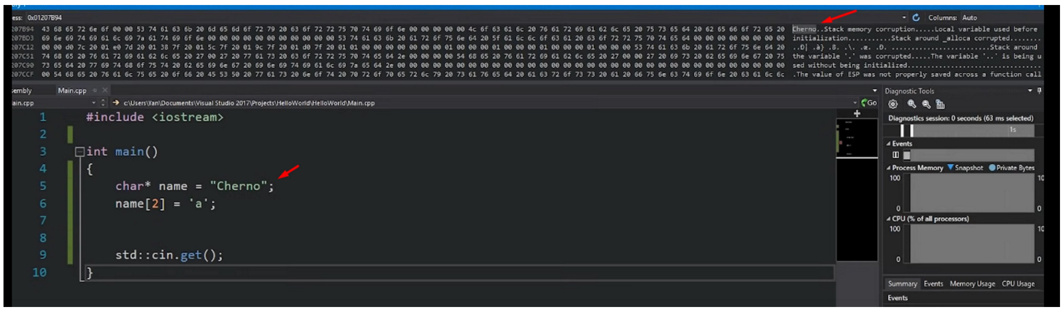
31 - How Strings Work in C++

quarta-feira, 19 de fevereiro de 2025 21:28

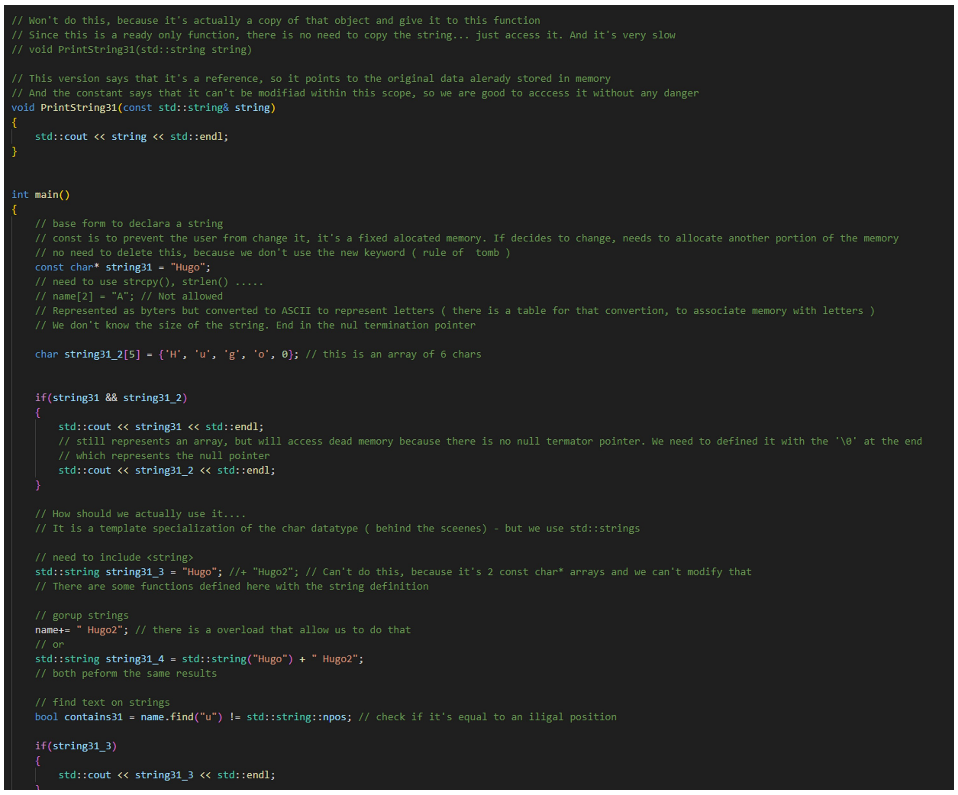
- Very much tied with arrays and pointers
- What is a string in general?
  - group of characters
  - represent text n a certain shape of format
  - could be a single char ou a paragraph
  - Way to reresent and manipulate the text
- First need to understand how char works
  - There is a data type called char, 1 byte of memory
  - Usefull for alocate buffers
  - very usefull for text
  - UTF1 -> 16 BITS to store.....
    - But c++ the base is char 1 byte (limted)
    - basicl english
- string is just na arry of chars
  - refers to const char* pointers



```
ess: 0x01207B94                                                                                               Columns: Auto
207B94  43 68 65 72 6e 6f 00 00 53 74 61 63 6b 20 6d 65 6d 6f 72 79 20 63 6f 72 72 75 70 74 69 6f 6e 00 00 00 00 00 4c 6f 63 61 6c 20 76 61 72 69 61 62 6c 65 20 75 73 65 64 20 62 65 66 6f 72 65 20  Cherno..Stack memory corruption.....Local variable used before
207BD3  69 6e 69 74 69 61 6c 69 7a 61 74 69 6f 6e 00 00 00 00 00 00 00 00 00 00 53 74 61 63 6b 20 61 72 6f 75 6e 64 20 5f 61 6c 6c 6f 63 61 20 63 6f 72 72 75 70 74 65 64 00 00 00 00 00 00 00  initialization..........Stack around _alloca corrupted.......
207C12  00 00 d0 7c 20 01 e0 7d 20 01 38 7f 20 01 5c 7f 20 01 9c 7f 20 01 d0 7f 20 01 01 00 00 00 00 00 00 01 00 00 00 01 00 00 00 01 00 00 00 01 00 00 00 53 74 61 63 6b 20 61 72 6f 75 6e 64 20  ..Ð| .à} .8. .\. .œ. .Ð. ........................Stack around
207C51  74 68 65 20 76 61 72 69 61 62 6c 65 20 27 00 27 20 77 61 73 20 63 6f 72 72 75 70 74 65 64 2e 00 00 00 00 54 68 65 20 76 61 72 69 61 62 6c 65 20 27 00 00 27 20 69 73 20 62 65 69 6e 67 20 75  the variable '.' was corrupted.....The variable '..' is being u
207C90  73 65 64 20 77 69 74 68 6f 75 74 20 62 65 69 6e 67 20 69 6e 69 74 69 61 6c 69 7a 65 64 2e 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  sed without being initialized...............................
207CCF  00 54 68 65 20 76 61 6c 75 65 20 6f 66 20 45 53 50 20 77 61 73 20 6e 6f 74 20 70 72 6f 70 65 72 6c 79 20 73 61 76 65 64 20 61 63 72 6f 73 73 20 61 20 66 75 6e 63 74 69 6f 6e 20 63 61 6c 6c  .The value of ESP was not properly saved across a function call
```

```cpp
#include <iostream>

int main()
{
    char* name = "Cherno";
    name[2] = 'a';


    std::cin.get();
}
```

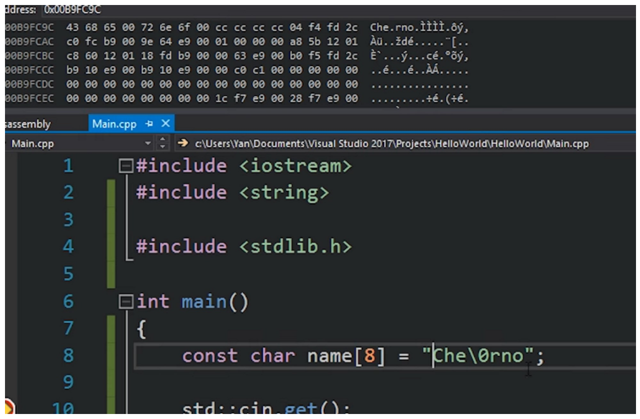
sembly
Main.cpp
ain.cpp
c:\Users\Yan\Documents\Visual Studio 2017\Projects\HelloWorld\HelloWorld\Main.cpp
Go
Diagnostic Tools
Diagnostics session: 0 seconds (63 ms selected)
1s
Events
Process Memory
Snapshot
Private Bytes
100
0
CPU (% of all processors)
100
0
Summary
Events
Memory Usage
CPU Usage
Events



```cpp
// Won't do this, because it's actually a copy of that object and give it to this function
// Since this is a ready only function, there is no need to copy the string... just access it. And it's very slow
// void PrintString31(std::string string)

// This version says that it's a reference, so it points to the original data alerady stored in memory
// And the constant says that it can't be modifiad within this scope, so we are good to acccess it without any danger
void PrintString31(const std::string& string)
{
    std::cout << string << std::endl;
}


int main()
{
    // base form to declara a string
    // const is to prevent the user from change it, it's a fixed alocated memory. If decides to change, needs to allocate another portion of the memory
    // no need to delete this, because we don't use the new keyword ( rule of  tomb )
    const char* string31 = "Hugo";
    // need to use strcpy(), strlen() .....
    // name[2] = "A"; // Not allowed
    // Represented as byters but converted to ASCII to represent letters ( there is a table for that convertion, to associate memory with letters )
    // We don't know the size of the string. End in the nul termination pointer

    char string31_2[5] = {'H', 'u', 'g', 'o', 0}; // this is an array of 6 chars


    if(string31 && string31_2)
    {
        std::cout << string31 << std::endl;
        // still represents an array, but will access dead memory because there is no null termator pointer. We need to defined it with the '\0' at the end
        // which represents the null pointer
        std::cout << string31_2 << std::endl;
    }

    // How should we actually use it....
    // It is a template specialization of the char datatype ( behind the sceenes) - but we use std::strings

    // need to include <string>
    std::string string31_3 = "Hugo"; //+ "Hugo2"; // Can't do this, because it's 2 const char* arrays and we can't modify that
    // There are some functions defined here with the string definition

    // gorup strings
    name+= " Hugo2"; // there is a overload that allow us to do that
    // or
    std::string string31_4 = std::string("Hugo") + " Hugo2";
    // both peform the same results

    // find text on strings
    bool contains31 = name.find("u") != std::string::npos; // check if it's equal to an iligal position

    if(string31_3)
    {
        std::cout << string31_3 << std::endl;
```

String Literals in C++

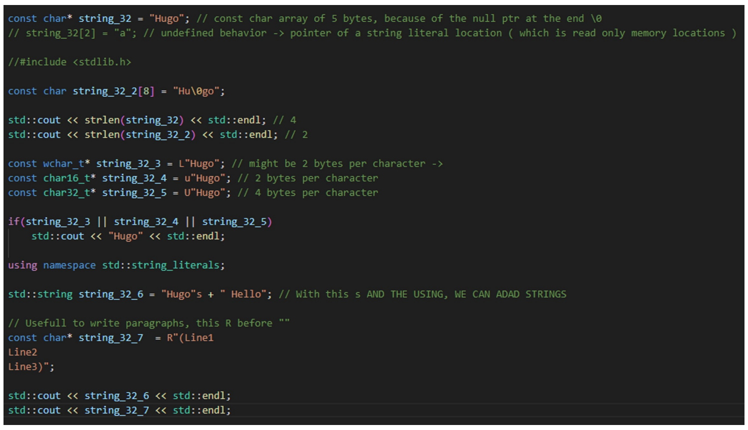
- series of characters between " "
- Always stored in const read memory, not possible to change it later on



```
ddress: 0x00B9FC9C
00B9FC9C  43 68 65 00 72 6e 6f 00 cc cc cc cc 04 f4 fd 2c  Che.rno.ÌÌÌÌ.ôý,
00B9FCAC  c0 fc b9 00 9e 64 e9 00 01 00 00 00 a8 5b 12 01  Àü..ždé.....¨[..
00B9FCBC  c8 60 12 01 18 fd b9 00 00 63 e9 00 b0 f5 fd 2c  È`...ý...cé.°õý,
00B9FCCC  b9 10 e9 00 b9 10 e9 00 00 c0 c1 00 00 00 00 00  ..é...é..ÀÁ.....
00B9FCDC  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00B9FCEC  00 00 00 00 00 00 00 00 1c f7 e9 00 28 f7 e9 00  ........÷é.(÷é.
```

```cpp
#include <iostream>
#include <string>

#include <stdlib.h>

int main()
{
    const char name[8] = "Che\0rno";

    std::cin.get();
```

sassembly
Main.cpp
Main.cpp
c:\Users\Yan\Documents\Visual Studio 2017\Projects\HelloWorld\HelloWorld\Main.cpp



```cpp
const char* string_32 = "Hugo"; // const char array of 5 bytes, because of the null ptr at the end \0
// string_32[2] = "a"; // undefined behavior -> pointer of a string literal location ( which is read only memory locations )

//#include <stdlib.h>

const char string_32_2[8] = "Hu\0go";

std::cout << strlen(string_32) << std::endl; // 4
std::cout << strlen(string_32_2) << std::endl; // 2

const wchar_t* string_32_3 = L"Hugo"; // might be 2 bytes per character ->
const char16_t* string_32_4 = u"Hugo"; // 2 bytes per character
const char32_t* string_32_5 = U"Hugo"; // 4 bytes per character

if(string_32_3 || string_32_4 || string_32_5)
    std::cout << "Hugo" << std::endl;

using namespace std::string_literals;

std::string string_32_6 = "Hugo"s + " Hello"; // With this s AND THE USING, WE CAN ADAD STRINGS

// Usefull to write paragraphs, this R before ""
const char* string_32_7  = R"(Line1
Line2
Line3)";

std::cout << string_32_6 << std::endl;
std::cout << string_32_7 << std::endl;
```