16 - POINTERS in C++

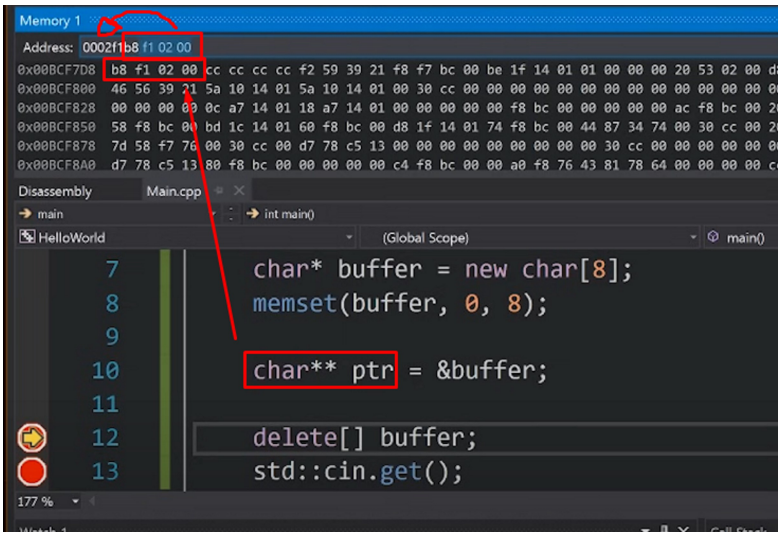
quarta-feira, 5 de fevereiro de 2025 07:57

- All application is stored in memory
- pointer is na integer that stores a memory adress
- Memory is one big line of memory, like a single streat n a town
  - every house has na adress (byte of data)
  - tell where the specific byte is
- don't need to use but it's usefull
- pointer is just na adress
- Types are someking of fixture to make our life easier, it's meaninless for pointers
- void* pointer -> don't need a type, just say that the data in that adress is suppose to be from that type
  - A type don't change what a ointer s
  - we can change the type of the pointer and it's ok, will store na integer with the adress anyway
  - Used for maniulation of that memory but not to store pointers (try to set a variable there for example)
- pointer = 0
  - 0 is not a valid memory adress
  - means null
  - NULL or nullptr is the same
- Every memory we create has a memory adress
  - The & gets the memory adress from a declared value
  - stores the integer adress into the pointer definition
  - with this memory value, we can search it in the memory data (used in visual studio)
- why not always use void *
  - dereference to acess the value on a certain pointer
  - get a value out of a pointer, what is in there?

```
void* ptr = 0;
   int var = 8;
   ptr = &var;
```

[POINTERS in C++]


```
// allocates 8 bytes of memory and return the pointer for the first one
char* buffer = new char[8];
// the a block of memory with a value.. Fill Buffer with zeros for 8 bytes
memset(buffer, 0, 8)
// need to delete the allocated memory
delete[] buffer;
```



- Pointers are just variables, that is why we can have a ointer of a pointer
  - a memory adress that pointes to a nother memory adress
  - stores 4 bytes with the adress of the original pointer (backwards)
  - the original pointer shows the actual data

- References
  - almos the same thing as pointers from what the computer will do, but its different on how we write
  - reference is a way to reference na existing reference
  - not new variables and don't have storages, just a reference to a variable
  - the & is parte of the type, and not in the value (int&)
    - set it equal to na existing variable
  - we create something called alias, just a reference. t just exist in our source code
  - in this case, ref17 is not a variable, just a17
  - there is no need for the compiler to create a new variable

```
int a17 = 5;
int *b17 = &a;
int& ref17 = a;
```

```
// void Increment(int value) // copy the value to a new one
void Increment(int& value) // pass the variable by value and we affect the one received as paramenter
```

```
// void Increment(int value) // copy the value to a new one
void Increment(int* value) // pass the variable by value and we affect the one received as paramenter
{
    (*value)++; // want to increment the value so need to dereference
    // otherwise it would increase the pointer
}
void Increment2(int& value) // pass the variable by value and we affect the one received as paramenter
{
    value++; // samething as before but easier
}
```