ESTRUTURA DE DADOS E ALGORITMOS I

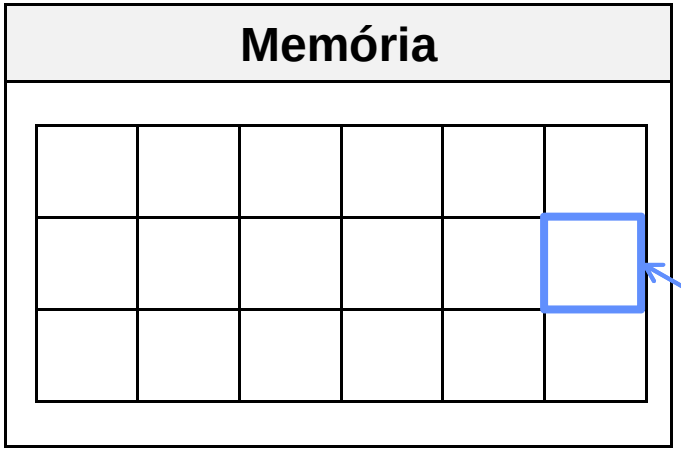
Ponteiros

Profº. Sérgio Roberto Costa Vieira, M.Sc.
Cursos de Computação
2º. Período

ARMAZENANDO INFORMAÇÕES

- O armazenamento de informações por meio de variáveis ocupa espaço na memória, associando a um endereço de memória.

- Vamos tentar visualizar este funcionamento:



Cada um dos blocos que compõe a grade equivale a 1 byte, isto é, só poderá armazenar 1 byte.

ARMAZENANDO INFORMAÇÕES

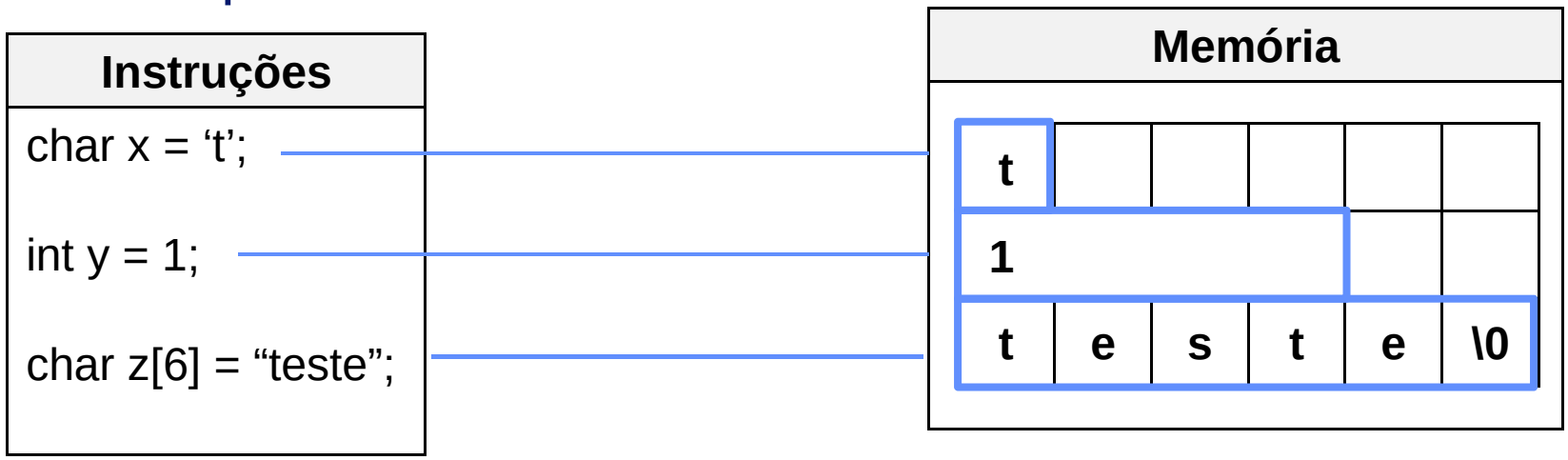
- Cada tipo de dado declarado em um programa ocupa um tamanho diferente na memória do computador (alocação de memória).

- Vejamos exemplos:

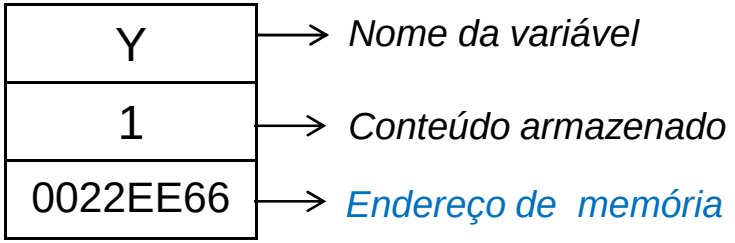
Tipo e dado	Memória
char	1 byte
int	4 bytes
float	4 bytes
double	8 bytes

ARMAZENANDO INFORMAÇÕES

- Exemplo:



- Cada espaço é dividido em identificador, conteúdo e endereço de memória.



DEFINIÇÃO DE PONTEIRO

- É uma variável que armazena um endereço de memória, isto é, o ponteiro é um tipo de variável capaz de atribuir somente os endereços de outra variável ao seu conteúdo.

- O ponteiro tem por função **apontar** para um endereço de memória determinado, endereço que o ponteiro armazena.

- O ponteiro é declarado como uma variável comum, com exceção que é obrigatório o uso do " * " antes do nome da variável:

```
Tipo  *nome;          int  *p;
```

POR QUE USAR PONTEIROS?

- Quando é necessário que uma função retorne mais de um valor, utilizamos as funções com passagem por referência. Como os ponteiros armazenam endereços de outras variáveis, isso possibilita que o conteúdo dos parâmetros seja modificado diretamente.
- Manipulação de vetores, matrizes e strings;
- Usando ponteiros como referência, o que é muito importante na criação de estruturas de dados, como por exemplo, listas encadeadas, árvores e grafos.
- Processo de alocação e liberação de memória das variáveis durante a execução de um programa – alocação dinâmica de memória.

OPERADORES DE MANIPULAÇÃO

- O operador (*) obtém o VALOR armazenado em um ENDEREÇO DE MEMÓRIA, e também sinaliza o ponteiro na declaração de variáveis.

- O operador (&) obtém o ENDEREÇO DE MEMÓRIA de uma variável.

- Já utilizamos o operador (&) no comando SCANF;

```
scanf("%c", &x);
```

Estrutura de Dados e Algoritmos I

Ponteiros

Exibindo o endereço de memória das variáveis

Exemplo

```
#include<stdio.h>
#include<conio.h>

int main() {
   char letra = ‘s’;
   int  idade = 35;
   char  nome[10] = “alberto”;
   float  peso = 87.8;
   float altura = 1.82;


   printf(“ Exibindo o endereço de memória das variáveis \n”);
   printf(“\n Letra %p”, &letra);
   printf(“\n Letra  = %c”, letra);
   printf(“\n Idade %p”, &idade);
   printf(“\n Idade  = %d”, idade);

   printf(“\n Nome %p”, nome);
   printf(“\n Nome  = %s”, nome);

   printf(“\n Peso %p”, &peso);
   printf(“\n Peso  = %.2f”, peso);

   printf(“\n Altura %p”, &altura);
   printf(“\n Altura  = %.2f”, altura);

getch();
return 0;
}
```

Estrutura de Dados e Algoritmos I

Ponteiros

Atribuindo um valor a uma variável ponteiro

Exemplo

```
#include<stdio.h>
#include<conio.h>

int main() {
   int  idade = 35;
   int  *ptr_idade;

   ptr_idade = &idade;

printf("\n Idade = %d", idade);
printf("\n Endereço de Idade  = %p",
&idade);
   printf("\n ptr_idade = %d",
*ptr_idade);
   printf("\n Endereço do ponteiro  =
%p", ptr_idade);

getch();
return 0;
}
```

No caso de inicializar um ponteiro com variáveis do tipo strings ou arrays, não é necessário o uso do operador de endereço (&), pois eles já são considerados ponteiros constantes.

Estrutura de Dados e Algoritmos I

Ponteiros

Outro exemplo de Atribuição de valores a uma variável ponteiro

Exemplo

```
#include<stdio.h>
#include<conio.h>

int main() {
   int  x = 4, y = 7;
   int  *px, *py;

printf("\n &x = %p\t  x = %d", &x, x);
printf("\n &y = %p\t  y = %d", &y, y);

px = &x;

py = &y;

printf("\n &px = %p\t  *px = %d", px, *px);

printf("\n &py = %p\t  *py = %d", py, *py);

getch();
return 0;
}
```

Estrutura de Dados e Algoritmos I

Ponteiros

Indexação de ponteiros com arrays

Exemplo

```
#include<stdio.h>
#include<conio.h>

int main() {
   int  i = 0;
  int  numeros[10]={1,2,3,4,5,6,7,8,9,10};
   int  *p;

   p = numeros;

   printf("Apresentar os valores");
   for(i=0; i < 10; i++){
        printf("%d", p[i]);
   }
   printf("\n Terceiro elemento do vetor %d", p[2]);

   *p = 10;
   printf("\n Primeiro elemento do vetor %d", p[0]);

   p = &numeros[2];
   printf("\n Primeiro elemento do vetor %d", p[0]);

   *p = 30;
   printf("\n Primeiro elemento do vetor %d", p[0]);
   printf("\n Terceiro elemento do vetor %d", p[2]);

   p = numeros;
   for(i=0; i < 10; i++){
        printf("%d", p[i]);
   }
   printf(" \n ");
getch();
return 0;  }
```

Estrutura de Dados e Algoritmos I

Ponteiros

Uso de ponteiros em expressões matemáticas

Exemplo

```
#include<stdio.h>
#include<conio.h>

int main() {
   int  x=0, a=5, b=2, *p;

   p = &a;

   x = a * b;
   printf("\n Valor = %d", x);

   x = (*p) + b;
   printf("\n Valor = %d", x);

   x = (*p) * b;
   printf("\n Valor = %d", x);

   x = (a + b) * a;
   printf("\n Valor = %d", x);

   x = ((*p) + b) * (*p);
   printf("\n Valor = %d", x);

getch();
return 0;
}
```

Estrutura de Dados e Algoritmos I

Ponteiros

Alterando o valor armazenado no endereço apontado

Exemplo

```
#include<stdio.h>
#include<conio.h>

int main() {
   int  numero = 35;
   int  *ptr;

   ptr = &numero;

 printf("\n Endereco do Valor = %p", ptr);
 printf("\n Conteudo = %d", *ptr);

   *ptr = 25;

 printf("\n Endereco do Valor = %p", ptr);
 printf("\n Conteudo = %d", *ptr);

getch();
return 0;
}
```

Estrutura de Dados e Algoritmos I

Ponteiros

Passando argumentos por referência com ponteiros

Exemplo

```
#include<stdio.h>
#include<conio.h>

void reajusta20( float * , float * );

int main() {
   float  val_preco, val_reaj;

   do {
 printf("\n Insira o preco atual: ");
 sacnf("%f", &val_preco);

 reajusta20( &val_preco, &val_reaj);
 //Enviando endereços de memória

printf("\n O novo preco é %.2f \n ", val_preco);
printf("\n O aumento foi de %.2f \n ", val_reaj);

} while (val_preco != 0.0);
```

```
 getch();
return 0;
}

//reajusta o preço em 20%
void reajusta20( float *preco, float *reaj){
  *reaj = *preco * 0.2;
   *preco *= 1.2;
}
```

Estrutura de Dados e Algoritmos I

Ponteiros

Utilizando ponteiros para percorrer arrays

Exemplo

```
#include<stdio.h>
#include<conio.h>

int main() {
   int M[5]={92, 81, 70, 69, 58};
   int i, *p;

   p = M;
   printf("Exemplo 1\n\n");
   for(i=0; i<5; i++){
            printf("%d\n", *(M+i));
   }
   printf("\n\nExemplo 2\n\n");
   for(i=0; i<5; i++){
            printf("%d\t", *(p++));
   }

   p = M;
   printf("\n\nExemplo 3\n\n");
   for(i=0; i<5; i++){
            printf("%d\t", *(p));
            p++;
   }

getch();
return 0;
}
```

PONTEIROS PARA PONTEIROS

- Uma outra possibilidade no uso dos ponteiros é a criação de uma variável, que aponta para um ponteiro, chamada variável do tipo ponteiro para ponteiro.

- Ponteiro que aponta para outro ponteiro.

- No caso, sua declaração utiliza o operador de conteúdo de forma dupla:

```
tipo  **ponteiro;
```

Estrutura de Dados e Algoritmos I

Ponteiros

Usando ponteiro para ponteiro

Exemplo

```
#include<stdio.h>
#include<conio.h>

int main() {
   int  x;
   int  *p_x;
   int  **p_p_x;

   p_x = &x;

   p_p_x = &p_x;

   x = 10;

   printf("\n Conteudo de X = %d", x);
   printf("\n Endereco de X = %p", p_x);
   printf("\n Endereco de p_x = %p", p_p_x);

   printf("\n Conteudo de X = %d", *p_x);
   printf("\n Endereco de X = %p", *p_p_x);
   printf("\n Conteudo de X = %d", **p_p_x);

getch();
return 0;
}
```

ARITMÉTICA DE PONTEIROS

- No processo de manipulação dos ponteiros, é possível realizar operações de soma e subtração nos ponteiros.
- Não se esqueça que os ponteiros armazenam apenas conteúdos, então você estará adicionando ou subtraindo valores baseados no endereço contido no ponteiro.
- Isso significa que, ao somar o valor 1 a um ponteiro, o endereço contido no ponteiro será modificado para o próximo endereço de memória correspondente ao tipo de dado.
    - Você deve conhecer o tamanho dos tipos de dados

ARITMÉTICA DE PONTEIROS

Tipo	Bits	Bytes	Escala
char	8	1	-128 a 127
int	32	4	-2.147.483.648 a 2.147.483.647
short	16	2	-32.765 a 32.767
long	32	4	-2.147.483.648 a 2.147.483.647
unsigned char	8	1	0 a 255
unsigned	32	4	0 a 4.294.967.295
float	32	4	3.4×10^{-38} a 3.4×10^{38}
double	64	8	1.7×10^{-308} a 1.7×10^{308}
long double	80	10	3.4×10^{-4932} a 3.4×10^{4932}
void	0	0	Nenhum valor

ARITMÉTICA DE PONTEIROS

- Vejamos exemplo:

char x; ⟶ *O endereço de x é alocado na memória e equivale a 2000.*

char *p; ⟶ *Declaração do ponteiro do tipo char.*

p = &x; ⟶ *O endereço de x (no caso 2000) é atribuído ao ponteiro.*

p = p + 4; ⟶ *O endereço resultante será 2004, pois estamos tratando do tipo de dados char que requer 1 byte de memória. Ocorre então a soma de 4 bytes ao endereço original.*

ARITMÉTICA DE PONTEIROS

- Se o mesmo exemplo fosse baseado no tipo de dado int:

int x; → *O endereço de x é alocado na memória e equivale a 2000.*

int *p; → *Declaração do ponteiro do tipo int.*

p = &x; → *O endereço de x (no caso 2000) é atribuído ao ponteiro.*

p = p + 4; → *O endereço resultante será 2016, pois estamos tratando do tipo de dados int que requer 4 bytes de memória. Ocorre então a soma de 4 bytes multiplicado por 4 ao endereço original. 2000 + 4 * (4 bytes)*

Estrutura de Dados e Algoritmos I

Ponteiros

Visualizando como funciona a aritmética de ponteiros

Exemplo

```
#include<stdio.h>
#include<conio.h>

int main() {
   char  letra[5] = {‘a’, ‘e’, ‘i’, ‘o’, ‘u’};
   int  nr[5] = {30, 12, 67, 13, 41};
   char  *ptr_letra;
   int  *ptr_nr;

   ptr_letra = letra;
   ptr_nr = nr;

   printf(“Visualizando como funciona a aritmetica dos ponteiros \n”);
   printf(“\n array letra = a,e,i,o,u”);
   printf(“\n array nr = 30,12,67,13,41”);
   printf(“\n Verificando o tamanho dos tipos de dados”);
   printf(“\n tipo char %d”, sizeof(char));
   printf(“\n tipo int %d”, sizeof(int));

   printf(“\n Ponteiro letra = %c”, *ptr_letra);
   printf(“\n Endereco letra = %p”, ptr_letra);
   printf(“\n Ponteiro nr = %d”, *ptr_nr);
   printf(“\n Endereco nr = %p”, ptr_nr);

printf(“\n Incrementando os ponteiros”);
ptr_letra += 3;
ptr_nr += 2;

   printf(“\n Ponteiro letra = %c”, *ptr_letra);
   printf(“\n Endereco letra = %p”, ptr_letra);
   printf(“\n Ponteiro nr = %d”, *ptr_nr);
   printf(“\n Endereco nr = %p”, ptr_nr);

getch();
return 0;
}
```

Estrutura de Dados e Algoritmos I

Ponteiros

Visualizando como funciona operações com ponteiros

Exemplo 1

```
#include<stdio.h>
#include<conio.h>

int main() {
   int  *p, x = 10;

   p = &x;   /* Atribuições */
printf("\n Conteudo apontado %d", *p );
   *p = (*p)+1;
printf("\n Conteudo apontado %d", *p );
   *p = (*p) * 10;
printf("\n Conteudo apontado %d", *p );
getch();
return 0; }
```

Exemplo 2

```
#include<stdio.h>
#include<conio.h>

int main() {
   int  *p, x = 10;

   p = &x;   /* Atribuições */
printf("\n Conteudo apontado %d", *p );
   *p = (*p)-1;
printf("\n Conteudo apontado %d", *p );
   *p = ((*p)+1) / 10.0;
printf("\n Conteudo apontado %d", *p );
getch();
return 0; }
```

Estrutura de Dados e Algoritmos I

Ponteiros

Utilizando ponteiro em chamada por referência

Exemplo

```
#include<stdio.h>
#include<conio.h>

int  valor( int  *a, int  *b ){
    *a = *a + 3;
    *b = *b + 2;

     pritnf("\nValores da função:\n");
     printf("n1 = %d", *a);
     printf("n2 = %d", *b);
}

int main() {

   int  n1 = 2, n2 = 3, total;

   printf(" \n ");

pritnf("\nValores Iniciais:\n");
printf("n1 = %d", n1);
printf("n2 = %d", n2);

valor(&n1, &n2);

pritnf("\nValores após Função:\n");
printf("n1 = %d", n1);
printf("n2 = %d", n2);

getch();
return 0;
}
```

Estrutura de Dados e Algoritmos I

Ponteiros

Operações relacionais com ponteiros

Exemplo 1

```
#include<stdio.h>
#include<conio.h>

int main() {

   int  *p, *p1, x, y;

   p = &x;
   p1 = &y;
   if( p == p1)
      printf("Ponteiros são iguais\n");
   else
      printf("Ponteiros diferentes\n");

getch();
return 0;
}
```

Exemplo 2

```
#include<stdio.h>
#include<conio.h>

int main() {

   int  *p, *p1, x, y;

   p = &x;
   p1 = p;
   if( p == p1)
      printf("Ponteiros são iguais\n");
   else
      printf("Ponteiros diferentes\n");

getch();
return 0;
}
```

Operações relacionais com ponteiros fazendo uma comparação

Exemplo

```
#include<stdio.h>
#include<conio.h>

int main() {

   int  *p, *p1, x=10, y=20;

   p = &x;
   p1 = &y;
   if( *p > *p1)
      printf("P %c maior \n",130);
   else
      printf("P não %c maior\n",130);

getch();
return 0;
}
```

Estrutura de Dados e Algoritmos I

Ponteiros

Equivalência entre ponteiros e arrays

Exemplo

```
#include<stdio.h>
#include<conio.h>

int main() {

   int  vet[5] = {1,2,3,4,5};
   int  *p, indice = 2;

   p = vet;
   //vet[0] é equivalente a *p
   printf("%d \n", *p);
   printf("%d \n", vet[0]);
   //vet[indice] é equivalente a *(p+indice)
   printf("%d \n", vet[indice]);
   printf("%d \n", *(p+indice));

   //vet é equivalente a &vet[0]
   printf("%d \n", vet );
   printf("%d \n", &vet[0] );

   //&vet[indice] é equivalente a (vet+indice)
   printf("%d \n", &vet[indice] );
   printf("%d \n", (vet+indice) );

getch();
return 0;
}
```

Estrutura de Dados e Algoritmos I

Ponteiros

Array de ponteiros

Exemplo

```
#include<stdio.h>
#include<conio.h>

int main() {
   int  *pvet[2];
   int  x =10, y[2] = {20,30};

   pvet[0] = &x;
   pvet[1] = y;

   //imprime os endereços das variáveis
   printf("pvet[0] = %p \n", pvet[0]);
   printf("pvet[1] = %p \n", pvet[1]);

   //imprime o conteúdo das variáveis
   printf("pvet[0] = %d \n", *pvet[0]);


   //imprime uma posição do vetor
   printf("pvet[1][0] = %d \n", pvet[1][0] );
    printf("pvet[1][1] = %d \n", pvet[1][1] );

getch();
return 0;
}
```

Estrutura de Dados e Algoritmos I

Ponteiros

Percorrendo um array de Structs com ponteiro

Exemplo

```
#include<stdio.h>
#include<conio.h>

typedef  struct CadastroAluno {
   char  nome[50];
   int  codigo, idade;
}Aluno;

int main() {
 Aluno A[ 3 ], *p;
 int i;

 p = A;

for( i=0; i < 3; i++ ) {
 printf("Informe os dados do %do Aluno:", i+1);
 printf("\n Digite o nome: ");
 fflush(stdin);
 gets(p->nome);
 printf("\n Digite o codigo: ");
 scanf("%d", &p->codigo);
 printf("\n Digite a idade: ");
 scanf("%d", &p->idade);
 p++;
}

p = A;

for( i=0; i < 3; i++ ) {
    printf("\n\n%do  Aluno: ", i+1);
    printf(“ Codigo: %d ”, p->codigo );
    printf(“ Nome: %s ”, p->nome );
    printf(“ Idade: %d ”, p->idade );
    p++;
}

getch();
return 0;
}
```

ESTRUTURA DE DADOS E ALGORITMOS I

Ponteiros

Profº. Sérgio Roberto Costa Vieira, M.Sc.
Cursos de Computação
2º. Período