Arquitetura de Computadores, Algoritmos, Lógica de Programação e Circuitos Digitais

Danilo Rios
29/08/2025

Recapitulando

Recapitulando

- Array bidimensional
- Array tridimensional

Ponteiro

- O que é?

Ponteiro

- O que é?
  - É uma variável

Variável

- O que é?

Variável

- O que é?
  - É onde armazenamos alguma informação

Ponteiro x Variável

- Qual a diferença entre um ponteiro e a variável i?

Ponteiro x Variável

- Qual a diferença entre um ponteiro e a variável i?
  - O que é armazenado

Ponteiro

- Armazena o endereço de memória em que está alguma variável

Vamos voltar num parênteses do semestre passado

- Alguma ideia de qual parênteses será?

```c
#include <stdio.h>

int main() {

    int numeroInt;
    float numeroFloat;
    double numeroDouble;
    char caractere;

    printf("Digite um inteiro: ");
    scanf("%d", &numeroInt);
    printf("\n\nDigite um float: ");
    scanf("%f", &numeroFloat);
    printf("\n\nDigite um double: ");
    scanf("%lf", &numeroDouble);
    printf("\n\nDigite um caractere: ");
    scanf("%c", &caractere);

    printf("\n\nDigitou: %d, %f, %f, %c", numeroInt, numeroFloat, numeroDouble, caractere);

    return 0;
}
```

- scanf(tipo informação, & variável);
  - Tipo informação:
    - %d - int
    - %f - float
    - %lf - double
    - %c - char
  - & - e comercial
  - Variável que vai armazenar a informação

& variável

- O que significa?

& variável

- O que significa?
  - A informação lida do teclado vai ser armazenada no endereço de memória da variável

scanf

- O scanf não atribui um valor para a variável
  - int x = 10;
  - int x = scanf(“%d”);
- Ele coloca a informação direto na memória do computador
  - Por isso que precisa do endereço de memória

- Quando o programa vai utilizar a variável, a variável “pega” a informação que está guardada no endereço de memória que é dela e está com a nova informação

Endereço de memória

```
int int1;
int int2;
float float1;
float float2;
double double1;
double double2;
char char1;
char char2;
```

```
int1 = 000000000062FE4C
int2 = 000000000062FE48
float1 = 000000000062FE44
float2 = 000000000062FE40
double1 = 000000000062FE38
double2 = 000000000062FE30
char1 = 000000000062FE2F
char2 = 000000000062FE2E
```

Endereço de memória

```
int int1;
int int2;
float float1;
float float2;
double double1;
double double2;
char char1;
char char2;
```

```
int1 = 000000000062FE4C
int2 = 000000000062FE48
float1 = 000000000062FE44
float2 = 000000000062FE40
double1 = 000000000062FE38
double2 = 000000000062FE30
char1 = 000000000062FE2F
char2 = 000000000062FE2E
```

} 4 bytes
} 4 bytes
} 4 bytes
} 8 bytes
} 8 bytes
} 1 byte
} 1 byte

Endereço de memória

- Sempre estão corretos?

Endereço de memória

- Sempre estão corretos?
  - Nem sempre...

```
int int1;
int int2;
float float1;
double double1;
char char1;
char char2;
```

```
int1 = 000000000062FE4C
int2 = 000000000062FE48
float1 = 000000000062FE44
double1 = 000000000062FE38
char1 = 000000000062FE37
char2 = 000000000062FE36
```

} 4 bytes
} 4 bytes
} 12 bytes
} 1 byte
} 1 byte

Endereço de memória

```
int int1;
int int2;
float float1;
double double1;
char char1;
char char2;
```

```
int1 = 000000000062FE4C
int2 = 000000000062FE48
float1 = 000000000062FE44
double1 = 000000000062FE38
char1 = 000000000062FE37
char2 = 000000000062F
```

} 12 bytes ?

No slide sobre tipo de variável não estava que double são 8 bytes !?!?

Endereço de memória

- Sempre a variável vai ter o tamanho correto
- O que pode acontecer é ter um endereços de memória que são pulados

Endereço de memória

- Compilador ao transformar o código em linguagem de máquina pode colocar esses “espaços em branco” para otimizar/melhorar o acesso à informação

Endereço de memória

- 1 byte = 8 bits
- SO 32 bits = 4 bytes
  - Cada “grupo” de memória são 4 bytes
- SO 64 bits = 8 bytes
  - Cada “grupo” de memória são 8 bytes

- Então no exemplo ao invés de dividir o double de 8 bytes entre 2 “grupos” de memória o compilador dá o “espaço” e coloca a informação num único grupo

Endereço de memória

Variável	Hexadecimal	Decimal	Dividido por 8
	62FE50	6487632	**810954**
int1	62FE4C	6487628	810953.5
int2	62FE48	6487624	810953
float1	62FE44	6487620	810952.5
double1	62FE38	6487608	**810951**
char1	62FE37	6487607	810950.875
char2	62FE36	6487606	810950.75

Endereço de memória

```
int1 = 000000000062FE4C
int2 = 000000000062FE48
float1 = 000000000062FE44
double1 = 000000000062FE38
char1 = 000000000062FE37
char2 = 000000000062FE36
```

62FE50	62FE51	62FE52	62FE53	62FE54	62FE55	62FE56	62FE57
62FE48	62FE49	62FE4A	62FE4B	62FE4C	62FE4D	62FE4E	62FE4F
int2	int2	int2	int2	int1	int1	int1	int1
62FE40	62FE41	62FE42	62FE43	62FE44	62FE45	62FE46	62FE47
				float	float1	float1	float1
62FE38	62FE39	62FE3A	62FE3B	62FE3C	62FE3D	62FE3E	62FE3F
double1	double1	double1	double1	double1	double1	double1	double1
62FE30	62FE31	62FE32	62FE33	62FE34	62FE35	62FE36	62FE37
						char2	char1

Endereço de memória

- Obs.: Quem quiser fazer em casa
- Utilizando %p vai imprimir a informação em hexadecimal
- &<variavel> é o endereço de memória da variável

```
printf("int1 = %p\n",&int1);
```

Fechando o parênteses

- E voltando para o assunto de hoje

Ponteiro

- Identificado pelo *
  - Asterisco
    - Não é “asteristico”

Ponteiro

```c
#include <stdio.h>

int main() {

    int numero = 5;
    int *ponteiro = &numero;

    printf("numero:\n");
    printf("Valor: %d\n", numero);
    printf("Endereco: %p\n", &numero);

    printf("\n\n");
    printf("ponteiro:\n");
    printf("Valor: %p\n", ponteiro);
    printf("Valor apontado: %d\n", *ponteiro);
    printf("Endereco: %p\n", &ponteiro);

    return 0;
}
```

Ponteiro

```c
#include <stdio.h>

int main() {

    int numero = 5;
    int *ponteiro = &numero;

    printf("numero:\n");
    printf("Valor: %d\n", numero);
    printf("Endereco: %p\n", &numero);

    printf("\n\n");
    printf("ponteiro:\n");
    printf("Valor: %p\n", ponteiro);
    printf("Valor apontado: %d\n", *ponteiro);
    printf("Endereco: %p\n", &ponteiro);

    return 0;
}
```

```
numero:
Valor: 5
Endereco: 000000000062FE4C


ponteiro:
Valor: 000000000062FE4C
Valor apontado: 5
Endereco: 000000000062FE40

--------------------------------
Process exited after 0.1816 seconds with return value 0
Press any key to continue . . .
```

Ponteiro

- Tipo do ponteiro
  - O tipo do ponteiro tem que ser o mesmo da variável

Ponteiro

```c
#include <stdio.h>

int main() {

    int numero = 5;
    int *ponteiro = &numero;

    printf("Valor do numero: %d\n", numero);

    numero++;
    printf("Valor do numero: %d\n", numero);

    //colocando 100 no endereço de memória apontado
    *ponteiro = 100;
    printf("Valor do numero: %d\n", numero);

    return 0;
}
```

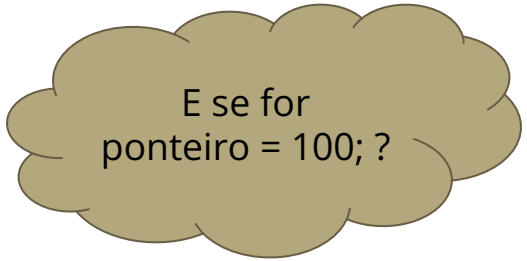
Ponteiro

```
Valor do numero: 5
Valor do numero: 6
Valor do numero: 100

--------------------------------
Process exited after 0.1703 seconds with return value 0
Press any key to continue . . .
```

```c
#include <stdio.h>

int main() {

    int numero = 5;
    int *ponteiro = &numero;

    printf("Valor do numero: %d\n", numero);

    numero++;
    printf("Valor do numero: %d\n", numero);

    //colocando 100 no endereço de memória apontado
    *ponteiro = 100;
    printf("Valor do numero: %d\n", numero);

    return 0;
}
```

Ponteiro

```c
#include <stdio.h>

int main() {

    int numero = 5;
    int *ponteiro = &numero;

    printf("Valor do numero: %d\n", numero);

    numero++;
    printf("Valor do numero: %d\n", numero);

    //colocando 100 no endereço de memória apontado
    *ponteiro = 100;
    printf("Valor do numero: %d\n", numero);

    return 0;
}
```



Ponteiro

```c
#include <stdio.h>

int main() {

    int numero = 5;
    int *ponteiro = &numero;

    printf("Ponteiro esta apontando para %p\n", ponteiro);
    printf("Valor apontado: %d\n", *ponteiro);

    printf("Valor do numero: %d\n", numero);

    numero++;
    printf("Valor do numero: %d\n", numero);

    //colocando 100 no endereço de memória apontado
    ponteiro = 100;
    printf("Valor do numero: %d\n", numero);

    printf("Ponteiro esta apontando para %p\n", ponteiro);
    printf("Valor apontado: %d\n", *ponteiro);

    return 0;
}
```

Ponteiro

```c
#include <stdio.h>

int main() {

    int numero = 5;
    int *ponteiro = &numero;

    printf("Ponteiro esta apontando para %p\n", ponteiro);
    printf("Valor apontado: %d\n", *ponteiro);

    printf("Valor do numero: %d\n", numero);

    numero++;
    printf("Valor do numero: %d\n", numero);

    //colocando 100 no endereço de memória apontado
    ponteiro = 100;
    printf("Valor do numero: %d\n", numero);

    printf("Ponteiro esta apontando para %p\n", ponteiro);
    printf("Valor apontado: %d\n", *ponteiro);

    return 0;
}
```

```
Ponteiro esta apontando para 000000000062FE44
Valor apontado: 5
Valor do numero: 5
Valor do numero: 6
Valor do numero: 6
Ponteiro esta apontando para 0000000000000064

--------------------------------
Process exited after 0.9711 seconds with return value 3221225477
Press any key to continue . . .
```

Array

- O que é mesmo?

Array

- O que é mesmo?
  - É uma variável que armazena + de 1 valor
  - Como que funciona com o ponteiro?

Ponteiro + Array

```c
#include <stdio.h>

int main() {

    int i;
    int numeros[] = {1,2,3,4,5,6,7,8,9,10};
    int *ponteiro = &numeros; //são varios valores, aponta para qual?

    printf("Ponteiro esta apontando para %p\n", ponteiro);

    for(i=0;i<10;i++) {
        //imprimindo os endereços de memória de cada posição
        printf("%d = %p\n", numeros[i], &numeros[i]);
    }

    return 0;
}
```

Ponteiro + Array

```
Ponteiro esta apontando para 000000000062FE10
1 = 000000000062FE10
2 = 000000000062FE14
3 = 000000000062FE18
4 = 000000000062FE1C
5 = 000000000062FE20
6 = 000000000062FE24
7 = 000000000062FE28
8 = 000000000062FE2C
9 = 000000000062FE30
10 = 000000000062FE34

--------------------------------
Process exited after 0.4777 seconds with return value 0
Press any key to continue . . .
```

```c
#include <stdio.h>

int main() {

    int i;
    int numeros[] = {1,2,3,4,5,6,7,8,9,10};
    int *ponteiro = &numeros; //são varios valores, aponta para qual?

    printf("Ponteiro esta apontando para %p\n", ponteiro);

    for(i=0;i<10;i++) {
        //imprimindo os endereços de memória de cada posição
        printf("%d = %p\n", numeros[i], &numeros[i]);
    }

    return 0;
}
```

Ponteiro + Array

```
Ponteiro esta apontando para 000000000062FE10
1 = 000000000062FE10
2 = 000000000062FE14
3 = 000000000062FE18
4 = 000000000062FE1C
5 = 000000000062FE20
6 = 000000000062FE24
7 = 000000000062FE28
8 = 000000000062FE2C
9 = 000000000062FE30
10 = 000000000062FE34

--------------------------------
Process exited after 0.4777 seconds with return value 0
Press any key to continue . . .
```

- Existe uma sequência
- Pula de 4 em 4 bytes
- Será que consigo mexer no ponteiro e acessar cada posição?

Ponteiro + Array

```c
#include <stdio.h>

int main() {
    int numeros[] = {1,2,3,4,5};
    int *ponteiro = numeros;

    printf("numeros[0] = %d\n", *ponteiro);
    printf("numeros[1] = %d\n", *(ponteiro + 1));
    printf("numeros[2] = %d\n", *(ponteiro + 2));
    printf("numeros[3] = %d\n", *(ponteiro + 3));
    printf("numeros[4] = %d\n", *(ponteiro + 4));


    return 0;
}
```

Ponteiro + Array

```
numeros[0] = 1
numeros[1] = 2
numeros[2] = 3
numeros[3] = 4
numeros[4] = 5

--------------------------------
Process exited after 0.1765 seconds with return value 0
Press any key to continue . . .
```

```c
#include <stdio.h>

int main() {
    int numeros[] = {1,2,3,4,5};
    int *ponteiro = numeros;

    printf("numeros[0] = %d\n", *ponteiro);
    printf("numeros[1] = %d\n", *(ponteiro + 1));
    printf("numeros[2] = %d\n", *(ponteiro + 2));
    printf("numeros[3] = %d\n", *(ponteiro + 3));
    printf("numeros[4] = %d\n", *(ponteiro + 4));


    return 0;
}
```

Ponteiro

- Como deve ser o código?
  - Pedir para a pessoa digitar 10 números que serão armazenados em um array
    - Porém cada número não deve ser guardado utilizando o índice, deve ser através do ponteiro
  - Imprimir todos os 10 valores 1 por linha

Ponteiro

```c
#include <stdio.h>

#define tamanho 10

int main() {
    int i;
    int numeros[tamanho];
    int *ponteiro = numeros;

    printf("Vou pedir para digitar 10 numeros\n");

    for(i=0;i<tamanho;i++) {
        printf("Digite #%d numero: ", i+1);
        scanf("%d", &(*(ponteiro + i)));
    }

    for(i=0;i<tamanho;i++) {
        printf("%d\n", numeros[i]);
    }

    return 0;
}
```

Exercício 5

- Todas as variáveis devem ter um ponteiro
- Não pode utilizar as variáveis, somente os ponteiros
  - No for pode usar i e j se quiser
- Menu com 2 opções
  - 1) Jogar
    - Criar um tabuleiro 4x4
    - Intercalar entre jogador 1 e jogador 2
    - Cada jogador deve escolher uma posição linha e coluna para colocar a sua peça X ou O
      - Se jogar numa casa já preenchida o mesmo jogador deve realizar nova jogada
    - Quando todas as casa estiverem preenchida finaliza e volta ao menu
  - 2) Sair
    - Finalizar o código
- Entrega até: 12/09

Perguntas?

Obrigado!

Até a próxima aula!