EXPLICANDO A AULA: Exercício: Funções que retornam Arrays Invertidos

Código da aula:

https://github.com/professormarcosp/AprendaCParaGamesUE4/blob/master/Arrays/RetornandoArrays.cpp

Na primeira linha do código declarei esta função

```
void DuplicaVetor(int *Array, int Tam);
```

Observe que ele não retorna nada(void) e será capaz de receber dois argumentos os quais serão enviados aos parâmetros desta função que são: int *Array e int Tam

O Parâmetro int *Array indica que Array é um ponteiro capaz de receber o endereço de memória de uma variável que contém nela um número inteiro.

int Tam – Este vai receber um valor inteiro(int) que será o tamanho do vetor

Veja que na função main() nós declaramos um vetor de nome Numeros e já iniciamos ele com dez elementos.

```
int Numeros[] = { 1,2,3,4,5,6,7,8,9,10 };
```

Lembre que podemos omitir o valor do colchete [] pois na hora da compilação o compilador irá contar quantidade de elementos em $\{1,2,3,4,5,6,7,8,9,10\}$ e vai colocar int Numeros[10] = $\{1,2,3,4,5,6,7,8,9,10\}$;

```
int Tam = sizeof(Numeros) / sizeof(int);
```

Já o tamanho(Tam) como já mencionado em aulas anteriores é calculado da seguinte forma

sizeof(Numeros) irá retornar o número de bytes total do vetor Numeros. E como são 10 inteiros e assumindo na arquitetura do computador cada inteiro tenha 4 bytes. sizeof(Numeros) irá retornar 10 elementos vezes 4 bytes sendo portando $10 \times 4 = 40$ bytes

já sizeof(int) irá retornar tamanho do tipo de dados int que neste caso é 4bytes

Portanto a expressão int Tam = sizeof(Numeros) / sizeof(int) ficará assim:

```
int Tam = 40 / 10;
```

logo int Tam = 10

assim a variável Tam da função main() terá valor 10(Dez) que é o número de elementos do vetor Numeros

E ()Então ao chamarmos a função DuplicaVetor na função main

```
int main()
{
    int Numeros[] = { 1,2,3,4,5,6,7,8,9,10 };
    int Tam = sizeof(Numeros) / sizeof(int);

    std::cout << "sizeof(Numeros)" << sizeof(Numeros) << " Bytes\n";
    std::cout << "sizeof(&Numeros[0])" << sizeof(&Numeros[0]) << " Bytes\n";

    //chama a função e passa o vetor Números para ela
    //0 endereço do primeiro elemento do vetor será enviado para Array
    MostraVetor(Numeros, Tam);</pre>
```

```
DuplicaVetor(Numeros, Tam);
   MostraVetor(Numeros, Tam);
   //Aqui Mostrar vetor recebe o retorno da função Inverte Vetor
   //E este retorno será o endereço do primeiro elemento do array inventido
   MostraVetor(InverteVetor(Numeros, Tam), Tam);
   system("PAUSE");
   return 0;
}
```

Ao chamarmos a função DuplicaVetor, estaremos na realidade enviado estes argumentos:

```
void DuplicaVetor(&Numero[0], 10);
```

Lembre-se e como já comentei em outras aulas que nas linguagens C++ e C o nome de um vetor é o mesmo que endereço de memória do primeiro elemento deste vetor ou &NomeDoVetor[0]

Logo ao chamar a função DuplicaVetor na função main você está na realidade enviando o argumentos desta forma.

DuplicaVetor(&Numeros[0], Tam);

Se você quiser inclusive você pode usar explicitamente &Numeros[0] ao invés de apenas o nome do vetor Numeros.

Agora após chamar a função DuplicaVetor(Numeros, Tam) os argumentos Numeros, Tam serão enviados respectivamente para os parâmetros int *Array, int Tam desta função que tem assinatura:

```
void DuplicaVetor(int *Array, int Tam)
```

Logo a variável ponteiro de inteiro Array (int *Array) irá receber o argumento Numeros que é na realidade &Numeros[0]. E o que isso significa?

Significa que a variável Array que é parâmetro int *Array da função DuplicaVetor irá armazenar dentro dela o endereço de memória do primeiro elemento do vetor Numeros

Ou seja, quando você declara uma função e indica que ela terá parâmetros, quando ela for chamada estes parâmetros são variáveis locais a esta função e serão criados e armazenados na memória apenas quando a função estiver sendo executada. Depois que a função for finalizada estas variáveis Array e Tam que são os parâmetros da função DuplicaVetor, serão descartadas automaticamente da memória e por isso, chamados estes parâmetros de variáveis locais a função.

Uma observação importante é que apesar de Tam ter o mesmo nome da variável Tam declarada na função main() eles são diferentes, pois são recursos diferentes e estão em endereços de memória diferentes! Uma é uma variável local da função main e a outro é uma variável local da função DuplicaVetor

Após enviar os argumentos para os parâmetros é hora de executar o código da função

Então o vamos supor para fins de aprendizado e simplificação que o parâmentro Array ou a variável declarada na função int* Array recebe o valor do endereço de memória 100

Ou seja, estou simplificando e imaginando que o endereço de memória do primeiro elemento do vetor Numeros seja 100 ou seja &Numeros[0] é cem! 100

Como inteiro int tem 4 bytes na memória o vetor Numeros estaria assim:

				N	umeros					
Índice do vetor	0	1	2	3	4	5	6	7	8	9
Endereço	100-103	104-107	108-111	112-115	116-119	120-123	124-127	128-131	132-135	136-139
Valor	1	2	3	4	5	6	7	8	9	10

Ou seja, o endereço de memória do primeiro elemento do vetor é 100 e para ler cada elemento contamos 4 bytes pois este é um vetor de inteiros ou do tipo int

Logo como o compilador sabe que é um array de inteiros pois você declarou assim:

```
int Numeros[] = \{1,2,3,4,5,6,7,8,9,10\};
```

Ele saberá ler e percorrer estes array pois, a cada 4 bytes lidos você lê um elemento deste vetor

E ai temos a seguir no código o trecho do loop for ou repetição usando for

Temos a variável \mathbf{i} de controle do loop for e que é declarada e iniciada com o valor zero int $\mathbf{i} = 0$

Além disso, temos a condição i < Tam, ou seja, como Tam tem valor 10 neste caso a condição de permanência no loop for será i < 10

Logo i vai de 0 até 9 e depois o loop é encerrado

Depois temos que incrementar i para ele ir para próximo elemento através de i++

Depois já no corpo do loop for entra o comando Array[i] = 2 * Array[i];

O que ele significa?

Lembre da aula de Aritmética de ponteiros!

Como Array é um ponteiro então Array[i] é o mesmo que *(Array + i)

Veja que você pode inclusive colocar desta forma no código e irá funcionar!

```
void DuplicaVetor(int* Array, int Tam)
{
    //função percorre o vetor e duplica os valores
    //do vetor
    for (int i = 0; i < Tam; i++)
    {
        //Array[i] = 2 * Array[i];
        *(Array + i) = 2 * *(Array + i);
    }
}</pre>
C:\Users\curso\source\repos\TesteAlunos\Debug\TesteAlunos.exe
[1 2 3 4 5 6 7 8 9 10 ]
[2 4 6 8 10 12 14 16 18 20 ]
Pressione qualquer tecla para continuar. . .
```

Logo **Array**[i] é uma simplicação, um atalho para o comando *(**Array** + i)

C++ manteve, portanto, algumas simplificações para poupar código e digitação, o que é importante para diminuir tamanho do código para ambientes como poucos recursos de memória, etc como microcontroladores, por exemplo.

Logo *(Ponteiro + i) pode ser simplicado para apenas Ponteiro[i]

Logo *(Array + i) é o mesmo que Array[i]

Mas o que significa então *(Array + i) = 2 * *(Array + i); ou Array[i] = 2 * Array[i];

Vamos percorrer o loop for e ver o que está ocorrendo

Primeiro i = 0 e Array recebe o valor do endereço de memória do primeiro elemento do vetor e Tam recebe o valor da Variavel Tam de main enviada.

Então temos que na função DuplicaVetor a variável Array irá receber o valor do endereço do primeiro elemento do vetor Numeros (&Numeros[0]) e que assumimos que seja 100 por questões da explicação

```
//chama a função e passa o vetor Números para ela
//O endereço do primeiro elemento do vetor será enviado para
MostraVetor(Numeros, Tam);
DuplicaVetor(Numeros, Tam);
MostraVetor(Numeros, Tam);
//Aqui Mostrar vetor recebe o retorno da função Inverte Vetor
//E este retorno será o endereço do primeiro elemento do arra
MostraVetor(InverteVetor(Numeros, Tam), Tam);
system("PAUSE");
return 0;
}

void DuplicaVetor(int* Array, int Tam)
{
    //função percorre o vetor e duplica os valores
    //do vetor
    for (int i = 0: i < Tam: i+t)</pre>
```

Logo Array, que é um ponteiro de inteiros, terá armazenado dentro dele o valor de memória 100 e Tam terá o valor 10. Ambos valores passados via argumentos na chamada da função DuplicaVetor como você observa acima na figura.

Temos na memória Ram carregado o vetor Numeros

				N	umeros					
Índice do vetor	0	1	2	3	4	5	6	7	8	9
Endereço	100-103	104-107	108-111	112-115	116-119	120-123	124-127	128-131	132-135	136-139
Valor	1	2	3	4	5	6	7	8	9	10

Então vamos executar o loop passo a passo

Para
$$i = 0$$
; temos Array[0] = 2 * Array[0]; ou *(Array + 0) = 2 * *(Array + 0);

Logo como Array recebeu o valor do endereço de memória de Numeros é isso que o compilador "enxergará":

$$*(100 + 0) = 2 * *(100 + 0);$$

ou $*(100) = 2 * *(100);$

Lembre que o operador de desrefenciamento asterisco * indica que queremos o valor armazenado no endereço de memória apontado pelo ponteiro.

Logo *(100) irá retornar o valor 1 do vetor Numeros, pois 100 é o endereço de memória do primeiro elemento do vetor Numeros...

Mas observe que o primeiro elemento está armazenado nos endereços 100 à 103

Mas uma vez relembrando o que já comentei em aulas anteriores, o endereço de memória de uma variável é o primeiro endereço que ela está armazenada, não importando o tamanho dela. Inclusive uma variável do tipo ponteiro só armazena um único e somente um endereço de memória.

Mas como o operador de desreferenciamento * sabe ler variável se o ponteiro só tem armazenado dentro dele o primeiro endereço desta variável?

É aí que entra o tipo do ponteiro. Como é um ponteiro do tipo int o compilador sabe que são 4 bytes de tamanho. Logo o primeiro elemento será lido os endereços 100, 101, 102 e 103 totalizando 4 bytes!

Endereço	100-103	104-107	108-111
Valor	1	2	3
	~		

Então retomando a execução do loop for temos

Logo
$$*(100 + 0) = 2 * *(100 + 0);$$

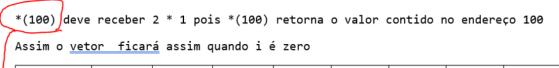
Assim podemos ler *(100) = 2 * *(100); da seguintes forma

$$*(100) = 2 * *(100);$$

ou *(100) = 2 * *(100);

Compilador atribua ao valor contido no endereço de memória 100 o resultado da multiplicação de 2 pelo valor contido no endereço de memória 100.

Assim temos



	Endereço	100-	103	104-107	108-111	112-115	116-119	120-123	124-127	128-131	132-135	136-139
V	Valor	2 -	$\overline{}$	2	3	4	5	6	7	8	9	10

Seguindo para i = 1 pois o for volta compara 1 < 10 e continua

IMPORTANTE DESTACAR AQUI O QUE JÁ COMENTEI NA AULA SOBRE ARITMÉTICA DE PONTEIROS

ESTE 100 + 1 NÃO DARÁ O VALOR 101!!!! ISSO É BEM IMPORTANTE!!!

ENDEREÇO DE MEMÓRIA 100 + 1 EM ARITMÉTICA DE PONTEIROS SERÁ 100 MAS O TAMANHO DO TIPO DO PONTEIRO

LOGO COMO INT TEM 4 BYTE ESTE 100 + 1 SERÁ NA REALIDADE 100 + 4 O QUE DARÁ ENDEREÇO 104!

E COM ISSO OBSERVE QUE CHAGAREMOS NO SEGUNDO ELEMENTO DO VETOR Numeros que tem endereço 104

Lembre-se também que como o ponteiro Array tem o endereço do vetor Numeros ele consegue alterar o valor deste vetor. É isso que chamamos de passagem de argumentos para a função por referência.

Pois foi enviado a função DuplicaVetor o endereço de memória do vetor Numeros que apesar ter 40 bytes de tamanho total seu endereço é O PRIMEIRO ENDEREÇO DE MEMÓRIA DELE! E que neste caso assumimos como 100.

									_	
Índice do vetor	0	1	2	3	4	5	6	7	8	9
Endereço	100-103	104-107	108-111	112-115	116-119	120-123	124-127	128-131	132-135	136-139
Valor	1	2	3	4	5	6	7	8	9	10

Assim *(100 + 1) = 2 * *(100 + 1); é na realidade Logo *(100 + 4) = 2 * *(100 + 4);

Ou Logo *(104) = 2 * *(104);

Ou Logo *(104) = 2 * 2;

Ou *(104) = 4;

Então podemos ler novamente que o valor contido no endereço de memória 104 deverá receber o valor 4 Assim o vetor fica assim

	Valor	2	4	3	4	5	6	7	8	9	10
indice do vector 0 1 2 5 4 5 0 7 8	Endereço	100-103	104-107	108-111	112-115	116-119	120-123	124-127	128-131	132-135	136-139
Índice do vetor A 1 2 2 3 4 5 6 7 9	Índice do vetor	0	1	2	3	4	5	6	7	8	9

Então se executarmos o for inteiro com i indo de zero até nove termos o seguinte

Valor 2 2 3 4 5 6 7 8 9 10	i	i <	Tam	Arra	y[i] = 2	* Array[:	i]					
i = 0				Arra	$y[0] = 2^*$	Array[0] ou					
*(100) = 2 * 1; aritmética desloca 4 bytes * i pois é ponteiro int *(100) = 2 // vá no endereço 100 e coloque o valor 2								9);				
*(100) = 2 // vá no endereço 100 e coloque o valor 2	i = 0	0 <	10 (V)									
Indice do vetor 0											onteiro :	int
Endereço 100-103 104-107 108-111 112-115 116-119 120-123 124-127 128-131 132-135 136-135				*(10	0) = 2 //	vá no ei	ndereço 1	l00 e col	oque o va	alor 2		
Valor 2 2 3 4 5 6 7 8 9 10	Índice do v	vetor	0	1	2	3	4	5	6	7	8	9
i i c Tam	Endere	ço	100-103	104-107	108-111	112-115	116-119	120-123	124-127	128-131	132-135	136-139
Array[1] = 2* Array[1] ou	Valor		2	2	3	4	5	6	7	8	9	10
* (Array + 1) = 2 * * (Array + 1);	i	i <	Tam	Arra	y[i] = 2	* Array[:	i]					
*(Array + 1) = 2 * *(Array + 1); *(100 + 1) = 2 * *(100 + 1); // aritmética de ponteiros!!! *(104) = 2 * 2; aritmética desloca 4 bytes vezes(*) i pois é ponteiro : *(104) = 4 // vá no endereço 104 e coloque o valor 4 *[ndice do vetor						Λ	•					
i = 1				Arra	y 1 = 2*	` Array 1	ou					
*(104) = 2 * 2; aritmética desloca 4 bytes vezes(*) i pois é ponteiro : *(104) = 4 // vá no endereço 104 e coloque o valor 4 fidice do vetor								L);				
*(104) = 4 // vá no endereço 104 e coloque o valor 4	i = 1	1 <	10 (V)	*(Ar	ray + 1)	= 2 * * (/	- Array + 1		tica de :	ponteiros	111	
Indice do vetor 0 1 2 3 4 5 6 7 8 9 Endereço 100-103 104-107 108-111 112-115 116-119 120-123 124-127 128-131 132-135 136-139 Valor 2 4 3 4 5 6 7 8 9 10 I i < Tam	i = 1	1 <	10 (V)	*(Ar *(10	ray + 1) 0 + 1) =	= 2 * *(/ 2 * *(100	- Array + 1 0 + 1); /	// aritmé				nteiro i
Valor 2 4 3 4 5 6 7 8 9 10 i i < Tam	i = 1	1 <	10 (V)	*(Ar *(10 *(10	ray + 1) 0 + 1) = 4) = 2 *	= 2 * *(/ 2 * *(100 2; aritme	- Array + 1 0 + 1); / ética des	// aritmé sloca 4 b	ytes veze	es(*) i p		nteiro i
i i C Tam				*(Ar *(10 *(10 *(10	ray + 1) 0 + 1) = 4) = 2 * 4) = 4 /	= 2 * *(/ 2 * *(100 2; aritmo // vá no 0	Array + 1 0 + 1); / ética des endereço	// aritmé sloca 4 b 104 e co	ytes veze loque o v	es(*) i p valor 4	ois é po	
Array[2] = 2* Array[2] ou *(Array + 2) = 2 * *(Array + 2); *(100 + 2) = 2 * *(100 + 2); // aritmética de ponteiros!!! *(108) = 2 * *(108); // este 2 é duas vezes 4 bytes logo 8 bytes *(108) = 2 * 3 *(108) = 6; // vá no endereço 108 e coloque o valor 6 findice do vetor 0 1 2 3 4 5 6 7 8 9 Endereço 100-103 104-107 108-111 112-115 116-119 120-123 124-127 128-131 132-135 136-139	Índice do v	vetor	0	*(Ar *(10 *(10 *(10	ray + 1) 0 + 1) = 4) = 2 * 4) = 4 /	= 2 * *(/ 2 * *(100 2; aritmo // vá no 0	Array + 1 0 + 1); / ética des endereço	// aritmé sloca 4 b 104 e co	ytes veze loque o	es(*) i p valor 4	ois é por	
Array[2] = 2* Array[2] ou *(Array + 2) = 2 * *(Array + 2); *(100 + 2) = 2 * *(100 + 2); // aritmética de ponteiros!!! *(108) = 2 * *(108); // este 2 é duas vezes 4 bytes logo 8 bytes *(108) = 2 * 3 *(108) = 6; // vá no endereço 108 e coloque o valor 6 findice do vetor 0 1 2 3 4 5 6 7 8 9 Endereço 100-103 104-107 108-111 112-115 116-119 120-123 124-127 128-131 132-135 136-139	Índice do v	vetor	0 100-103	*(Ar *(10) *(10) *(10) 1 104-107	ray + 1) 0 + 1) = 4) = 2 * 4) = 4 / 2 108-111	= 2 * *() 2 * *(100 2; aritme // vá no 0	Array + 1 0 + 1); / ética des endereço 4 116-119	// aritmésloca 4 b 104 e co 5 120-123	ytes veze loque o v	es(*) i p valor 4 7 128-131	8 132-135	9 136-139
*(Array + 2) = 2 * *(Array + 2); *(100 + 2) = 2 * *(100 + 2); // aritmética de ponteiros!!! *(108) = 2 * *(108); // este 2 é duas vezes 4 bytes logo 8 bytes *(108) = 2 * 3 *(108) = 6; // vá no endereço 108 e coloque o valor 6 findice do vetor 0 1 2 3 4 5 6 7 8 9 Endereço 100-103 104-107 108-111 112-115 116-119 120-123 124-127 128-131 132-135 136-139	Índice do v Endere Valor	vetor	0 100-103	*(Ar *(10 *(10 *(10 1 104-107	ray + 1) 0 + 1) = 4) = 2 * 4) = 4 / 2 108-111 3	= 2 * *(100 2 * *(100 2; aritmo // vá no 6 3 112-115 4	Array + 1 0 + 1); / ética des endereço 4 116-119 5	// aritmésloca 4 b 104 e co 5 120-123	ytes veze loque o v	es(*) i p valor 4 7 128-131	8 132-135	9 136-139
*(100 + 2) = 2 * *(100 + 2); // aritmética de ponteiros!!! *(108) = 2 * *(108); // este 2 é duas vezes 4 bytes logo 8 bytes *(108) = 2 * 3 *(108) = 6; // vá no endereço 108 e coloque o valor 6 findice do vetor 0 1 2 3 4 5 6 7 8 9 Endereço 100-103 104-107 108-111 112-115 116-119 120-123 124-127 128-131 132-135 136-139	Índice do v Endere Valor	vetor	0 100-103	*(Ara	ray + 1) 0 + 1) = 4) = 2 * 4) = 4 / 2 108-111 3 y[i] = 2	= 2 * *(100 2 * *(100 2; aritme // vá no 0 3 112-115 4 * Array[:	Array + 1 0 + 1); / ética des endereço 4 116-119 5	// aritmésloca 4 b 104 e co 5 120-123	ytes veze loque o v	es(*) i p valor 4 7 128-131	8 132-135	9 136-139
*(108) = 2 * *(108); // este 2 é duas vezes 4 bytes logo 8 bytes *(108) = 2 * 3 *(108) = 6; // vá no endereço 108 e coloque o valor 6 indice do vetor 0 1 2 3 4 5 6 7 8 9 Endereço 100-103 104-107 108-111 112-115 116-119 120-123 124-127 128-131 132-135 136-139	Índice do v Endere Valor	vetor	0 100-103	*(Ar *(10) *	ray + 1) 0 + 1 = 2 + 4 0 + 1 = 2 + 4 0 + 1 = 4 + 7 0 + 1 = 4 +	= 2 * *(100 2 * *(100 2; aritme // vá no 0 3 112-115 4 * Array[:	Array + 1 2 + 1); / ética des endereço 4 116-119 5 i] ou	// aritmésloca 4 b 104 e co 5 120-123	ytes veze loque o v	es(*) i p valor 4 7 128-131	8 132-135	9 136-139
*(108) = 2 * 3 *(108) = 6; // vá no endereço 108 e coloque o valor 6 Índice do vetor 0 1 2 3 4 5 6 7 8 9 Endereço 100-103 104-107 108-111 112-115 116-119 120-123 124-127 128-131 132-135 136-139	índice do v Endere Valor	vetor ÇO	0 100-103 2 Tam	*(Ara *(10)	ray + 1) 0 + 1) = 4) = 2 * 4) = 4 / 2 108-111 3 y[i] = 2 y[2] = 2* ray + 2)	= 2 * *() 2 * *(100) 2; aritme // vá no e 3 112-115 4 * Array[: 4 Array[: 5 Array[2] = 2 * *()	Array + 1 2 + 1); / ética des endereço 4 116-119 5 i] ou Array + 2	// aritmésloca 4 b 104 e co 5 120-123 6	ytes veze loque o v 6 124-127 7	es(*) i p valor 4 7 128-131 8	8 132-135 9	9 136-139
*(108) = 6; // vá no endereço 108 e coloque o valor 6 Indice do vetor 0 1 2 3 4 5 6 7 8 9 Endereço 100-103 104-107 108-111 112-115 116-119 120-123 124-127 128-131 132-135 136-139	índice do v Endere Valor	vetor ÇO	0 100-103 2 Tam	*(Ar *(10) *	ray + 1) 0 + 1) = 2 * 4) = 4 / 2 108-111 3 y[i] = 2 y[2] = 2* ray + 2) 0 + 2) =	= 2 * *() 2 * *(100) 2; aritment // vá no entre signal	Array + 1 2 + 1); / ética des endereço 4 116-119 5 i] ou Array + 2 2 + 2); /	// aritmés loca 4 b 104 e co 5 120-123 6	ytes veze loque o v 6 124-127 7	valor 4 7 128-131 8	8 132-135 9	9 136-139 10
Índice do vetor 0 1 2 3 4 5 6 7 8 9 Endereço 100-103 104-107 108-111 112-115 116-119 120-123 124-127 128-131 132-135 136-139	índice do v Endere Valor	vetor ÇO	0 100-103 2 Tam	*(Ar *(10) *(10) *(10) *(10) *(10) Arra *(Ar *(10) *(10)	ray + 1) 0 + 1) = 4) = 2 * 4) = 4 / 2 108-111 3 y[i] = 2 y[2] = 2* ray + 2) 0 + 2) = 8) = 2 *	= 2 * *(/ 2 * *(100) 2; aritmon // vá no o 3 112-115 4 * Array[: 4 Array[: 2 * *(100) *(108); /	Array + 1 2 + 1); / ética des endereço 4 116-119 5 i] ou Array + 2 2 + 2); /	// aritmés loca 4 b 104 e co 5 120-123 6	ytes veze loque o v 6 124-127 7	valor 4 7 128-131 8	8 132-135 9	9 136-139 10
	índice do v Endere Valor	vetor ÇO	0 100-103 2 Tam	*(Ar *(10) *(10) *(10) *(10) * Arra Arra *(Ar *(10) *(10)	ray + 1) 0 + 1) = 2 * 4) = 4 / 2 108-111 3 y[i] = 2 * ray + 2) = 2 * 8	= 2 * *(/ 2 * *(100 2; aritme // vá no e 112-115 4 * Array[2 = 2 * *(/ 2 * *(100); / 3	Array + 1 2 + 1); / ética des endereço 4 116-119 5 i] ou Array + 2 2 + 2); / // este 2	// aritmésloca 4 b 104 e co 5 120-123 6 2); // aritmé	ytes vezes 4 by ytes vezes 4 b	ponteiros	8 132-135 9	9 136-139 10
· · · · · · · · · · · · · · · · · · ·	i = 2	i <	0 100-103 2 Tam	*(Ar *(10 *(10) *(10) *(10) *(10) *(10) *(10) *(10) *(10)	ray + 1) 0 + 1) = 4) = 2 * 4) = 4 / 2 108-111 3 y[i] = 2 y[2] = 2* ray + 2) = 8) = 2 * 8) = 2 * 8) = 6; /	= 2 * *() 2 * *(100) 2; aritmo // vá no 6 * Array[2] = 2 * *() 2 * *(100) *(108); 3 // vá no 6	Array + 1 2 + 1); / ética des endereço 4 116-119 5 i] ou Array + 2 2 + 2); / // este 2	7/ aritmésloca 4 b 104 e co 5 120-123 6 2); 7/ aritmé 2 é duas 108 e co	ytes vezes de proposes de prop	ponteiros bytes log	8 132-135 9	9 136-139 10
	i = 2	i <	0 100-103 2 Tam 10 (V)	*(Ar *(10 *(10 *(10) *(10) *(10) *(10) *(10) *(10) *(10) *(10)	ray + 1) 0 + 1) = 4) = 2 * 4) = 4 / 2 108-111 3 y[i] = 2 y[2] = 2* ray + 2) 0 + 2) = 8) = 2 * 8) = 2 * 8) = 6; /	= 2 * *(/ 2 * *(100) 2; aritmo // vá no 6 3 112-115 4 * Array[2 = 2 * *(/ 2 * *(100) *(108); / 3 // vá no 6	Array + 1 2 + 1); / ética des endereço 4 116-119 5 i] ou Array + 2 2 + 2); / // este 2 endereço	7/ aritmé sloca 4 b 104 e co 5 120-123 6 2); 7/ aritmé 2 é duas 108 e co 5	ytes vezes loque o vezes 4 loque o vezes 4 loque o vezes 6	ponteiros bytes log	8 132-135 9 !!! to 8 bytes	9 136-139 10

Ĺ	i <	Tam	Arra	y[i] = 2	* Array[i	i]						
i = 3	3 <	: 10 (V)	*(Ar	Array[3] = 2* Array[3] ou *(Array + 3) = 2 * *(Array + 3); *(100 + 3) = 2 * *(100 + 3); // aritmética de ponteiros!!! *(112) = 2 * *(112); // este 3 é três vezes 4 bytes logo 12 bytes *(112) = 2 * 4 // aritmética desloca 4 bytes * i pois é ponteiro int *(112) = 8; // vá no endereço 112 e coloque o valor 8								
			*(11	2) = 2 *	4 // ar	itmética	desloca	4 bytes *	ˈi pois ˈ			
Índice do	vetor	0	*(11	2) = 2 *	4 // ar	itmética	desloca	4 bytes *	ˈi pois ˈ			
Índice do		0 100-103	*(11	2) = 2 *	4 // ar	itmética	desloca	4 bytes * loque o \	ˈi pois ˈ			

i	i	< Tam	Arra	y[i] = 2	* Array[:	i]					
			Arra	y[4] = 2*	Array[4]	ou					
				ray + 4)							
i = 4	4	< 10 (V)		0 + 4) =							
				6) = 2 * 6) = 2 *							
				6) = 2 6) = 10;						e poncein	O IIIC
Índice do v	vetor	0	1	2	3	4	5	6	7	8	9
		100-103	104-107	108-111	112-115	116-119	120-123	124-127	128-131	132-135	136-139
Endere	2Ç0										
Valor		2	4	6	8	10	6	7	8	9	10
i	i	< Tam		y[i] = 2							
				y[5] = 2* ray + 5)			١.				
				(0 + 5) =				tica de n	onteiros	111	
i = 5	5	< 10 (V)		0) = 2 *							es
				0) = 2 *							
			*(12	0) = 12;	// vá no	endereço	120 e c	oloque o	valor 12		
Índice do v	vetor	0	1	2	3	4	5	6	7	8	9
Endere	2ÇO	100-103	104-107	108-111	112-115	116-119	120-123	124-127	128-131	132-135	136-139
Valor	-	2	4	6	8	10	12	7	8	9	10
i	j	< Tam		y[i] = 2	_		I	I .			
	†			$y[6] = 2^*$							
				ray + 6)			; ();				
i = 6	ے	< 10 (V)	*(10	0 + 6) =	2 * *(100	+ 6); /	/ aritmé				
1 - 0	"	(10 (V)		4) = 2 *							
				4) = 2 *						é ponteir	o int
			*(12	4) = 14;	// vá no	endereço	124 e c	oloque o	valor 14		
Índice do v	vetor	0	1	2	3	4	5	6	7	8	9
Endere	2ÇO	100-103	104-107	108-111	112-115	116-119	120-123	124-127	128-131	132-135	136-139
Valor											
101		2	4	6	8	10	12	14	8	9	10
V 0 1 0 1		2	4	6	8	10	12	14	8	9	10
i	i <	Tam		/[i] = 2			12	14	8	9	10
	i <		Array	/[i] = 2 /[7] = 2*	* Array[i Array[7]	.] ou		14	8	9	10
	i <		Array Array *(Arr	y[i] = 2 y[7] = 2* y[7] = 4	* Array[i Array[7] = 2 * *(A	.] ou array + 7);				10
i		Tam	Array Array *(Arr	y[i] = 2 y[7] = 2* y[7] = 2* y[7] = 2* y[7] = 2*	* Array[i Array[7] = 2 * *(A 2 * *(100	ou rray + 7); / aritmé	tica de p	oonteiros	!!!	
Ĺ			Array Array *(Arr *(100	y[i] = 2 $y[7] = 2^*$ $y[7] = 2^*$ $y[7] = 2^*$ $y[7] = 2^*$	* Array[i Array[7] = 2 * *(A 2 * *(100 *(128); /	.] ou krray + 7 (+ 7); / // este 7); / aritmé é sete	tica de p	oonteiros oytes log	!!! o 28 byt	es
Ĺ		Tam	Array Array *(Arr *(100 *(128 *(128	y[i] = 2 y[7] = 2* ray + 7) = 3 ray + 7) = 3	* Array[i Array[7] = 2 * *(A 2 * *(100 *(128); / 8 // ari	ou array + 7 (+ 7); / (/ este 7); / aritmé é sete desloca	tica de p vezes 4 b 4 bytes *	oonteiros oytes log	!!! o 28 byt é pontei	es
i = 7	7 <	Tam 10 (V)	Array Array *(Arr *(100 *(128 *(128 *(128	y[i] = 2 y[7] = 2* ay + 7) = 2* ay + 7 = 3 ay + 7 = 4 ay + 7 = 4 a	* Array[i Array[7] = 2 * *(A 2 * *(100 *(128); / 8 // ari // vá no	ou rray + 7 + 7); / // este 7 tmética endereço); / aritmé é sete desloca 128 e c	tica de p vezes 4 b 4 bytes *	oonteiros oytes log i pois valor 16	!!! o 28 byt é pontei	es ro int
i = 7	7 <	Tam 10 (V)	Array Array *(Arr *(100 *(128 *(128	y[i] = 2 y[7] = 2* ray + 7) = 3 + 7) = 8) = 2 * 8) = 2 * 8) = 16;	* Array[i Array[7] = 2 * *(A 2 * *(100 *(128); / 8 // ari // vá no	ou rray + 7) + 7); / // este 7 tmética endereço); / aritmé é sete desloca 128 e co	tica de p vezes 4 t 4 bytes * oloque o	ponteiros pytes log i pois valor 16	!!! o 28 byt é pontei	es ro int
i = 7	7 <	Tam 10 (V)	Array Array *(Arr *(100 *(128 *(128 *(128	y[i] = 2 y[7] = 2* ay + 7) = 2* ay + 7 = 3 ay + 7 = 4 ay + 7 = 4 a	* Array[i Array[7] = 2 * *(A 2 * *(100 *(128); / 8 // ari // vá no	ou rray + 7 + 7); / // este 7 tmética endereço); / aritmé é sete desloca 128 e c	tica de p vezes 4 b 4 bytes *	oonteiros oytes log i pois valor 16	!!! o 28 byt é pontei	es ro int
i = 7	7 <	Tam 10 (V)	Array Array *(Arr *(100 *(128 *(128	y[i] = 2 y[7] = 2* ray + 7) = 3 + 7) = 8) = 2 * 8) = 2 * 8) = 16;	* Array[i Array[7] = 2 * *(A 2 * *(100 *(128); / 8 // ari // vá no	ou rray + 7) + 7); / // este 7 tmética endereço); / aritmé é sete desloca 128 e co	tica de p vezes 4 t 4 bytes * oloque o	ponteiros pytes log i pois valor 16	!!! o 28 byt é pontei	es ro int
i = 7 Indice do verence con v	7 <	Tam 10 (V) 0 100-103	Array Array *(Arr *(100 *(128 *(128 *(128 1 104-107	y[i] = 2 y[7] = 2* ray + 7) = 3 + 7) = 3 + 7 = 2 3 + 2 * 3 + 2 * 3 + 2 * 3 + 2 * 3 + 16; 2 + 108-111	* Array[i Array[7] = 2 * *(A 2 * *(100 *(128); / 8 // ari // vá no 3 112-115	ou (rray + 7); / (/ este 7) tmética endereço 4 116-119); / aritmé é sete desloca 128 e co	tica de p vezes 4 t 4 bytes * oloque o 6 124-127	oonteiros bytes log i pois valor 16 7 128-131	!!! o 28 byte é pontei 8 132-135	es ro int 9 136-139
i = 7 Indice do verence con v	7 <	Tam 10 (V) 0 100-103	Array Array *(Arr *(100 *(128 *(128 *(128 1 104-107 4 Array	y[i] = 2 y[7] = 2* ray + 7) = 3 + 7) = 3 = 2 * 3 = 2 * 3 = 16; 2 108-111 6	* Array[i Array[7] = 2 * *(A 2 * *(100 *(128); / 8 // ari // vá no 3 112-115 8 * Array[i	ou (rray + 7); / (/ este 7) tmética endereço 4 116-119 10); / aritmé é sete desloca 128 e co	tica de p vezes 4 t 4 bytes * oloque o 6 124-127	oonteiros bytes log i pois valor 16 7 128-131	!!! o 28 byte é pontei 8 132-135	es ro int 9 136-139
i = 7 Indice do verende reç Valor	7 <	Tam 10 (V) 0 100-103	Array Array *(Arr *(100 *(128 *(128 *(128 1 104-107 4 Array Array	y[i] = 2 y[7] = 2* (7) = 2* (2) + 7) = 3 (3) = 2* (3) = 2* (3) = 16; (4) = 10* (5) = 10* (6)	* Array[i Array[7] = 2 * *(A 2 * *(100 *(128); / 8 // ari // vá no 3 112-115 8 * Array[i Array[8]	ou (rray + 7); / (/ este 7) tmética endereço 4 116-119 10); / aritmé é sete desloca 128 e co 5 120-123	tica de p vezes 4 t 4 bytes * oloque o 6 124-127	oonteiros bytes log i pois valor 16 7 128-131	!!! o 28 byte é pontei 8 132-135	es ro int 9 136-139
i = 7 Indice do verence con v	7 <	Tam 10 (V) 0 100-103	Array Array *(Arr *(100 *(128 *(128 *(128 *(128 1 104-107 4 Array Array *(Arr	y[i] = 2 y[7] = 2* ray + 7) = 3 + 7) = 3 = 2 * 3 = 2 * 3 = 16; 2 108-111 6 y[i] = 2 y[8] = 2*	* Array[i Array[7] = 2 * *(A 2 * *(100 *(128); / 8 // ari // vá no 3 112-115 8 * Array[i Array[8] = 2 * *(A	ou array + 7 7 + 7); / 7 / este 7 atmética endereço 4 116-119 10 ou array + 8); / aritmé é sete desloca 128 e co 5 120-123	tica de p vezes 4 b 4 bytes * oloque o 6 124-127	oonteiros bytes log i pois valor 16 7 128-131	!!! o 28 byte é pontei 8 132-135 9	es ro int 9 136-139
= 7 Indice do ve Endereç Valor	7 <	Tam 10 (V) 0 100-103	Array Array *(Arr *(100 *(128 *(128 *(128 *(128 1 104-107 4 Array Array *(Arr *(100	y[i] = 2 y[7] = 2* ray + 7) = 3 + 7 = 3 = 2 * 3 + 2 = 16; 2	* Array[i Array[i Array[7] = 2 * *(A 2 * *(100 *(128); / 8	ou (rray + 7); / // este 7 tmética endereço 4 116-119 10 ou (rray + 8 () + 8); /); / aritmé é sete desloca 128 e co 5 120-123 12); / aritmé	tica de p vezes 4 b 4 bytes * oloque o 6 124-127 14	oonteiros oytes log i pois valor 16 7 128-131 16	!!! o 28 byte é pontei 8 132-135 9	es ro int 9 136-139 10
= 7 Indice do ve Endereç Valor	7 <	Tam 10 (V) 0 100-103 2 Tam	Array Array *(Arr *(100 *(128 *(128 *(128 1104-107 4 Array Array *(Arr *(100 *(132	y[i] = 2 y[7] = 2* ray + 7) = 3 3 + 7) = 3 3 = 2 * 3 = 16; 2 108-111 6 y[i] = 2 y[8] = 2* ray + 8) = 2 2 + 8) = 2	* Array[i Array[7] = 2 * *(A 2 * *(100 *(128); / 8 // ari // vá no 3 112-115 8 * Array[i Array[8] = 2 * *(A 2 * *(100 *(132); /	ou (rray + 7); / (/ este 7); / (/ este 7); / (/ este 7); / (/ este 8); / (/ este 8); / (/ este 8); /); / aritmé é sete desloca 128 e co 5 120-123 12); / aritmé é oito	tica de p vezes 4 t 4 bytes * oloque o 6 124-127 14 tica de p	oonteiros oytes log i pois valor 16 7 128-131 16 oonteiros oytes log	!!! o 28 byto é pontei 8 132-135 9	es ro int 9 136-139 10
. = 7 Índice do ve Endereç Valor	7 <	Tam 10 (V) 0 100-103 2 Tam	Array (108	y[i] = 2 y[7] = 2* ray + 7) = 3 3	* Array[i Array[i Array[7] = 2 * *(A 2 * *(100 *(128); / 8	ou array + 7) + 7); / / este 7 tmética endereço 4 116-119 10 ou array + 8) + 8); / / este 8 tmética); / aritmé é sete desloca 128 e co 5 120-123 12); / aritmé é oito desloca	tica de provezes 4 to de provezes de provezes 4 to de pro	oonteiros bytes log i pois valor 16 7 128-131 16 conteiros bytes log i pois	!!! o 28 byto é pontei 8 132-135 9	es ro int 9 136-139 10
indice do verende de la companya del companya de la companya del companya de la companya del companya de la companya de la companya de la companya del companya de la companya della companya de la companya de la companya della compa	7 < etc	Tam 10 (V) 0 100-103 2 Tam 10 (V)	Array Array *(Arr *(100 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(132 *(132 *(132 *(132) *(132) *(132) *(132)	y[i] = 2 y[7] = 2* (7) = 2* (3) + 7) = 3 (3) = 2* (3) = 2* (3) = 16; (4) = 2* (5) = 2* (6) = 2*	* Array[i Array[i Array[7] = 2 * *(A 2 * *(100 *(128); / 8	ou (rray + 7); / / este 7 tmética endereço 4 116-119 10 ou (rray + 8 0 + 8); / / este 8 tmética endereço); / aritmé é sete desloca 128 e co 5 120-123 12); / aritmé é oito desloca 132 e co	tica de prezes 4 bytes * coloque o 6 124-127 14 tica de prezes 4 bytes * coloque o	oonteiros oytes log i pois valor 16 7 128-131 16 oonteiros oytes log i pois valor 18	!!! o 28 byt é pontei 8 132-135 9	9 136-139 10
i = 7 Indice do ve Endereç Valor i = 8	7 < etcr i < 8 <	Tam 10 (V) 0 100-103 2 Tam 10 (V)	Array Array *(Arr *(100 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(132 *(132 *(132 *(132) *(13	y[i] = 2 y[7] = 2* ray + 7) = 3 3 + 7) = 3 3 + 2 * 3 = 2 * 3 = 16; 2 108-111 6 y[i] = 2 y[8] = 2* ray + 8) = 2 2 + 8) = 2 2 + 2 = 2 * 2 + 2 = 2 * 2 = 18;	* Array[i Array[7] = 2 * *(A 2 * *(100 *(128); / 8	ou (rray + 7); / / este 7 tmética endereço 4 116-119 10 ou (rray + 8); / / este 8 tmética endereço); / aritmé é sete desloca 128 e co 5 120-123 12); / aritmé é oito desloca 132 e co	tica de provezes 4 bytes * coloque o 6 124-127 14 tica de provezes 4 bytes * coloque o	oonteiros ytes log i pois valor 16 7 128-131 16 oonteiros ytes log i pois valor 18	!!! o 28 byto é pontei 8 132-135 9	9 136-139 10
indice do ve Endereç Valor Indice do ve Endereç Endereç	7 < etcr i < 8 <	Tam 10 (V) 0 100-103 2 Tam 10 (V)	Array Array *(Arr *(106 *(128 *(128 *(128 104-107 4 Array Array *(Arr *(106 *(132 *(132 *(132 1104-107	y[i] = 2 y[7] = 2* ray + 7) = 3 + 7) = 3 + 7) = 3 + 7) = 3 2 * 3 16; 2 108-111 6 y[i] = 2 y[8] = 2* ray + 8) = 2 + 8) = 2 2 * 2 2 * 3 3 * 3	* Array[i Array[i Array[7] = 2 * *(A 2 * *(100 *(128); / 8 // ari // vá no 3 112-115 8 * Array[i Array[8] = 2 * *(A 2 * *(100 *(132); / 9 // ari // vá no 3 112-115	ou array + 7 7 + 7); / 6 / este 7 tmética endereço 4 116-119 10 ou array + 8 0 + 8); / 6 / este 8 tmética endereço 4 116-119); / aritmé é sete desloca 128 e co 5 120-123 12); / aritmé é oito desloca 132 e co 5 120-123	tica de prezes 4 te d	oonteiros bytes log i pois valor 16 7 128-131 16 conteiros bytes log i pois valor 18 7 128-131	!!! o 28 byte é pontei 8 132-135 9 !!! o 32 byte é pontei	9 136-139 10 es ro int
indice do verende de la companya del companya del companya de la companya del companya de la companya de la companya del companya de la companya del companya del companya de la companya de la companya de la companya de la companya del compan	7 < etcr	Tam 10 (V) 0 100-103 2 Tam 10 (V)	Array Array *(Arr *(106 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(132 Array *(Array *(Array *(132 *(132 *(132 *(132 1 104-107 4	y[i] = 2 y[7] = 2* ray + 7) = 3 + 7 = 3 + 7 = 2 3 + 7 = 3 + 7 = 3 = 2 * 3 = 16; 2	* Array[i Array[i Array[7] = 2 * *(A 2 * *(100 *(128); / 8	ou (rray + 7); / / este 7 tmética endereço 4 116-119 10 ou (rray + 8); / / este 8 tmética endereço 4 116-119 10); / aritmé é sete desloca 128 e co 5 120-123 12); / aritmé é oito desloca 132 e co	tica de provezes 4 bytes * coloque o 6 124-127 14 tica de provezes 4 bytes * coloque o	oonteiros ytes log i pois valor 16 7 128-131 16 oonteiros ytes log i pois valor 18	!!! o 28 byto é pontei 8 132-135 9	9 136-139 10
i = 7 Indice do ve Endereç Valor i = 8 Indice do ve Endereç	7 < etcr	Tam 10 (V) 0 100-103 2 Tam 10 (V)	Array Array *(Arr *(106 *(128 *(128 *(128 *(128 1 104-107 4 Array Array *(Arr *(106 *(132 *(132 *(132 1 104-107 4 Array Array	y[i] = 2 y[7] = 2* ray + 7) = 3 + 7) = 3 + 7) = 3 = 2 * 3 = 16; 2 108-111 6 y[i] = 2 y[8] = 2* ray + 8) = 2 + 8) = 2 + 8) = 2 + 8 = 2 * 2 + 8 = 2 * 3 + 8 = 2 * 4 + 8 = 2 * 5 + 8 = 2 * 5 + 8 = 2 * 5 + 8 = 2 * 6 + 8 = 2	* Array[i Array[i Array[7] = 2 * *(A 2 * *(100 *(128); /8 8	ou (rray + 7); / (/ este 7 tmética endereço 4 116-119 10 ou (rray + 8); / (/ este 8 tmética endereço 4 116-119 10); / aritmé é sete desloca 128 e co 5 120-123 12); / aritmé é oito desloca 132 e co 5 120-123	tica de prezes 4 te d	oonteiros bytes log i pois valor 16 7 128-131 16 conteiros bytes log i pois valor 18 7 128-131	!!! o 28 byte é pontei 8 132-135 9 !!! o 32 byte é pontei	9 136-139 10 es ro int
i = 7 Indice do verende e e e e e e e e e e e e e e e e e e	7 < etcr	Tam 10 (V) 0 100-103 2 Tam 10 (V)	Array Array *(Arr *(106 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(132 *(132 *(132 *(132 *(132 *(132 Array Array Array Array Array Array Array	y[i] = 2 y[7] = 2* ray + 7) = 3 + 7 = 3 + 7 = 3 = 2 * 3 + 7 = 3 = 16; 2	* Array[i Array[i Array[7] = 2 * *(A 2 * *(100 *(128); /8 8	ou (rray + 7); / / este 7 tmética endereço 4 116-119 10 ou (rray + 8); / / este 8 tmética endereço 4 116-119 10); / aritmé é sete desloca 128 e co 5 120-123 12); / aritmé é oito desloca 132 e co 5 120-123 12	tica de prezes 4 te d	oonteiros bytes log i pois valor 16 7 128-131 16 conteiros bytes log i pois valor 18 7 128-131	!!! o 28 byte é pontei 8 132-135 9 !!! o 32 byte é pontei	9 136-139 10 es ro int
i = 7 Indice do verende e e e e e e e e e e e e e e e e e e	7 < etcr	Tam 10 (V) 0 100-103 2 Tam 10 (V)	Array Array *(Arr *(100 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(128 Array Array *(Arr *(100 *(132 *(132 *(132 *(132 Array	y[i] = 2 y[7] = 2* ray + 7) = 3 + 7) = 3 + 7) = 3 = 2 * 3 = 16; 2 108-111 6 y[i] = 2 y[8] = 2* ray + 8) = 2 + 8) = 2 + 8) = 2 + 8 = 3 + 8 = 3 + 8 = 3 + 8 = 3 + 8 = 4	* Array[i Array[i Array[7] = 2 * *(A 2 * *(100 *(128); / 8	ou (rray + 7); / / este 7 tmética endereço 4 116-119 10 ou (rray + 8); / / este 8 tmética endereço 4 116-119 10 ou (rray + 8); / / este 8); / aritmé é sete desloca 128 e co 5 120-123 12); / aritmé é oito desloca 132 e co 5 120-123 12	tica de prezes 4 bytes * coloque o 6 124-127 14 tica de prezes 4 bytes * coloque o 6 124-127 14	oonteiros ytes log i pois valor 16 7 128-131 16 oonteiros ytes log i pois valor 18 7 128-131 16	!!! o 28 byte é pontei 8 132-135 9 !!! o 32 byte é pontei 8 132-135 18	9 136-139 10 es ro int
i = 7 Indice do verente de la companya del companya del companya de la companya del companya de la companya de la companya del companya de la companya del companya de la companya de la companya de la companya de la companya del companya de la companya de la companya de la companya de la companya del	7 < etcr cco i < 8 < i column 1 i < i column 2 i < i column 2 i < i column 2 i < i column 3 i <	Tam 10 (V) 0 100-103 2 Tam 10 (V) 100-103 2 Tam	Array Array *(Arr *(100 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(132	y[i] = 2 y[7] = 2* ray + 7) = 3 3 + 7) = 3 3 + 2 * 3 + 2 * 3 + 2 * 3 + 2 + 2 + 2 + 2 + 2 + 3 + 3 + 2 + 3 + 3 + 2 + 4 + 3 + 2 + 5 + 6 + 2 + 7 + 7 + 7 + 2 + 8 + 7 + 2 + 8 + 8 + 3 + 8 + 3 + 8 + 8 + 3 + 8 +	* Array[i Array[i Array[7] = 2 * *(A 2 * *(100 *(128); / 8	ou (rray + 7); / / este 7 tmética endereço 4 116-119 10 ou (rray + 8); / / este 8 tmética endereço 4 116-119 10 ou (rray + 9); /); / aritmé é sete desloca 128 e co 5 120-123 12); / aritmé é oito desloca 132 e co 5 120-123 12); / aritmé / aritmé	tica de prezes 4 bytes * coloque o 6 124-127 14 tica de prezes 4 bytes * coloque o 6 124-127 14	oonteiros ytes log i pois valor 16 7 128-131 16 oonteiros ytes log i pois valor 18 7 128-131 16	!!! o 28 byte é pontei 8 132-135 9 !!! o 32 byte é pontei 8 132-135 18	9 136-139 10 es ro int 9 136-139 20
i = 7 Indice do verende e e e e e e e e e e e e e e e e e e	7 < etcr cco i < 8 < i column 1 i < i column 2 i < i column 2 i < i column 2 i < i column 3 i <	Tam 10 (V) 0 100-103 2 Tam 10 (V)	Array Array *(Arr *(100 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(132	y[i] = 2 y[7] = 2* ray + 7) = 3 3 + 7) = 3 3 + 2 * 3 = 2 * 3 = 16; 2 108-111 6 y[i] = 2 y[8] = 2* ray + 8) = 2 2 + 8) = 2 2 + 8 = 2 3 + 8 = 2 4 + 8 = 2 5 + 8 = 2 5 + 8 = 2 6 + 9 = 2 6 + 9 = 2 6 + 9 = 2	* Array[i Array[i Array[7] = 2 * *(A 2 * *(100 *(128); / 8	ou (rray + 7); / / este 7 tmética endereço 4 116-119 10 ou (rray + 8); / / este 8 tmética endereço 4 116-119 10 ou (rray + 9); / / este 9); / aritmé é sete desloca 128 e co 5 120-123 12); / aritmé é oito desloca 132 e co 5 120-123 12); / aritmé é nove	tica de provezes 4 bytes * coloque o 6 124-127 14 tica de provezes 4 bytes * coloque o 6 124-127 14 tica de provezes 4 bytes *	oonteiros ytes log i pois valor 16 7 128-131 16 oonteiros ytes log i pois valor 18 7 128-131 16	!!! o 28 byte é pontei 8 132-135 9 !!! o 32 byte é pontei 8 132-135 18	9 136-139 10 9 136-139 20 20
i = 7 Indice do verente de la companya del companya del companya de la companya del companya de la companya de la companya del companya de la companya del companya de la companya de la companya de la companya de la companya del companya de la companya de la companya de la companya de la companya del	7 < etcr cco i < 8 < i column 1 i < i column 2 i < i column 2 i < i column 2 i < i column 3 i <	Tam 10 (V) 0 100-103 2 Tam 10 (V) 100-103 2 Tam	Array Array *(Arr *(100 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(132	y[i] = 2 y[7] = 2* ray + 7) = 3 3 + 7) = 3 3 + 2 * 3 + 2 * 3 + 3 = 16; 2 108-111 6 y[i] = 2 y[8] = 2* ray + 8) = 2 2 + 8 = 2 3 + 8 = 2 4 + 8 = 2 5 + 8 = 2 5 + 9 = 2* 6 + 9 = 2 * 6 + 9 = 2 * 6 + 9 = 2 *	* Array[i Array[i] Array[7] = 2 * *(A 2 * *(100 *(128); / 8	ou (rray + 7); / / este 7 tmética endereço 4 116-119 10 ou (rray + 8); / / este 8 tmética endereço 4 116-119 10 ou (rray + 9); / / este 9 itmética); / aritmé é sete desloca 128 e co 5 120-123 12); / aritmé é oito desloca 132 e co 5 120-123 12); / aritmé é nove desloca	tica de prezes 4 to d	oonteiros ytes log i pois valor 16 7 128-131 16 oonteiros ytes log i pois valor 18 7 128-131 16 oonteiros ytes log i pois valor log i pois valor i pois	!!! o 28 byte é pontei 8 132-135 9 !!! o 32 byte é pontei 8 132-135 18	9 136-139 10 9 136-139 20 20
indice do verende de la companya del companya del companya de la companya del companya de la companya de la companya del companya de la companya del companya del companya de la companya del co	7 < etcr cco i < 8 < i column 1 i < i column 2 i < i column 2 i < i column 2 i < i column 3 i <	Tam 10 (V) 0 100-103 2 Tam 10 (V) 100-103 2 Tam	Array Array *(Arr *(100 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(128 *(132	y[i] = 2 y[7] = 2* ray + 7) = 3 3 + 7) = 3 3 + 2 * 3 = 2 * 3 = 16; 2 108-111 6 y[i] = 2 y[8] = 2* ray + 8) = 2 2 + 8) = 2 2 + 8 = 2 3 + 8 = 2 4 + 8 = 2 5 + 8 = 2 5 + 8 = 2 6 + 9 = 2 6 + 9 = 2 6 + 9 = 2	* Array[i Array[i] Array[7] = 2 * *(A 2 * *(100 *(128); / 8	ou (rray + 7); / / este 7 tmética endereço 4 116-119 10 ou (rray + 8); / / este 8 tmética endereço 4 116-119 10 ou (rray + 9); / / este 9 itmética); / aritmé é sete desloca 128 e co 5 120-123 12); / aritmé é oito desloca 132 e co 5 120-123 12); / aritmé é nove desloca	tica de prezes 4 to d	oonteiros ytes log i pois valor 16 7 128-131 16 oonteiros ytes log i pois valor 18 7 128-131 16 oonteiros ytes log i pois valor log i pois valor i pois	!!! o 28 byte é pontei 8 132-135 9 !!! o 32 byte é pontei 8 132-135 18	9 136-139 10 9 136-139 20 20

•	1	2	3	4	5	6	7	8	9
100-103	104-107	108-111	112-115	116-119	120-123	124-127	128-131	132-135	136-139
2	4	6	8	10	12	14	16	18	20
(10 (F)	SAI [00 LOOP FO	OR!						
	2	2 4	2 4 6	2 4 6 8	2 4 6 8 10	2 4 6 8 10 12	2 4 6 8 10 12 14	2 4 6 8 10 12 14 16	2 4 6 8 10 12 14 16 18

Agora vem a chamada da função MostraVetor(Numeros, Tam);

```
void MostraVetor(int* Array, int Tam)
{
    std::cout << "[ ";
    for (int i = 0; i < Tam; i++)
    {
        std::cout << Array[i] << " ";
    }
    std::cout << "] \n\n";
}</pre>
```

Observe que a função possui os mesmos parâmetros int* Array, int Tam

void MostraVetor(int* Array, int Tam)

Logo a variável ponteiro irá receber o endereço de memória do primeiro elemento do vetor Numeros

E novamente vamos assumir que este endereço seja 100

Então primeiro vai para tela um colchete apenas para deixar mais bonita a saída

```
std::cout << "[ ";
```

Depois começa o loop for que vai usar a aritmética de ponteiros

```
for (int i = 0; i < Tam; i++)
{
std::cout << Array[i] << " ";
}</pre>
```

Veja que mais uma vez temos Array[i] que é o mesmo que *(Array + i)

Lembrando que o Vetor Numeros foi alterado pela função DuplicaVetor e ficou assim:

Índice do vetor	0	1	2	3	4	5	6	7	8	9
Endereço	100-103	104-107	108-111	112-115	116-119	120-123	124-127	128-131	132-135	136-139
Valor	2	4	6	8	10	12	14	16	18	20

Assim teremos para cada valor de i o seguinte

```
std::cout << Array[0] << " ";
                                       O mesmo que std::cout << *(Array + 0) << "
i = 0
                                       std::cout << *(Array) << " "; ou
                                       std::cout << *(100) << " ";
                                       logo vai para tela o valor contido no endereço 100 que é 2
        std::cout << Array[1] << " ";
                                       O mesmo que std::cout << *(Array + 1) << " ";ou
i = 1
                                       std::cout << *(Array + 1) << " "; ou
                                       std::cout << *(100 + 1) << " "; // 1 é 4 bytes aritmética
                                       de ponteiros!
                                       é como se fosse std::cout << *(100 + 4) << " ";
                                       ou std::cout << *(104) << " ";
                                       logo vai para tela o valor contido no endereço 104 que é 4
        std::cout << Array[2] <<
                                       std::cout << *(Array + 2) << " "; ou
i = 2
                                       std::cout << *(100 + 8) << " "; ou
                                       std::cout << *(108) << " ";
                                       logo vai para tela o valor contido no endereço 108 que é 6
```

E então a aritmética de ponteiros percorre todo Array colocando na tela seus 10 elementos

```
[ 2 4 6 8 10 12 14 16 18 20 ]
```

Temos então a chamada de uma função MostraVetor(InverteVetor(Numeros, Tam), Tam);

```
MostraVetor(Numeros, Tam);
DuplicaVetor(Numeros, Tam);
MostraVetor(Numeros, Tam);
//Aqui Mostrar vetor recebe o retorno da função Inverte Vetor
//E este retorno será o endereço do primeiro elemento do array inventid
MostraVetor(InverteVetor(Numeros, Tam), Tam);
system("PAUSE");
return 0;
```

Observe que uma função pode retornar valores ou não. Além disso, você pode ter chamadas de funções como argumentos na chamada de funções...

Como assim?

Veja que estamos chamando a função InverteVetor(Numeros, Tam) dentro da região destinada aos argumentos da função MostraVetor

Lembre que a função MostraVetor pode receber os seguintes argumentos:

Um ponteiro do tipo int que será colocado no ponteiro Array e um valor inteiro que será colocado em Tam. void MostraVetor(int* Array, int Tam);

logo a linha de código abaixo primeiro vai chamar a função InverteVetor(Numeros, Tam) para somente depois chamar a função MostraVetor. Pois é primeiro necessário saber o valor do primeiro argumento

```
MostraVetor(InverteVetor(Numeros, Tam), Tam);
MostraVetor(int* Array, int Tam);
```

Portanto o compilador vai primeiro executar: InverteVetor(Numeros, Tam)

```
int* InverteVetor(int* Array, int Tam)
{
  int j = 0;
  static int ArrayInvertido[10];
  for (int i = Tam - 1; i >= 0; i--)
  {
   ArrayInvertido[j] = Array[i];
   j++;
  }
  return ArrayInvertido;
}
```

O que já chama atenção e confunde alunos pelo mundo todo é o tipo de retorno desta função

Veja que o tipo de retorno da função InverteVetor é do tipo int*

Logo podemos ler int* como sendo um ponteiro do tipo int. Então esta função InverteVetor deverá retornar um ponteiro do tipo int. Como ponteiros são variáveis especiais que armazenam dentro dele um endereço de memória esta função vai retornar um endereço de memória!

A função começa declarando uma variável auxiliar j que é iniciada com zero

```
int j = 0;
```

Depois declaramos um Vetor estático.

Mas por que este static?

Lembre das aulas que as variáveis de uma função são locais a ela e são destruídas da memória quando a função parar sua execução. Mas nós precisaremos manter o array invertido na memória para poder colocar ele na tela. Senão tivesse static este ArrayInvertido[10] seria destruído na saída da função e não é isso que queremos. Logo indicamos que ele é uma variável estática da função InverteVetor. Isto é, uma variável que é preservada na memória mesmo após a função ser encerrada.

Depois disso iniciamos o loop for.

```
for (int i = Tam - 1; i >= 0; i--)
{
    ArrayInvertido[j] = Array[i];
j++;
```

```
A ideia da inversão é simples.

J vai de zero até 9, enquanto i vai de 9 até 0
assim teremos dentro do for todas estas execuções

ArrayInvertido[0] = Array[9];

ArrayInvertido[1] = Array[8];

ArrayInvertido[2] = Array[7];

ArrayInvertido[3] = Array[6];

ArrayInvertido[4] = Array[5];

ArrayInvertido[5] = Array[4];

ArrayInvertido[6] = Array[3];

ArrayInvertido[7] = Array[2];

ArrayInvertido[8] = Array[1];

ArrayInvertido[9] = Array[0];

E isso vai inverter o Array.
```

E ai vem algo importante! O retorno da função. Este array acima é o array estático ArrayInvertido declarado e preenchido na função com valores invertidos que vieram do vetor Numeros

Veja que temos o comando return ArrayInvertido;

E o que este comando quer dizer?

Lembre-se pois já comentei isso, o nome de um vetor(também chamado de array) é o mesmo que o endereço de memória do primeiro elemento dele. Supondo que seja 200 o primeiro endereço desta variável vetor de nome **ArrayInvertido**

Logo este comando return está na realidade retornando o seguinte: return & ArrayInvertido[0];

Assim temos agora o nosso retorno e o compilador poderá executar a função MostraVetor

A chamada fica assim:

MostraVetor(InverteVetor(Numeros, Tam), Tam); // Função chamada, mas tem que resolver primeiro outra chamada de função InverteVetor(Numeros, Tam) para poder compor seus argumentos

Logo outra função é chamada passando como argumentos o endereço de memória do vetor Numeros e o Tamanho deste vetor - InverteVetor(Numeros, Tam)

Esta função InverteVetor constrói outro vetor de nome **ArrayInvertido com os valores invertidos do vetor Numeros**

				Array	Invertid	0				
Índice do vetor	0	1	2	3	4	5	6	7	8	9
Endereço	200-203	204-207	208-211	212-215	216-219	220-223	224-227	228-231	232-135	236-239
Valor	20	18	16	14	12	10	8	6	4	2

E ai ela retorna o valor do endereço de memória do primeiro elemento do vetor

return ArrayInvertido; que significa return & ArrayInvertido[0];

este valor retornado vai compor o argumento da função MostraVetor

MostraVetor(InverteVetor(Numeros, Tam), Tam);

Vai ficar assim

MostraVetor(&ArrayInvertido[0], Tam);

Com isso a função MostraVetor irá ser chamada passando estes argumentos acima:

&ArrayInvertido[0], Tam

```
MostraVetor(&ArrayInvertido[0], Tam);

void MostraVetor(int* Array, int Tam)
{
    //coloca colchete antes do for
    std::cout << "[ ";
    //Coloca os elementos e finaliza o loop
    for (int i = 0; i < Tam; i++)
    {
        std::cout << Array[i] << " ";
    }
    //e depois coloca ] para fechar o vetor
    std::cout << "] \n\n";
}</pre>
```

Desta forma a função MostraVetor será capaz de percorrer via aritmética de ponteiros todos os elementos vetor ArrayInvertido pois seu parâmetro int* Array recebeu o endereço de memória de ArrayInvertido. Com isso, ele irá colocar os elementos de ArrayInvertido na tela seguindo a aritmética de ponteiros que já expliquei amplamente acima

```
[ 1 2 3 4 5 6 7 8 9 10 ]
[ 2 4 6 8 10 12 14 16 18 20 ]
[ 20 18 16 14 12 10 8 6 4 2 ]
```

Ufa! Foram 12 páginas, mas acredito que esclareceu muita coisa sobre a aulas e os conceitos abordados nela!

Grande Abraço!

Professor Marcos Pacheco