

Alocação Dinâmica de Memória

ELC1067 - Laboratório de Programação II

João Vicente Ferreira Lima (UFSM)

Departamento de Linguagens e Sistemas de Computação
Universidade Federal de Santa Maria

`jvlima@inf.ufsm.br`

`http://www.inf.ufsm.br/~jvlima`

2023/1

Outline

- 1 Alocação de memória
- 2 Ponteiros
- 3 Matrizes
- 4 Valgrind

Outline

- 1 Alocação de memória
- 2 Ponteiros
- 3 Matrizes
- 4 Valgrind

Layout de memória

The image shows a screenshot of the Visual Studio Code editor with two panels. The left panel displays a C++ source file named 'C++ source #1'. The code defines a static character array `v2` of size 10 and a function `foo` that initializes `v1` and `v2` with specific characters. The right panel shows the assembly output for the `foo` function, generated by x86-64 gcc 12.2. The assembly code shows the function prologue, pushing the base pointer, and then moving values into the `v1` and `v2` arrays using `mov` instructions with immediate offsets and register pointers. The function ends with a `ret` instruction.

```
1 static char v2[10];
2
3
4 void foo(void) {
5     char v1[10];
6
7     v1[0] = 'a';
8     v1[9] = 'b';
9     v2[0] = 'c';
10    v2[9] = 'd';
11 }
```

```
1 foo():
2     push    rbp
3     mov     rbp, rsp
4     mov     BYTE PTR [rbp-10], 97
5     mov     BYTE PTR [rbp-1], 98
6     mov     BYTE PTR v2[rip], 99
7     mov     BYTE PTR v2[rip+9], 100
8
9     nop
10    pop     rbp
11    ret
```

<https://godbolt.org/z/envYnj8E9>

Alocação de memória

C

```
1 int *p = NULL;
2 p = (int*) malloc( sizeof(int)*10 );
3
4 *p = 1;
5 p[1] = 2;
6 *(p+2) = 3;
7
8 free( (void*)p );
```

C++

```
1 int *p = nullptr;
2 p = new int[10]
3
4 *p = 1;
5 p[1] = 2;
6 *(p+2) = 3;
7
8 delete[] p;
```

C++ não tem realloc

Alocação de memória

C

```
1 int *p = NULL;
2 p = (int*) malloc( sizeof(int)*10 );
3
4 *p = 1;
5 p[1] = 2;
6 *(p+2) = 3;
7
8 free( (void*)p );
```

C++

```
1 int *p = nullptr;
2 p = new int[10]
3
4 *p = 1;
5 p[1] = 2;
6 *(p+2) = 3;
7
8 delete[] p;
```

C++ não tem realloc

Outline

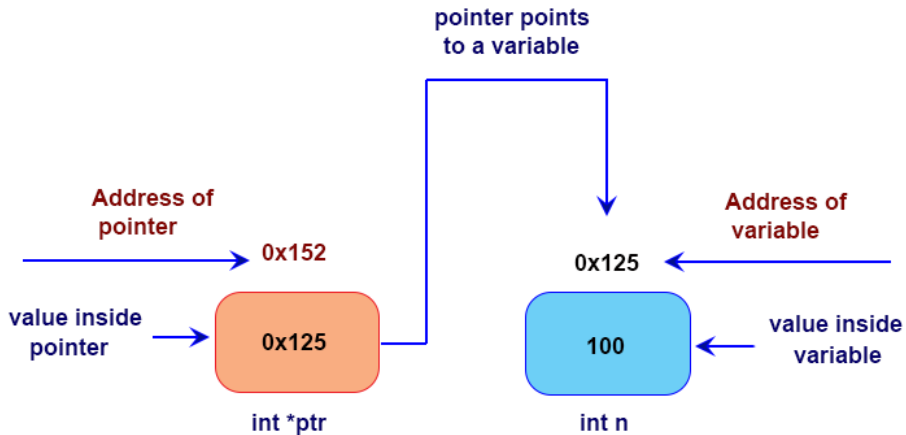
1 Alocação de memória

2 Ponteiros

3 Matrizes

4 Valgrind

Ponteiros



<https://www.w3resource.com/c-programming/c-pointer.php>

C - Pointers

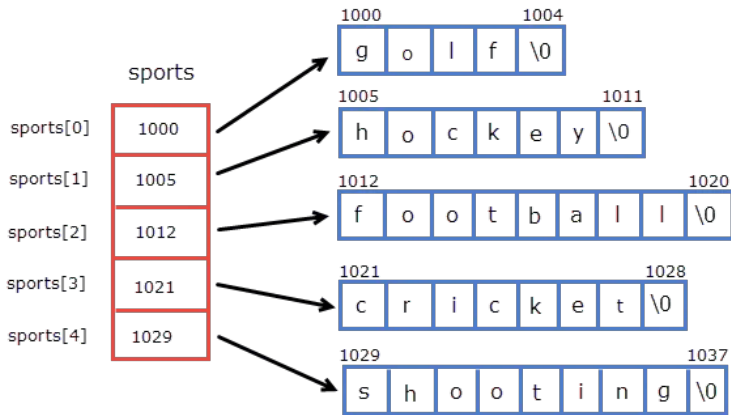
```
int var = 10;  
int *p;  
p = &var;
```



P is a pointer that stores the address of variable var.
The data type of pointer p and variable var should match because an integer pointer can only hold the address of integer variable.

<https://beginnersbook.com/2014/01/c-pointers/>

Matriz de caracteres



Memory representation of array of pointers

Alocação de memória

New e delete

```
int* ptr1 = nullptr; // nullptr eh NULL em C++
int* ptr2 {nullptr}; // inicializador padrao

auto num = new int; // aloca inteiro
*num = 33;
delete num;          // libera memoria

auto vetor = new int[10]; // vetor de 10 numeros
delete[] vetor;          // libera um vetor

int* vetor {new int[10]}; // outra forma
delete[] vetor;
```

Outline

- 1 Alocação de memória
- 2 Ponteiros
- 3 Matrizes**
- 4 Valgrind

Alocação de memória

```
int** matriz {nullptr};  
int N = 10;  
matriz = new int*[N]; // vetor com ponteiros  
for(auto i = 0; i < N; i++)  
    matriz[i] = new int[N]; // linha da matriz  
  
// inicia valores  
for(auto i = 0; i < N; i++)  
    for(auto j = 0; j < N; j++)  
        matriz[i][j] = 0;  
  
// destroi matriz: cada linha e depois matriz  
for(auto i = 0; i < N; i++)  
    delete[] matriz[i];  
delete[] matriz;
```

Outline

- 1 Alocação de memória
- 2 Ponteiros
- 3 Matrizes
- 4 Valgrind**

Valgrind

Valgrind é uma ferramenta de depuração e instrumentação de programas para análise de memória.

Exemplo

```
int main(void)
{
    auto vetor = new int[100];
    for(auto i= 0; i < 100; i++)
        vetor[i] = 1;
    delete[] vetor;
    return 0;
}
```

Execução

```
$ valgrind --leak-check=full ./exemplo
```


Saída (correto)

HEAP SUMMARY:

```
    in use at exit: 72,704 bytes in 1 blocks
    total heap usage: 2 allocs, 1 frees, 73,104 bytes allocated
```

LEAK SUMMARY:

```
    definitely lost: 0 bytes in 0 blocks
    indirectly lost: 0 bytes in 0 blocks
    possibly lost: 0 bytes in 0 blocks
    still reachable: 72,704 bytes in 1 blocks
    suppressed: 0 bytes in 0 blocks
```

```
ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Exemplo com erro 1

```
int main(void)
{
    auto vetor = new int[100];

    for(auto i= 0; i < 100; i++)
        vetor[i] = 1;

    delete vetor;

    return 0;
}
```

Valgrind

Saída (erro 1)

```
Mismatched free() / delete / delete []
    at 0x4C2C2BC: operator delete(void*) (in ...)
    by 0x4007B4: main (in /home/jvlima/alloc2)
Address 0x5a37c80 is 0 bytes inside a block of size 400 alloc'd
    at 0x4C2B800: operator new[](unsigned long) (in ...)
    by 0x400777: main (in /home/jvlima/alloc2)
```

HEAP SUMMARY:

```
    in use at exit: 72,704 bytes in 1 blocks
total heap usage: 2 allocs, 1 frees, 73,104 bytes allocated
```

LEAK SUMMARY:

```
    definitely lost: 0 bytes in 0 blocks
    indirectly lost: 0 bytes in 0 blocks
    possibly lost: 0 bytes in 0 blocks
    still reachable: 72,704 bytes in 1 blocks
```

Exemplo com erro 2

```
int main(void)
{
    auto vetor = new int[100];

    for(auto i= 0; i < 100; i++)
        vetor[i] = 1;

    return 0;
}
```

Saída (erro 2)

HEAP SUMMARY:

```
in use at exit: 73,104 bytes in 2 blocks
total heap usage: 2 allocs, 0 frees, 73,104 bytes allocated
```

LEAK SUMMARY:

```
definitely lost: 400 bytes in 1 blocks
indirectly lost: 0 bytes in 0 blocks
possibly lost: 0 bytes in 0 blocks
still reachable: 72,704 bytes in 1 blocks
suppressed: 0 bytes in 0 blocks
```

Exemplo com erro 3

```
int main(void)
{
    auto vetor = new int[100];

    for(auto i= 0; i < 101; i++)
        vetor[i] = 1;

    return 0;
}
```

Causa segmentation fault?

Exemplo com erro 3

```
int main(void)
{
    auto vetor = new int[100];

    for(auto i= 0; i < 101; i++)
        vetor[i] = 1;

    return 0;
}
```

Causa segmentation fault?

Saída (erro 3)

```
==1651898== Invalid write of size 4
==1651898== at 0x109184: main (erro.cpp:5)
==1651898== Address 0x4dbfe10 is 0 bytes after a block of size
    400 alloc'd
==1651898== at 0x483C583: operator new[](unsigned long) (in /usr/
    lib/x86_64-linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.
    so)
==1651898== by 0x10915E: main (erro.cpp:3)
```

Causa segmentation fault? Não!!!

<https://joao-ufsm.github.io/l22023a/>

