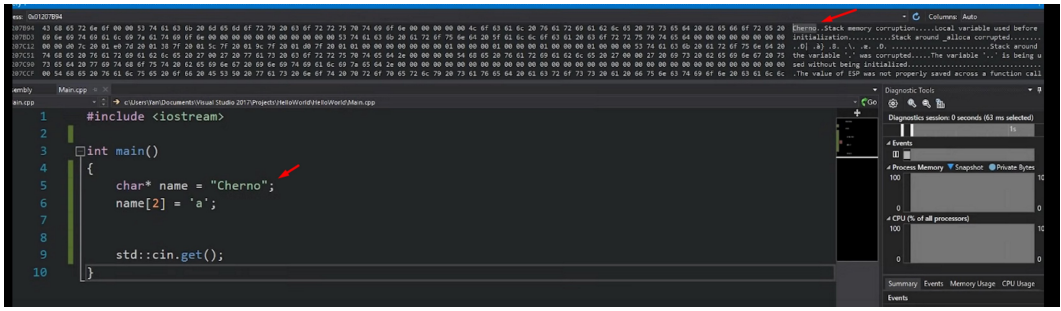
31 - How Strings Work in C++

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




```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;
    }
```