PROGRAMAÇÃO O.O.

(*C#*)

< Conceitos sobre a Memória

Professor: João Luiz Lagôas

>

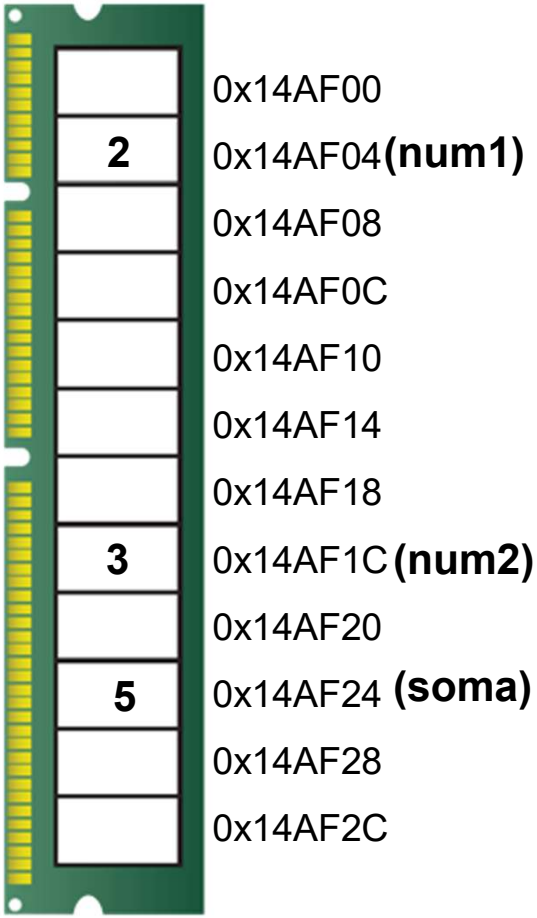
Conceitos sobre Memória

Colégio Pedro II

- Nomes de variáveis como num1, num2 e soma, na verdade correspondem a **localizações** na memória do computador.

- Toda variável tem um **nome**, um **tipo**, um **tamanho** (em bits/bytes) e um **valor**.

```
int num1 = 2;
int num2 = 3;

int soma = num1 + num2;
```



Nome de variáveis

Identificadores

Colégio Pedro II

- Quais nomes minhas variáveis podem assumir?

- Devem começar com:
  - Letra;
  - Cifrão ($)
  - Caracter de conexão (_)
  - Não podem ser **palavras reservadas (*keywords*)**

Palavras reservadas (*keywords*)

- Aqui está uma lista de palavras reservadas no C#. Você não pode utilizar nenhuma delas como nome de uma variável.

abstract	as	base	bool
break	byte	case	catch
char	checked	class	const
continue	decimal	default	delegate
do	double	else	enum
event	explicit	extern	false
finally	fixed	float	for
foreach	goto	if	implicit
in	int	interface	internal
is	lock	long	namespace
new	null	object	operator
out	override	params	private
protected	public	readonly	ref
return	sbyte	sealed	short
sizeof	stackalloc	static	string
struct	switch	this	throw
true	try	typeof	uint
ulong	unchecked	unsafe	ushort
using	using static	virtual	void
volatile	while		

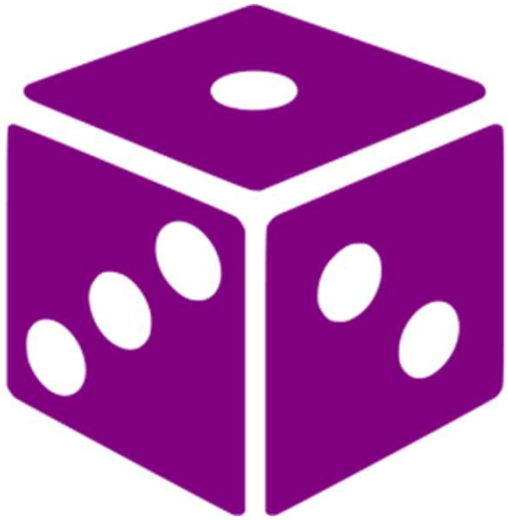
Tipos de dados

Colégio Pedro II

- O C# possui dois tipos de dados: **tipos primitivos** (por valor) e **tipos de referência** (por referência).

- Os tipos primitivos são **bool, char, byte, sbyte, short, ushort, int** e **uint, long, ulong, decimal, double, float**.

- Os tipos por referência, são classes que especificam os tipos de objeto. Exemplos: **array** e **string**



Em resumo, se uma variável não é primitiva, então ela necessariamente é uma referência.

Tipos de dados

Colégio Pedro II

Primitivos

int

bool

ulong

uint

long

decimal

double

sbyte

ushort

byte

float

char

short

Button

Math

String

Label

List<int>

float[]

Heroi

Form

FuncaoLinear

MovimentoUniforme

Relogio

int[]

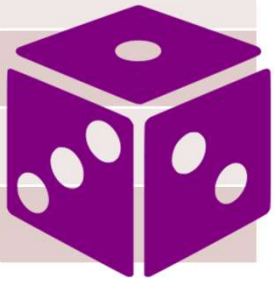
Aluno

Referência

Tipos Primitivos

Colégio Pedro II

Tipo	Bits	Valores
bool	1	true ou false
char	16	‘\u0000’ a ‘\uFFFF’
byte	8	0 a 255
sbyte	8	-128 a 127
short	16	-32.768 a 32.767
ushort	16	0 a 65.535
int	32	-2.147.483.648 a 2.147.483.647
uint	32	0 a 4.292.967.295
long	64	-9.223.372.036.854.775.808 a 9.223.372.036.854.775.807
ulong	64	0 a 18.446.744.073.709.551.615
decimal	128	1x10^-28 a 7,9x10^28
float	32	
double	64	

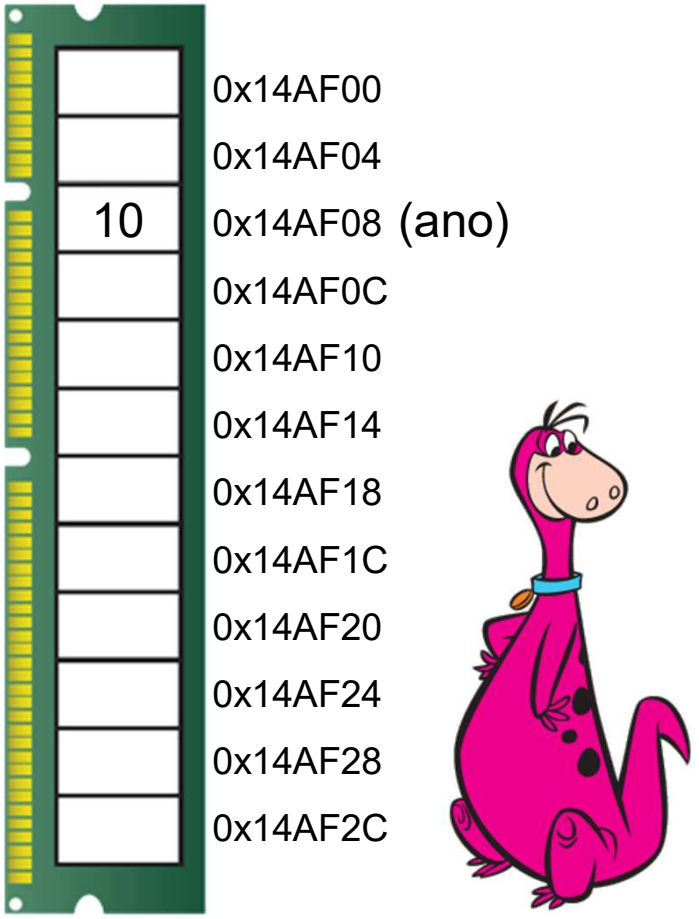


Tipos Primitivos

Colégio Pedro II

- Variáveis de tipo primitivo armazenam um valor de dado em uma posição de memória cujo endereço está associado ao nome da variável.

```
int ano = 10;
```

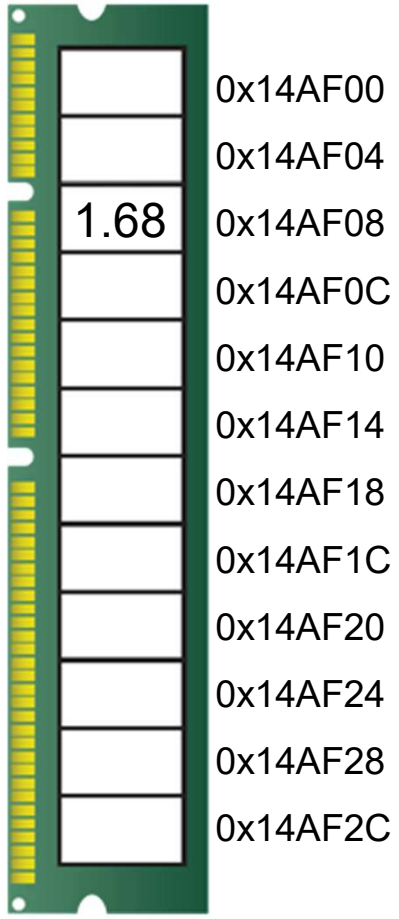


Tipos Primitivos

Colégio Pedro II

- Variáveis de tipo primitivo armazenam um valor de dado em uma posição de memória cujo endereço está associado ao nome da variável.

```
double altura = 1.68;
```



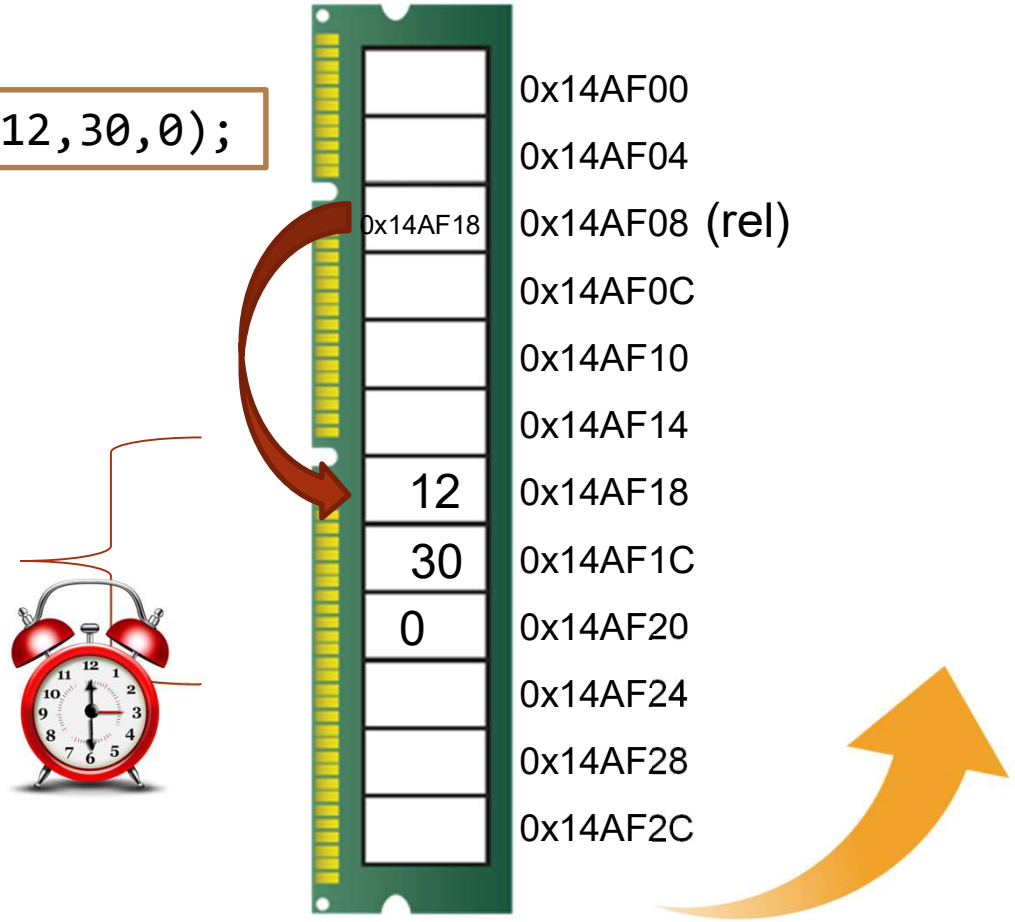
Até aqui... Nada novo...

Tipos por Referência

Colégio Pedro II

- Variáveis de referência não armazenam um valor propriamente, e sim um endereço de memória. Esse endereço de memória aponta onde de fato os dados dessa variável estão armazenados.

```
Relogio rel = new Relogio(12,30,0);
```



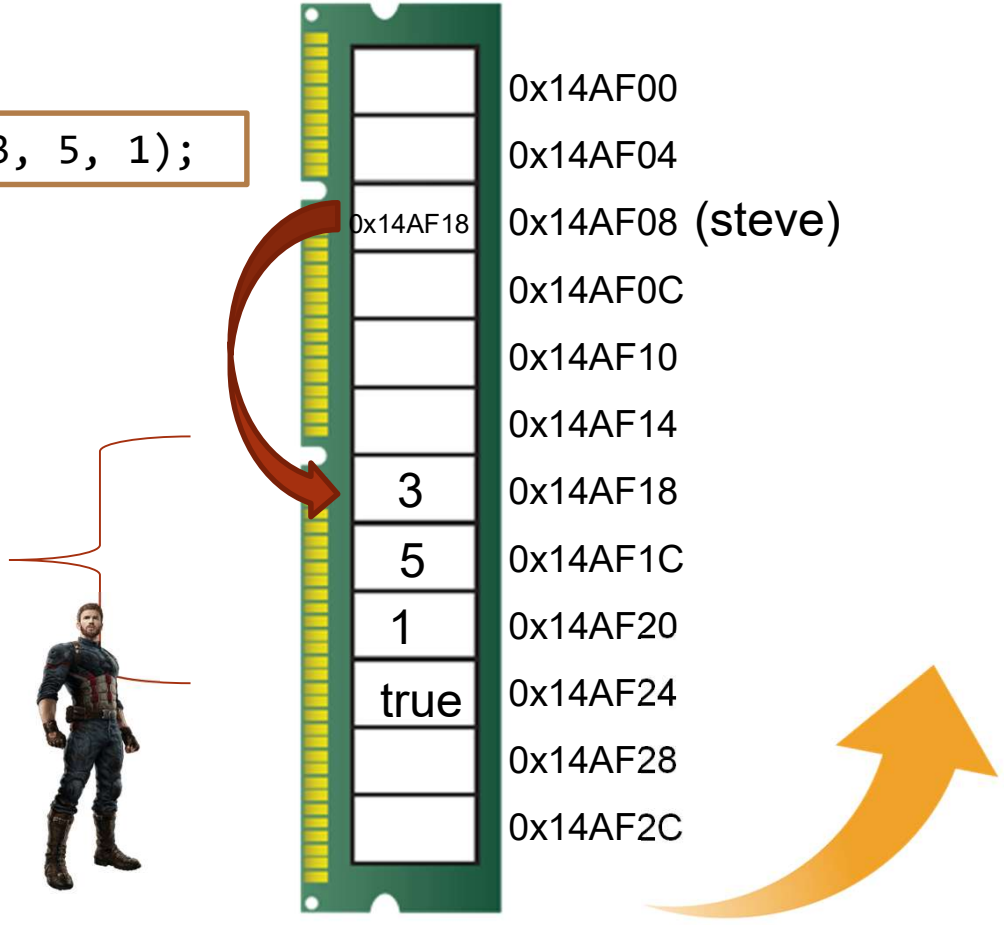
Lugar da memória onde o objeto **rel** da **classe Relógio** está armazenado.

Tipos por Referência

- Variáveis de referência não armazenam um valor propriamente, e sim um endereço de memória. Esse endereço de memória aponta onde de fato os dados dessa variável estão armazenados.

```
Heroi steve = new Heroi(true, 3, 5, 1);
```

Lugar da memória onde o objeto **steve** da **classe Herói** está armazenado.

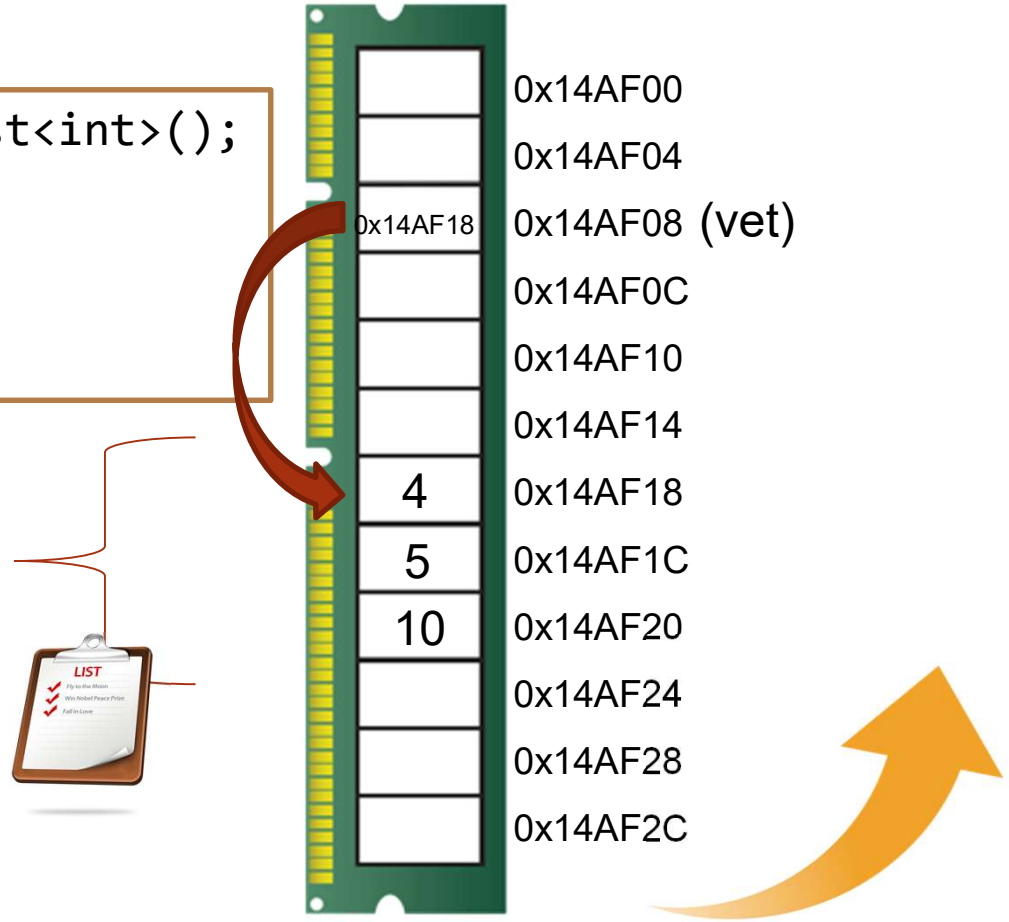


Tipos por Referência

- Variáveis de referência não armazenam um valor propriamente, e sim um endereço de memória. Esse endereço de memória aponta onde de fato os dados dessa variável estão armazenados.

```
List<int> list = new List<int>();

list.Add(4);
list.Add(5);
list.Add(10);
```

Lugar da memória onde o objeto **list** da classe **Lista de inteiros** está armazenado.



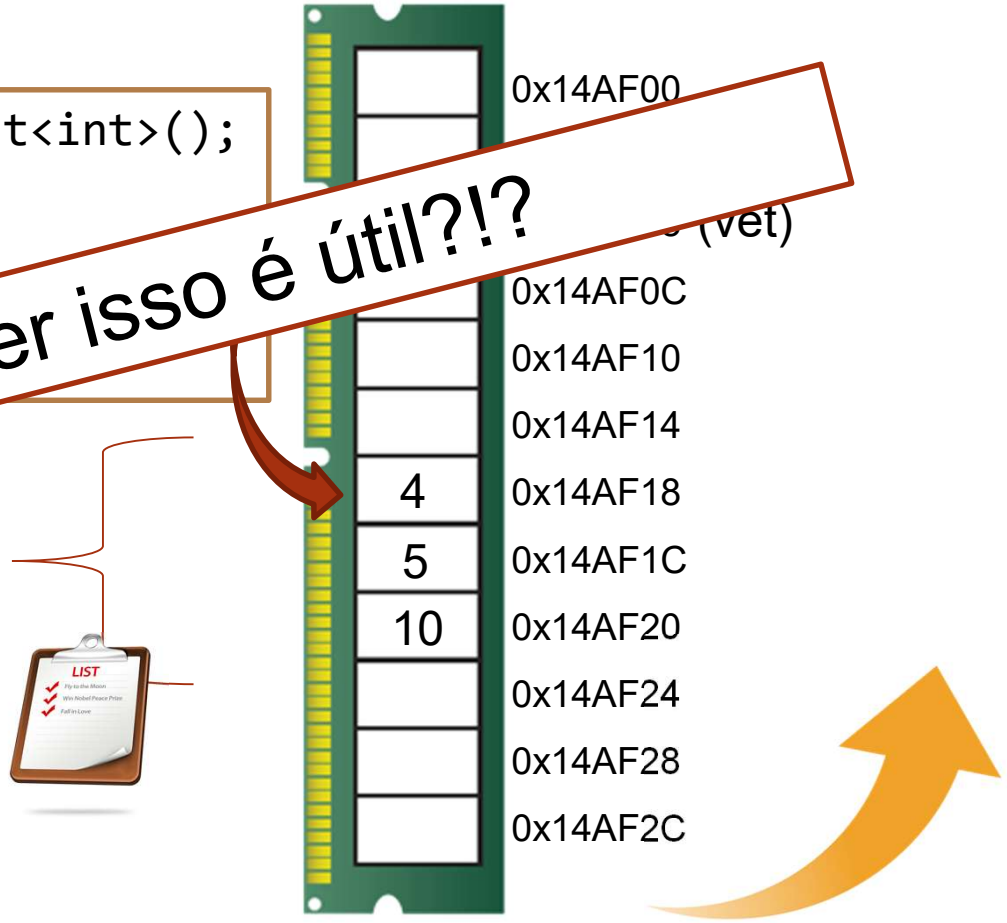
Tipos por Referência

- Variáveis de referência não armazenam um valor propriamente, e sim um endereço de memória. Esse endereço de memória aponta onde de fato os dados dessa variável estão armazenados.

```
List<int> list = new List<int>();

list.Add(4);
list.Add(5);
list.Add(10);
```

Por que saber isso é útil?!?

Lugar da memória onde o objeto **list** da classe **Lista de inteiros** está armazenado.



Passando variável primitiva a métodos

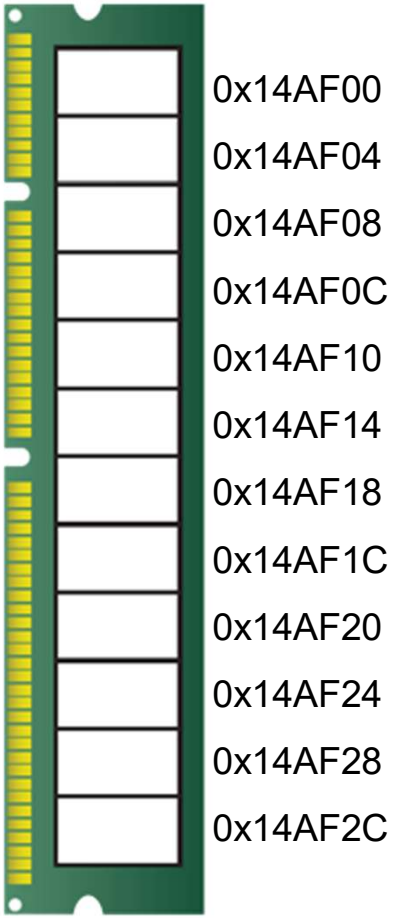
Sempre que um método é chamado, este recebe uma cópia dos valores passados como parâmetros e os coloca nos argumentos.

- Em C#, sempre que uma variável de tipo primitivo é passada como argumento, acontece uma simples cópia do seu valor para o argumento.

Passando variável primitiva a métodos

Colégio Pedro II

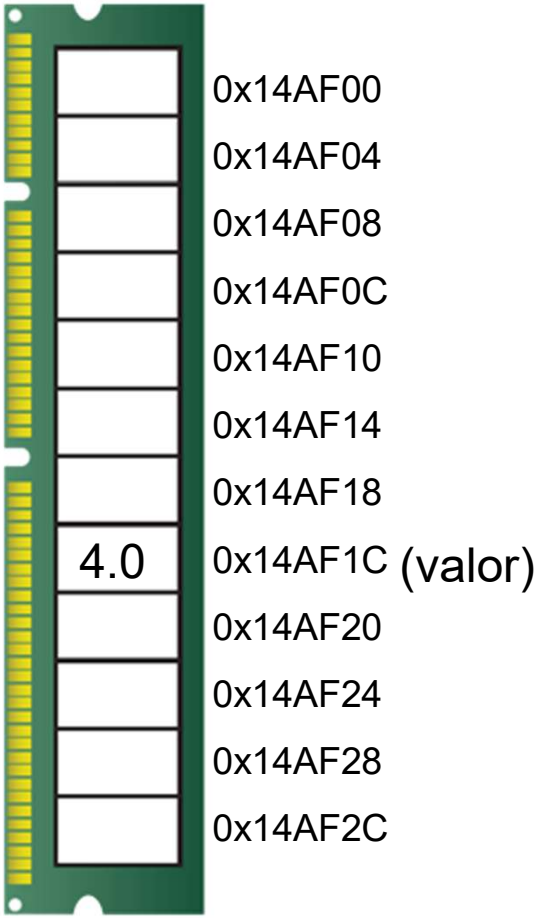
```
class Program
    {
        static void Main(string[] args)
        {
            float valor = 4.0F;

            float q = Quadrado(valor);

            Console.WriteLine("O quadrado vale {0}", q);

            Console.ReadLine();
        }

        static public float Quadrado(float val)
        {
            float q = val * val;
            return q;
        }
    }
```



Passando variável primitiva a métodos

Colégio Pedro II

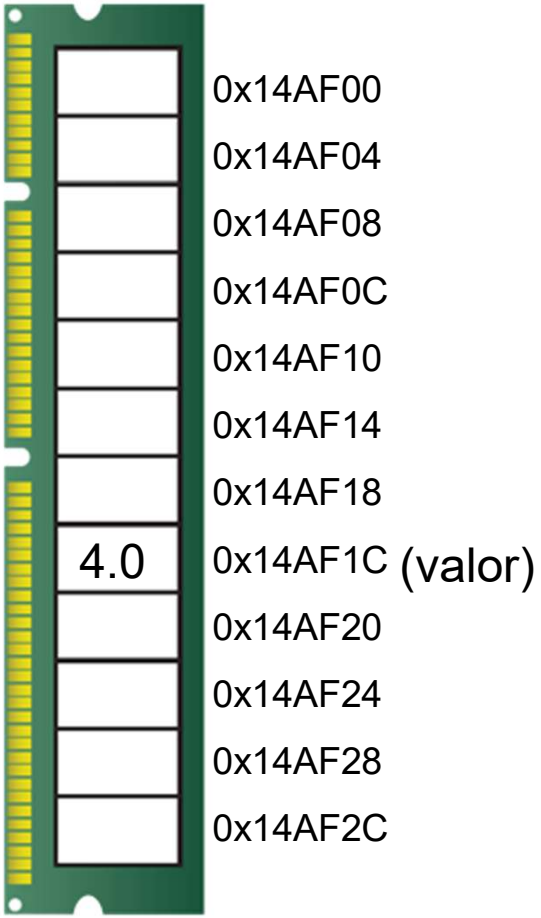
```
class Program
   {
       static void Main(string[] args)
       {
           float valor = 4.0F;

           float q = Quadrado(valor);

           Console.WriteLine("O quadrado vale {0}", q);

           Console.ReadLine();
       }

       static public float Quadrado(float val)
       {
           float q = val * val;
           return q;
       }
   }
```



Passando variável primitiva a métodos

Colégio Pedro II

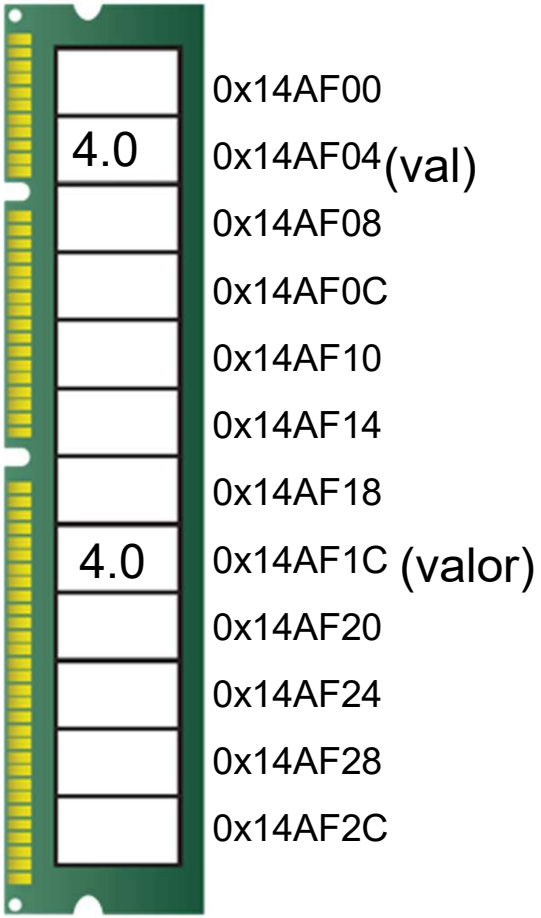
```
class Program
    {
        static void Main(string[] args)
        {
            float valor = 4.0F;

            float q = Quadrado(valor);

            Console.WriteLine("O quadrado vale {0}", q);

            Console.ReadLine();
        }

        static public float Quadrado(float val)
        {
            float q = val * val;
            return q;
        }
    }
```



Passando variável primitiva a métodos

Colégio Pedro II

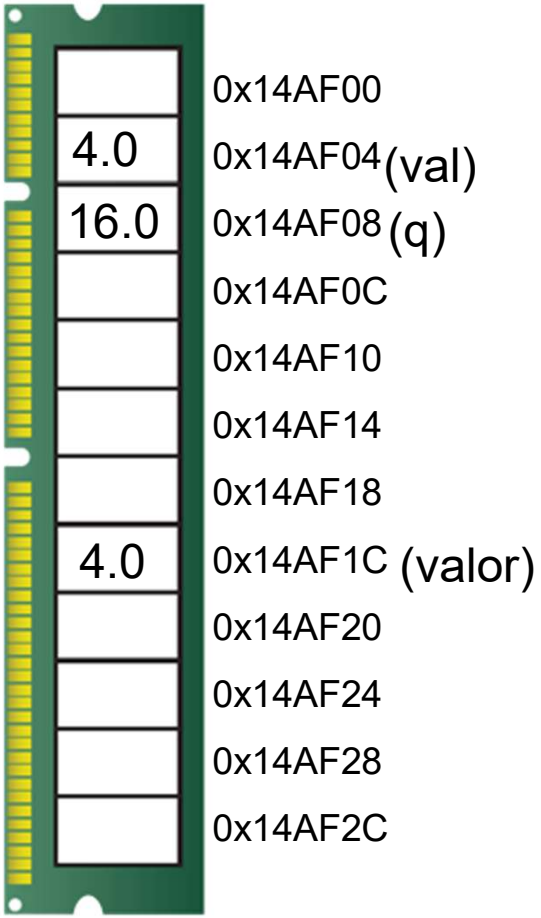
```
class Program
    {
        static void Main(string[] args)
        {
            float valor = 4.0F;

            float q = Quadrado(valor);

            Console.WriteLine("O quadrado vale {0}", q);

            Console.ReadLine();
        }

        static public float Quadrado(float val)
        {
            float q = val * val;
            return q;
        }
    }
```



Passando variável primitiva a métodos

Colégio Pedro II

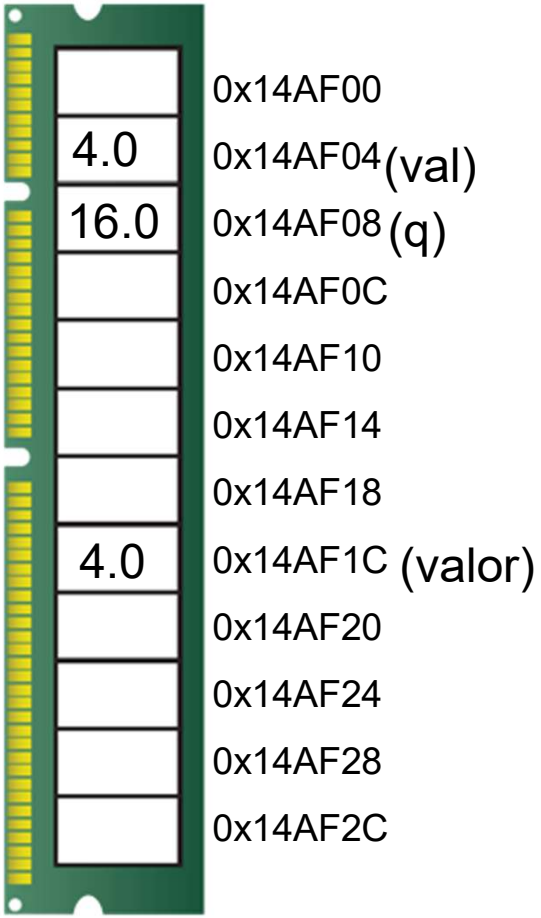
```
class Program
    {
        static void Main(string[] args)
        {
            float valor = 4.0F;

            float q = Quadrado(valor);

            Console.WriteLine("O quadrado vale {0}", q);

            Console.ReadLine();
        }

        static public float Quadrado(float val)
        {
            float q = val * val;
            return q;
        }
    }
```



Passando variável primitiva a métodos

Colégio Pedro II

```
class Program
    {
        static void Main(string[] args)
        {
            float valor = 4.0F;

            float q = Quadrado(valor);

            Console.WriteLine("O quadrado vale {0}", q);

            Console.ReadLine();
        }

        static public float Quadrado(float val)
        {
            float q = val * val;
            return q;
        }
    }
```

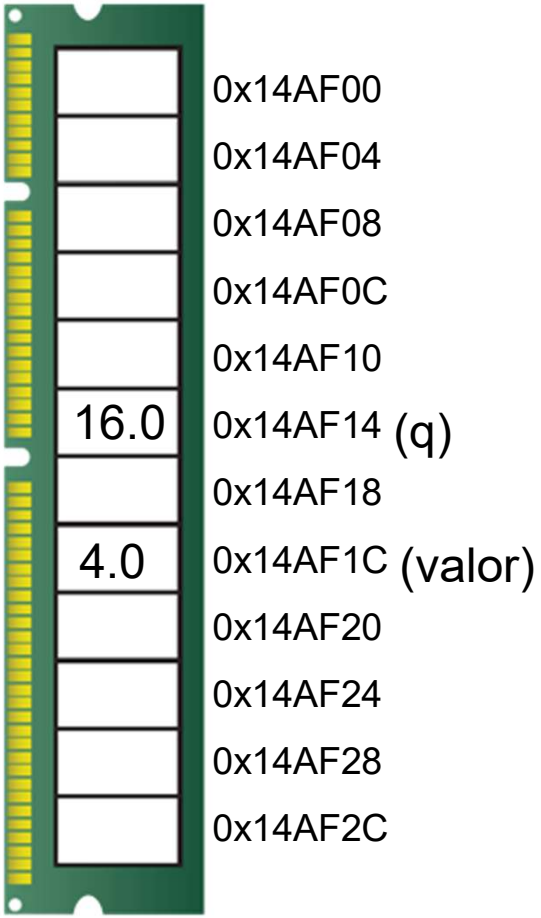


Fim do escopo do método Quadrado! Todas as variáveis locais serão apagadas da memória.

Passando variável primitiva a métodos

Colégio Pedro II

```
class Program
    {
        static void Main(string[] args)
        {
            float valor = 4.0F;

            float q = Quadrado(valor);

            Console.WriteLine("O quadrado vale {0}", q);

            Console.ReadLine();
        }

        static public float Quadrado(float val)
        {
            float q = val * val;
            return q;
        }
    }
```

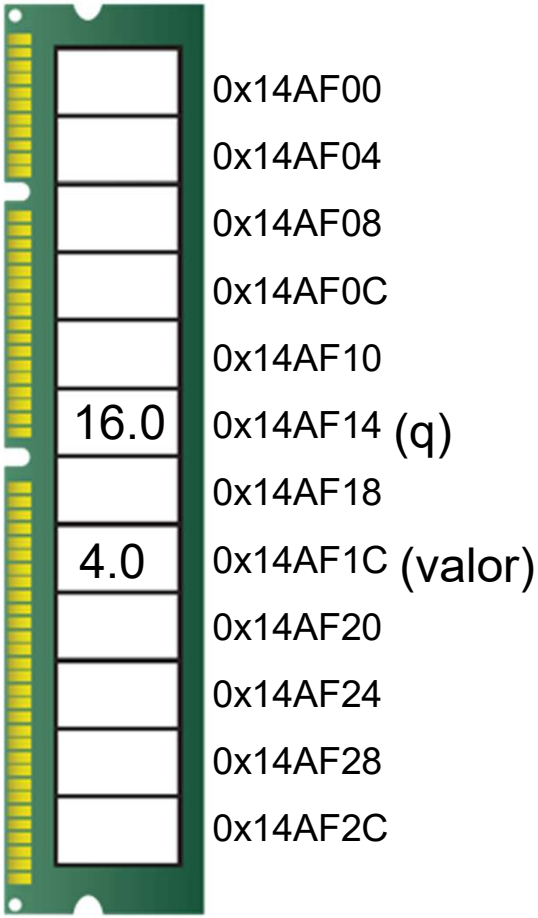


Passando variável primitiva a métodos

Colégio Pedro II

O quadrado vale 16

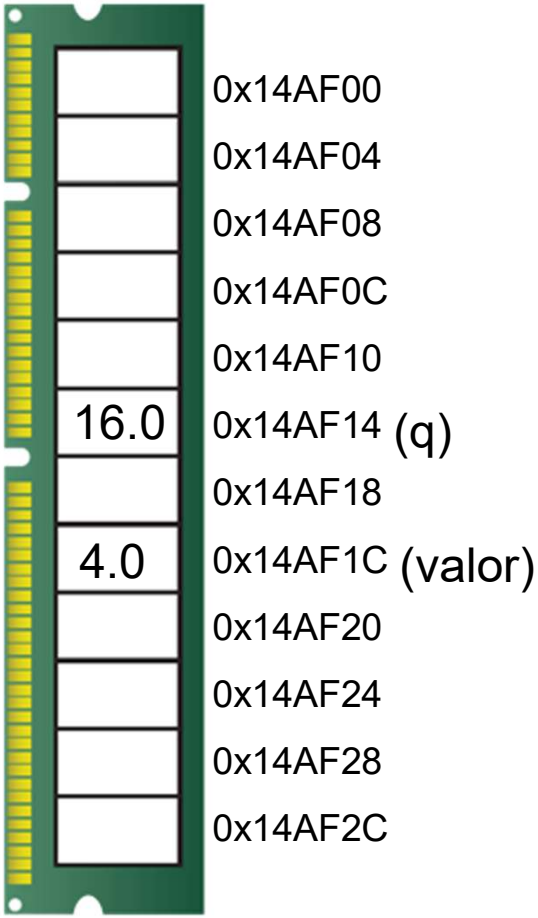
```
class Program
    {
        static void Main(string[] args)
        {
            float valor = 4.0F;

            float q = Quadrado(valor);

            Console.WriteLine("O quadrado vale {0}", q);

            Console.ReadLine();
        }

        static public float Quadrado(float val)
        {
            float q = val * val;
            return q;
        }
    }
```



Passando variável primitiva a métodos

Colégio Pedro II

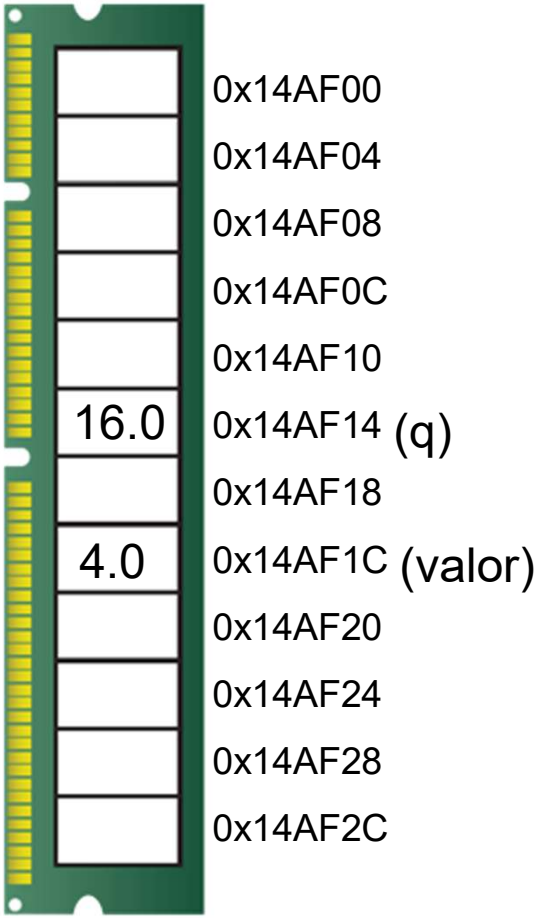
```
class Program
    {
        static void Main(string[] args)
        {
            float valor = 4.0F;

            float q = Quadrado(valor);

            Console.WriteLine("O quadrado vale {0}", q);

            Console.ReadLine();
        }

        static public float Quadrado(float val)
        {
            float q = val * val;
            return q;
        }
    }
```



Passando variável primitiva a métodos

Colégio Pedro II

```
class Program
    {
        static void Main(string[] args)
        {
            float valor = 4.0F;

            float q = Quadrado(valor);

            Console.WriteLine("O quadrado vale {0}", q);

            Console.ReadLine();
        }

        static public float Quadrado(float val)
        {
            float q = val * val;
            return q;
        }
    }
```

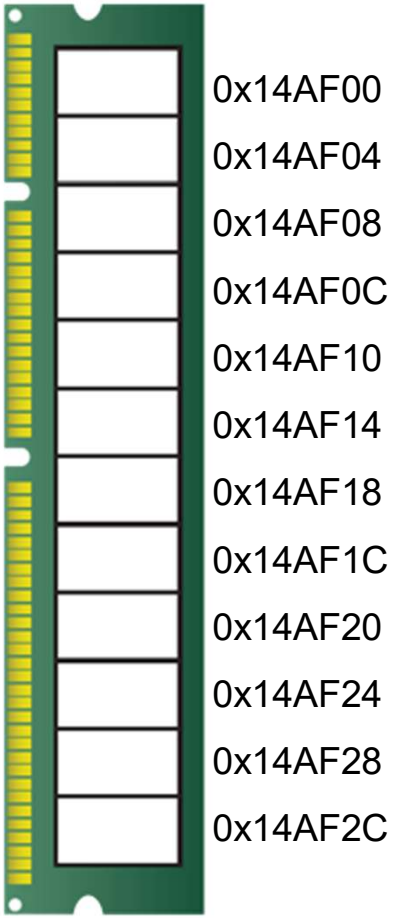


Fim do escopo do método Main! Todas as variáveis locais serão apagadas da memória.

Passando variável primitiva a métodos

Colégio Pedro II

```
class Program
    {
        static void Main(string[] args)
        {
            float valor = 4.0F;

            float q = Quadrado(valor);

            Console.WriteLine("O quadrado vale {0}", q);

            Console.ReadLine();
        }

        static public float Quadrado(float val)
        {
            float q = val * val;
            return q;
        }
    }
```



Passando variável de referência a métodos

Sempre que um método é chamado, este recebe uma cópia do valor desse argumento.

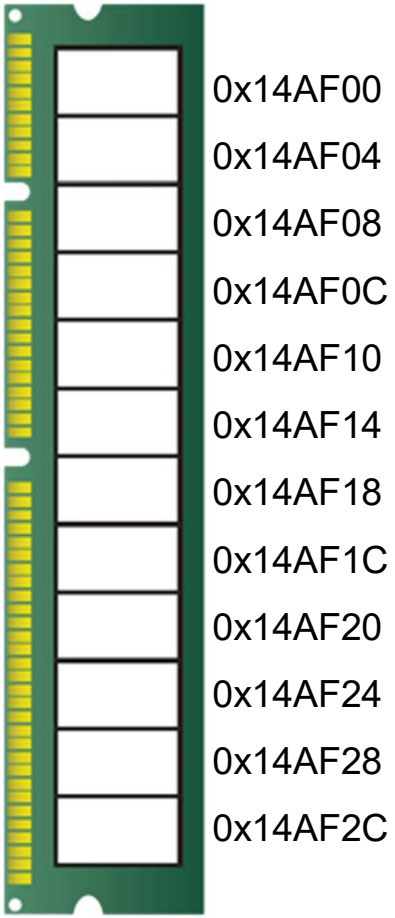
- Em C#, sempre que uma variável de referência, como um objeto, é passada como argumento de um método, então o método terá acesso a variável do escopo que a chamou.

- Isso é intuitivo já que o valor passado na realidade como argumento é uma cópia da referência!

Passando variável de referência a métodos

Colégio Pedro II

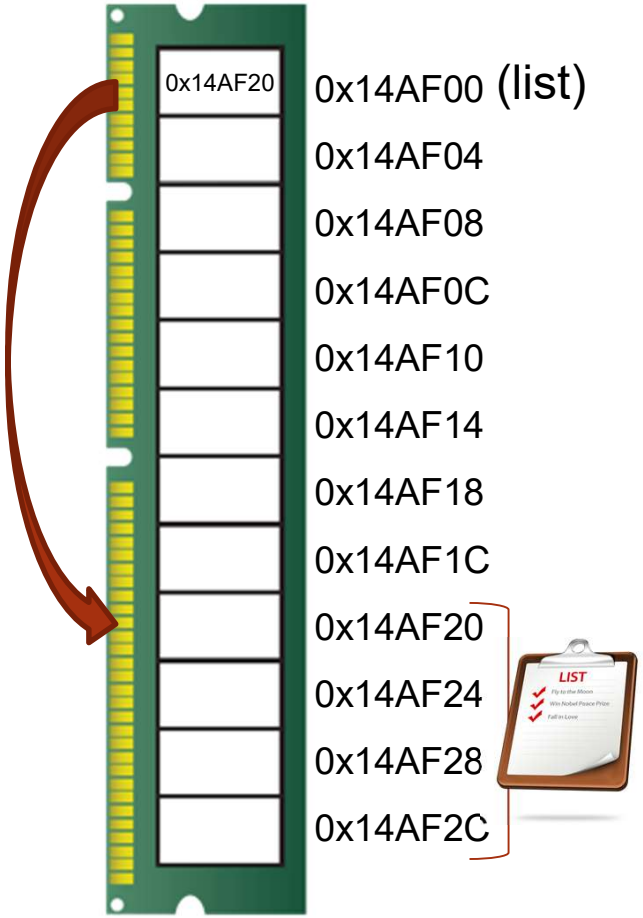
```
class Program
   {
       static void Main(string[] args)
       {
           List<float> list = new List<float>();
           list.Add(2); list.Add(5); list.Add(2);

                ZeraLista(list);

           foreach(float elemento in argList)
                 Console.WriteLine(elemento);

           Console.ReadLine();
       }

       static public void ZeraLista(List<float> argList)
       {
                 foreach(float elemento in argList)
                      elemento = 0;
       }

   }
```



Passando variável de referência a métodos

Colégio Pedro II

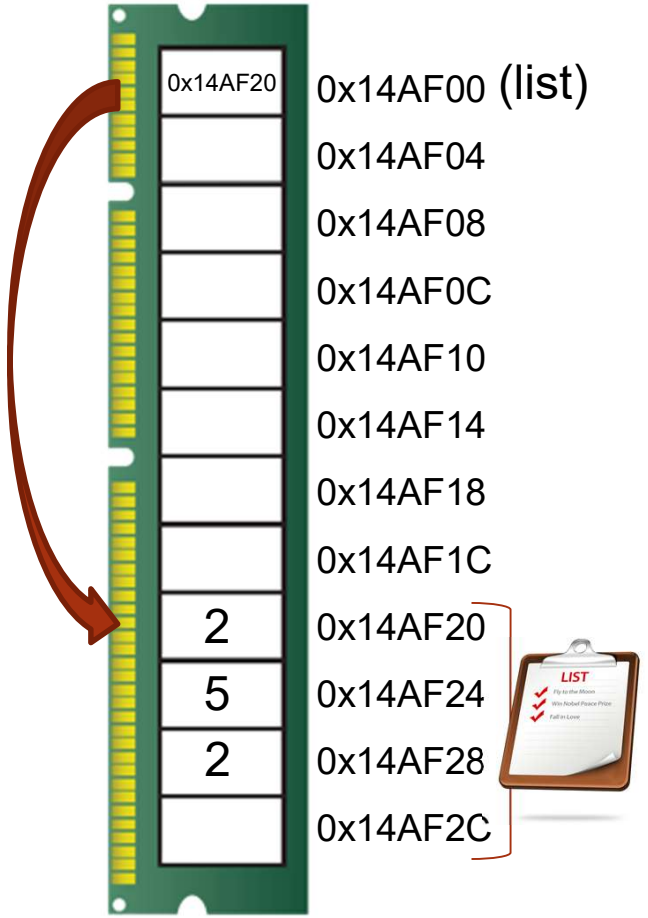
```
class Program
   {
       static void Main(string[] args)
       {
           List<float> list = new List<float>();
           list.Add(2); list.Add(5); list.Add(2);

                ZeraLista(list);

           foreach(float elemento in argList)
                 Console.WriteLine(elemento);

           Console.ReadLine();
       }

       static public void ZeraLista(List<float> argList)
       {
                 foreach(float elemento in argList)
                      elemento = 0;
       }

   }
```



Passando variável de referência a métodos

Colégio Pedro II

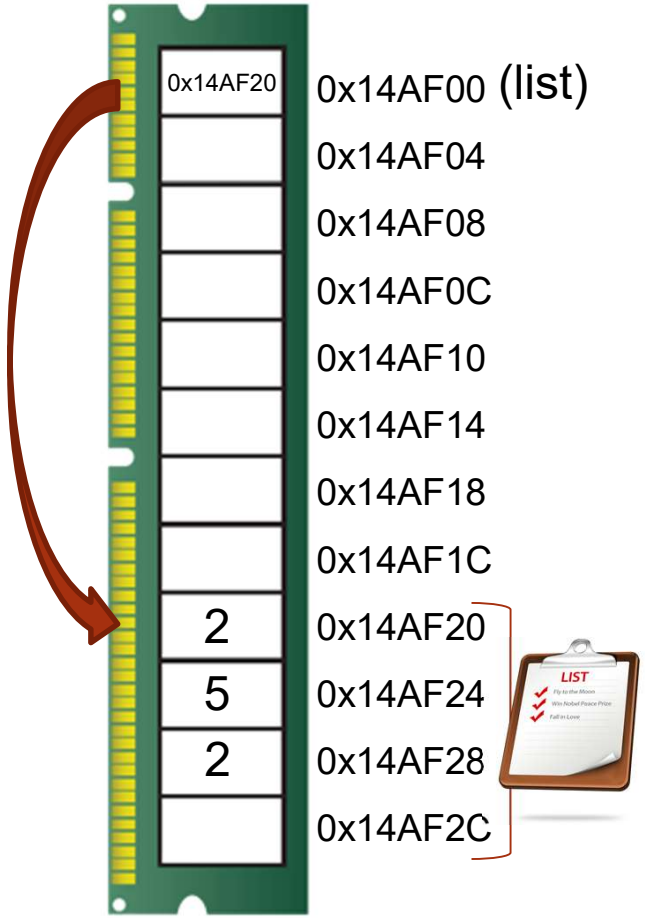
```
class Program
   {
       static void Main(string[] args)
       {
           List<float> list = new List<float>();
           list.Add(2); list.Add(5); list.Add(2);

                ZeraLista(list);

           foreach(float elemento in argList)
                 Console.WriteLine(elemento);

           Console.ReadLine();
       }

       static public void ZeraLista(List<float> argList)
       {
                 foreach(float elemento in argList)
                      elemento = 0;
       }

   }
```



Passando variável de referência a métodos

Colégio Pedro II

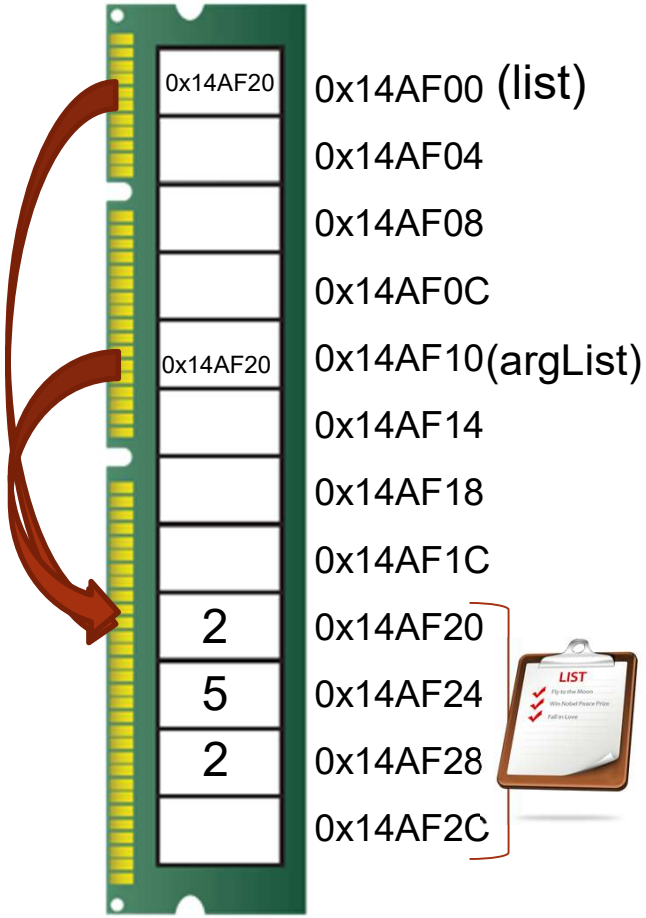
```
class Program
   {
       static void Main(string[] args)
       {
           List<float> list = new List<float>();
           list.Add(2); list.Add(5); list.Add(2);

                ZeraLista(list);

           foreach(float elemento in argList)
                 Console.WriteLine(elemento);

           Console.ReadLine();
       }

       static public void ZeraLista(List<float> argList)
       {
                 foreach(float elemento in argList)
                      elemento = 0;
       }

   }
```



Passando variável de referência a métodos

Colégio Pedro II

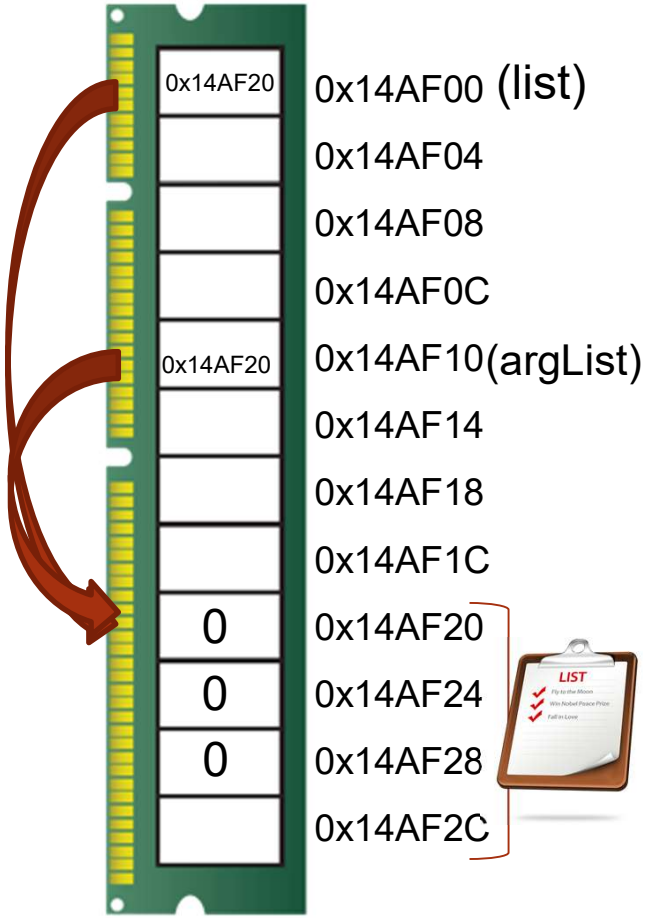
```
class Program
   {
       static void Main(string[] args)
       {
           List<float> list = new List<float>();
           list.Add(2); list.Add(5); list.Add(2);

                ZeraLista(list);

           foreach(float elemento in argList)
                 Console.WriteLine(elemento);

           Console.ReadLine();
       }

       static public void ZeraLista(List<float> argList)
       {
                 foreach(float elemento in argList)
                      elemento = 0;
       }

   }
```



Passando variável de referência a métodos

Colégio Pedro II

```
class Program
   {
       static void Main(string[] args)
       {
           List<float> list = new List<float>();
           list.Add(2); list.Add(5); list.Add(2);

                ZeraLista(list);

           foreach(float elemento in argList)
                 Console.WriteLine(elemento);

           Console.ReadLine();
       }

       static public void ZeraLista(List<float> argList)
       {
                foreach(float elemento in argList)
                     elemento = 0;
       }

   }
```

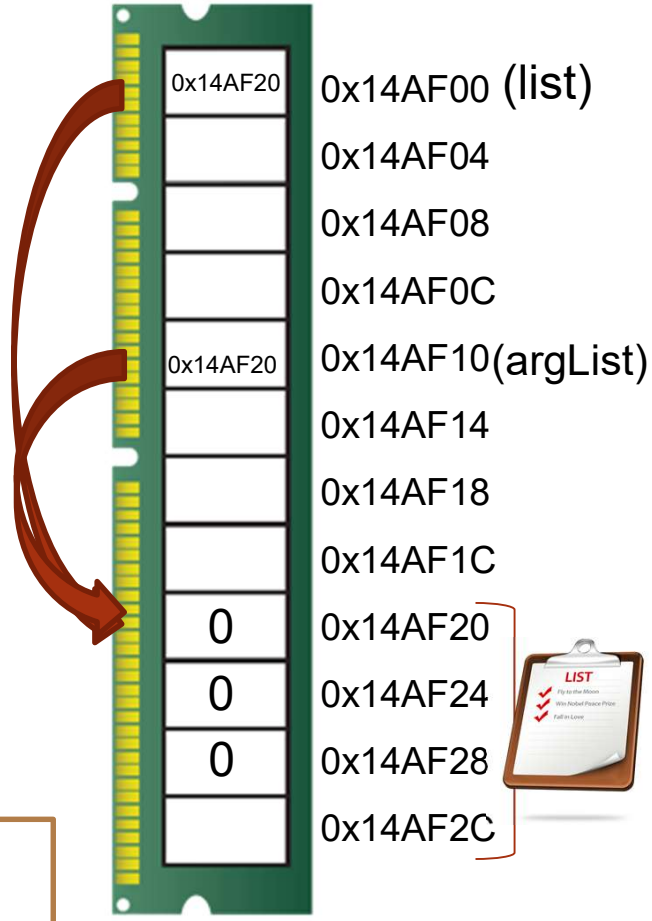


Passando variável de referência a métodos

Colégio Pedro II

```
class Program
   {
       static void Main(string[] args)
       {
           List<float> list = new List<float>();
           list.Add(2); list.Add(5); list.Add(2);

                ZeraLista(list);

           foreach(float elemento in argList)
                 Console.WriteLine(elemento);

           Console.ReadLine();
       }

       static public void ZeraLista(List<float> argList)
       {
                 foreach(float elemento in argList)
                      elemento = 0;
       }

   }
```

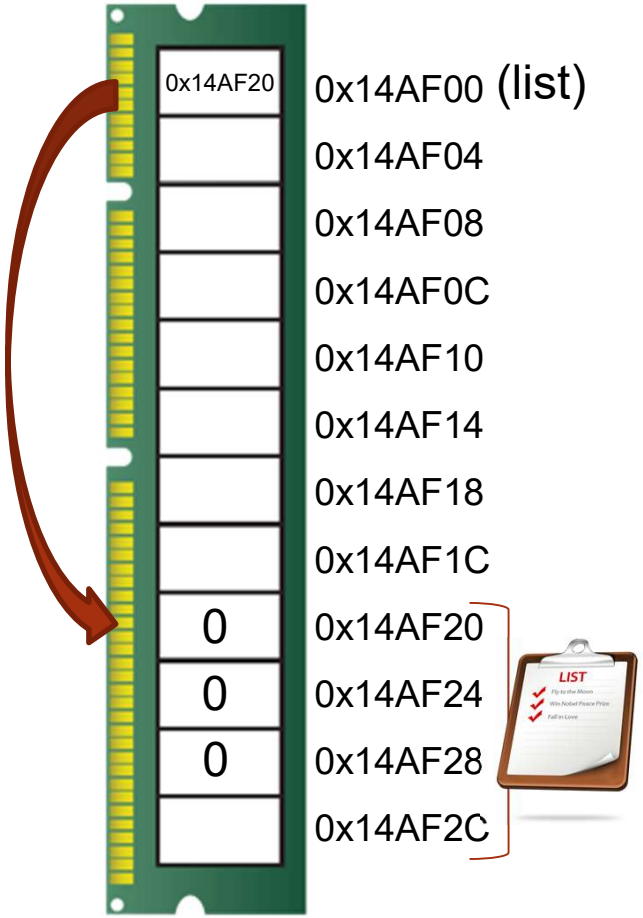
Fim do escopo do método ZeraLista! Todas as variáveis locais serão apagadas da memória.



Passando variável de referência a métodos

Colégio Pedro II

```
class Program
    {
        static void Main(string[] args)
        {
            List<float> list = new List<float>();
            list.Add(2); list.Add(5); list.Add(2);

                ZeraLista(list);

            foreach(float elemento in argList)
                    Console.WriteLine(elemento);

            Console.ReadLine();
        }

        static public void ZeraLista(List<float> argList)
        {
                    foreach(float elemento in argList)
                        elemento = 0;
        }

    }
```

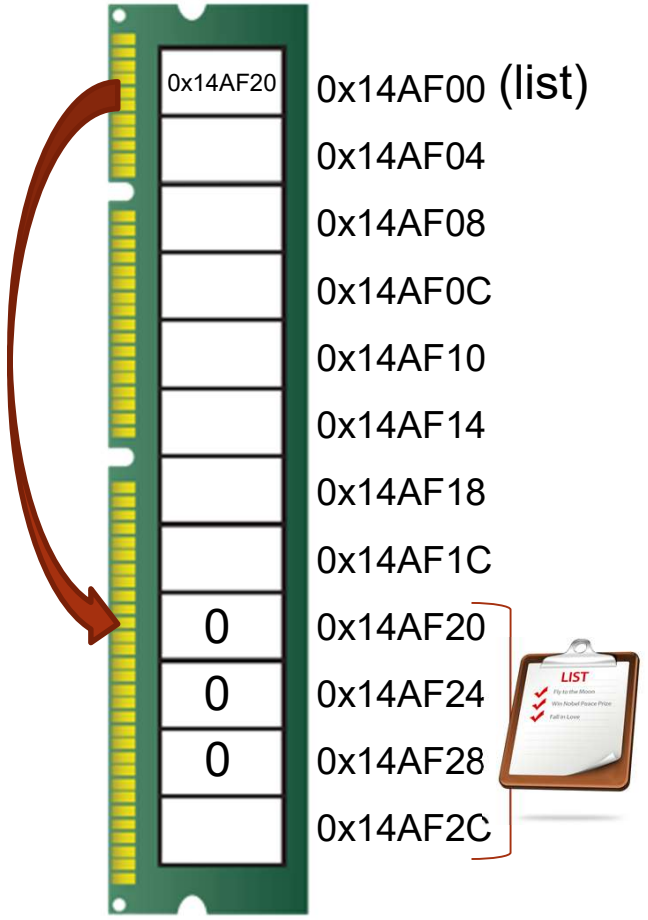


Passando variável de referência a

Colégio Pedro II

```
0
0
0
```

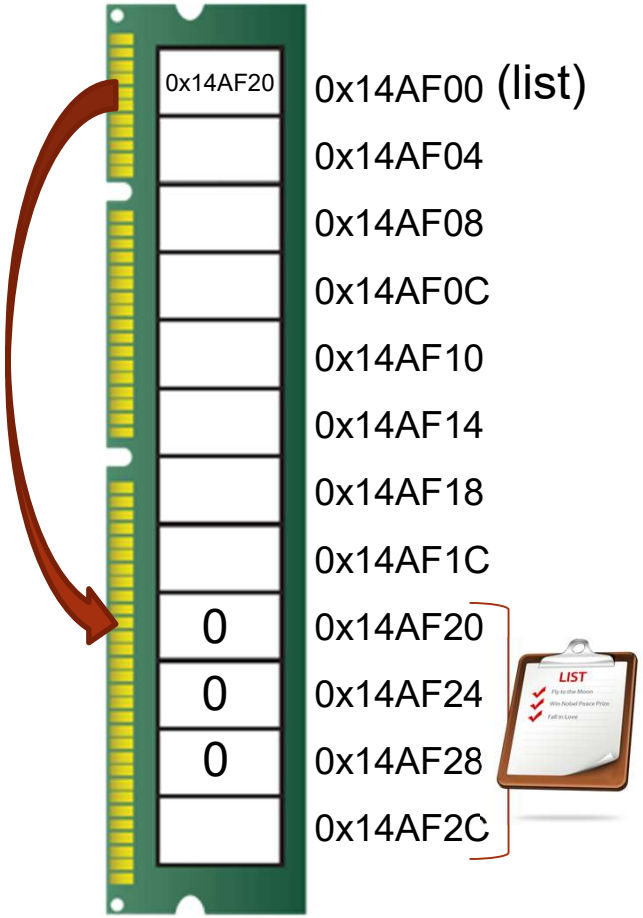
```
class Program
    {
        static void Main(string[] args)
        {
            List<float> list = new List<float>();
            list.Add(2); list.Add(5); list.Add(2);

                ZeraLista(list);

            foreach(float elemento in argList)
                    Console.WriteLine(elemento);

            Console.ReadLine();
        }

        static public void ZeraLista(List<float> argList)
        {
                    foreach(float elemento in argList)
                        elemento = 0;
        }

    }
```



Passando variável de referência a métodos

Colégio Pedro II

```
class Program
   {
       static void Main(string[] args)
       {
           List<float> list = new List<float>();
           list.Add(2); list.Add(5); list.Add(2);

                ZeraLista(list);

           foreach(float elemento in argList)
                 Console.WriteLine(elemento);

           Console.ReadLine();
       }

       static public void ZeraLista(List<float> argList)
       {
                 foreach(float elemento in argList)
                      elemento = 0;
       }

   }
```

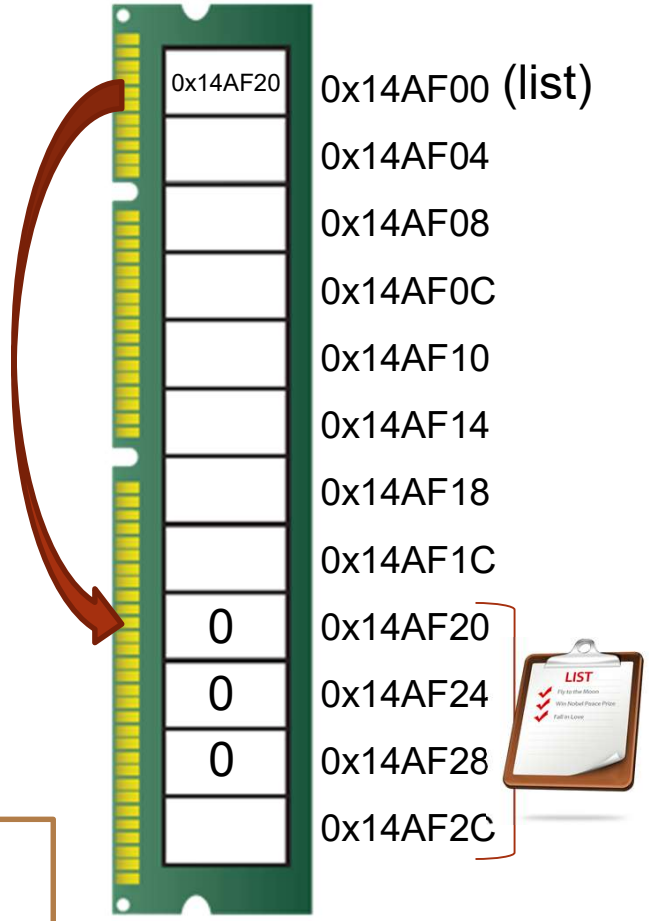


Passando variável de referência a métodos

Colégio Pedro II

```
class Program
   {
       static void Main(string[] args)
       {
           List<float> list = new List<float>();
           list.Add(2); list.Add(5); list.Add(2);

                ZeraLista(list);

           foreach(float elemento in argList)
                 Console.WriteLine(elemento);

           Console.ReadLine();
       }

       static public void ZeraLista(List<float> argList)
       {
                 foreach(float elemento in argList)
                      elemento = 0;
       }
   }
```

Fim do escopo do método Main! Todas as variáveis locais serão apagadas da memória.



Passando variável de referência a métodos

Colégio Pedro II

```
class Program
   {
       static void Main(string[] args)
       {
           List<float> list = new List<float>();
           list.Add(2); list.Add(5); list.Add(2);

                ZeraLista(list);

           foreach(float elemento in argList)
                 Console.WriteLine(elemento);

           Console.ReadLine();
       }

       static public void ZeraLista(List<float> argList)
       {
                 foreach(float elemento in argList)
                      elemento = 0;
       }

   }
```