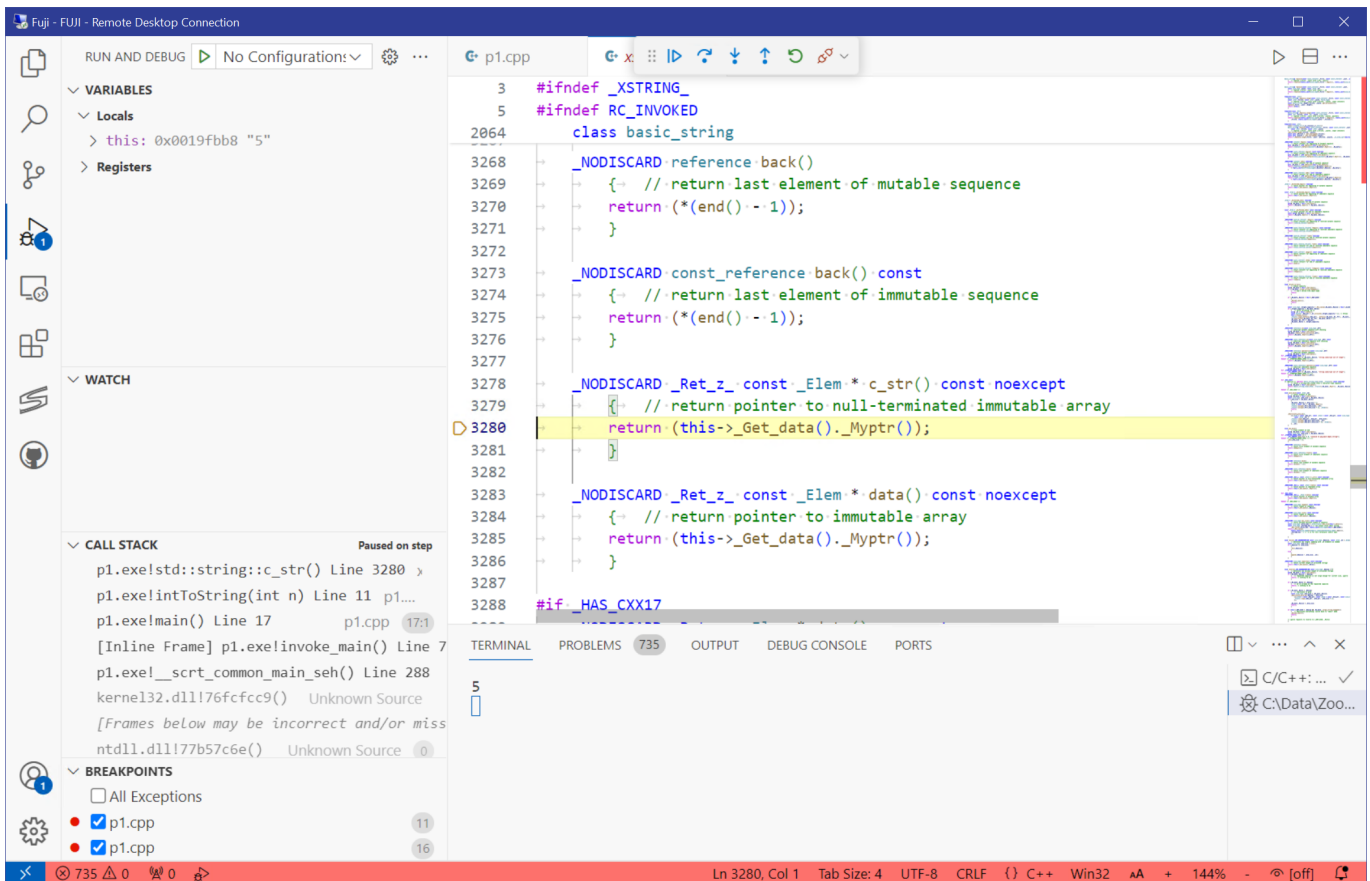
```

5
6
==>

```

The status bar at the bottom indicates the cursor is at line 22, column 1, with 4 spaces, UTF-8 encoding, CRLF line endings, C++ language, Win32 architecture, and 144% zoom.

Proverimo šta radi `c_str()`: vraća interni sadržaj objekta. Znači, zapravo smo imali sreće - program je mogao i da otkáže, zato što vraćamo pokazivač na sadržaj koji se u međuvremenu obriše.



Možemo da probamo da napravimo trajni objekat, koji će preživeti povratak iz funkcije:

```
static string s = ss.str();
```

Sada se program ponaša drugačije, ali ponovo ne radi. Oba puta vraća 5, umesto 6.

Izvršavamo korak po korak da bismo videli u čemu je problem.

Korak 3

Problem je u tome što se statički objekat inicijalizuje samo pri prvoj upotrebi. Znači, potrebno je da tu jednu naredbu podelimo na dve:

```
static string s;  
s = ss.str();
```

Sada program radi.

Ali ovakvo rešenje je veoma loše, zato što se vraćaju podaci koji nisu nezavisni od konteksta upotrebe. Da bismo to pokazali, hajde da promenimo glavni program i da uradimo ovako nešto:

```
auto n5 = intToString( 5 );  
auto n6 = intToString( 6 );
```

```
cout << n5 << '\n';  
cout << n6 << '\n';
```

Sada se oba puta ispisuje 6. I opet imamo sreće, jer da je operacija napravila neku značajno dužu nisku, onda bi nova niska bila na drugoj adresi i veliko je pitanje šta bismo dobili.

Za nauk: Nikada ne vraćati sadržaj privremenog objekta. Nikada ne vraćati ni sadržaj statičkog objekta, osim ako je apsolutno sigurno da se se on neće menjati.