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

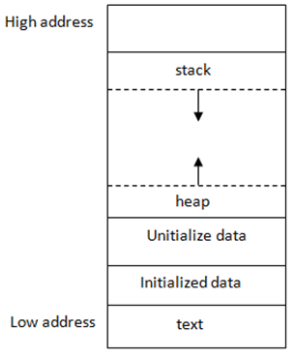
2023/1

Outline

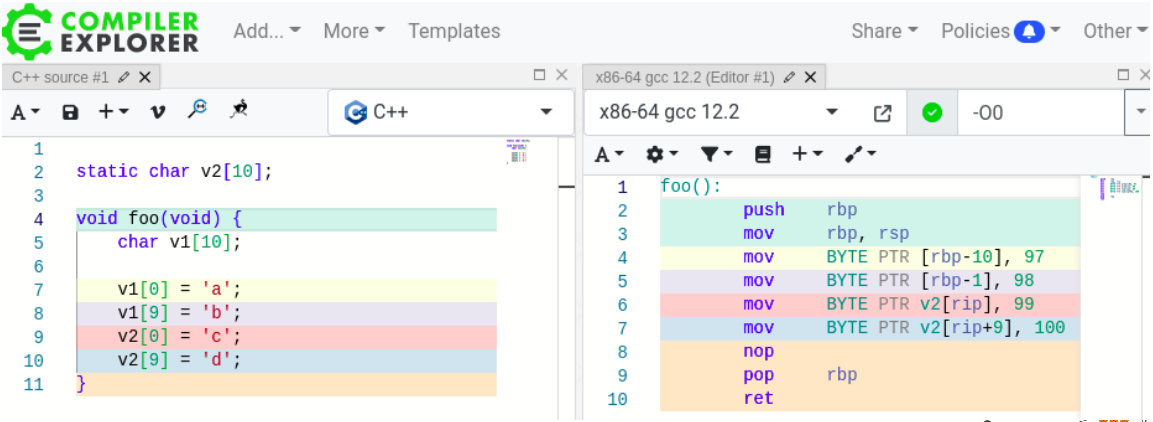
Outline

Layout de memória

- A alocação de memória é sempre contígua, tanto estático quanto dinâmico.
- `text` é o código da aplicação.
- A **pilha** (*stack*) cresce do fim para o começo.
  - Armazena variáveis locais
- O **heap** cresce do começo ao fim.
  - Memória alocada dinamicamente



Layout de memória



https://godbolt.org/z/envYnj8E9

Alocação de memória

C

```c
int *p = NULL;
p = (int*) malloc( sizeof(int)*10 );

*p = 1;
p[1] = 2;
*(p+2) = 3;

free( (void*)p );
```

C++

```cpp
int *p = nullptr;
p = new int[10]

*p = 1;
p[1] = 2;
*(p+2) = 3;

delete[] p;
```

C++ não tem realloc

Alocação de memória

C

```c
int *p = NULL;
p = (int*) malloc( sizeof(int)*10 );

*p = 1;
p[1] = 2;
*(p+2) = 3;

free( (void*)p );
```

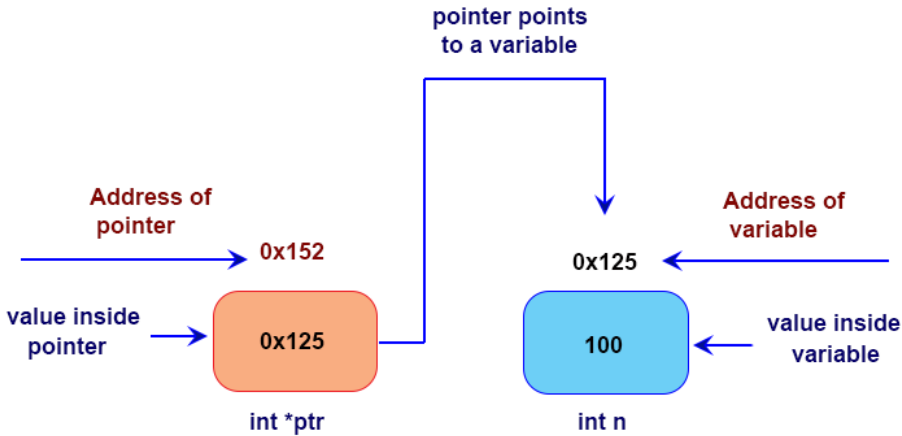
C++

```cpp
int *p = nullptr;
p = new int[10]

*p = 1;
p[1] = 2;
*(p+2) = 3;

delete[] p;
```

C++ não tem realloc

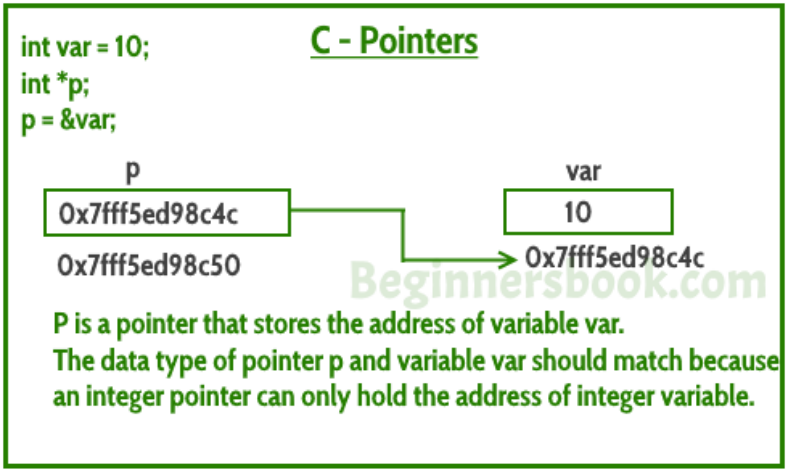
Outline

Ponteiros



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

Ponteiros

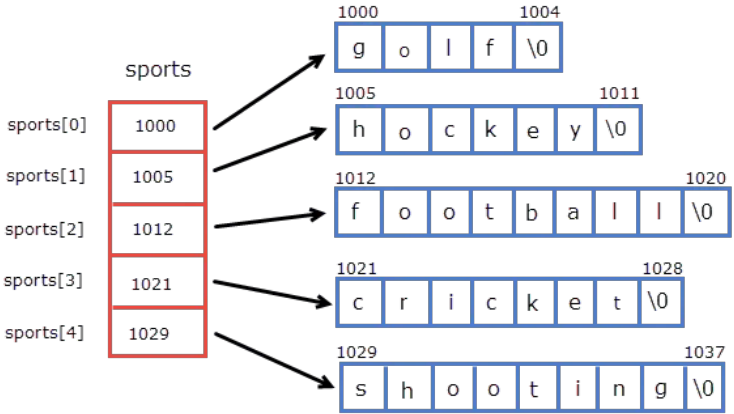


https://beginnersbook.com/2014/01/c-pointers/

informática
Universidade Federal de Santa Maria

Universidade Federal de Santa Maria
1960

Matriz de caracteres



Memory representation of array of pointers

TheCguru.com

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

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

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
```

Valgrind

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)
```

Valgrind

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;
}
```

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
```

Valgrind

Exemplo com erro 2

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

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

   return 0;
}
```

Valgrind

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
```

Valgrind

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?

Valgrind

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!!!**

informática
Universidade Federal de Santa Maria

Universidade Federal de Santa Maria
1960

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