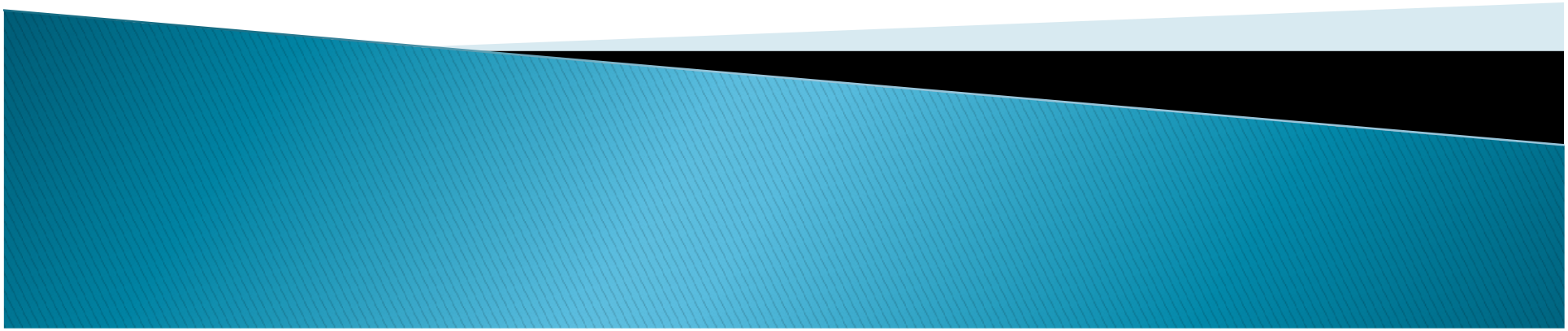
Alocação Dinâmica de Memória +exemplos

Prof. Nilton Cézar de Paula



Ex2

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int a, *b;

    a = 3;
    b = &a;
    *b += 5;

    printf("Valor de a: %d\n",a);
    printf("Valor de b: %d\n",*b);

    return 0;
}
```

Ex3

```c
#include <stdio.h>

void swap(int a, int b)
{
    int x;
    x = a;
    a = b;
    b = x;
}

int main()
{
    int a, b;

    a = 3;
    b = 5;
    swap(a, b);

    printf("Valor de a: %d\n",a);
    printf("Valor de b: %d\n",b);

    return 0;
}
```

Ex4

```c
#include <stdio.h>

void swap(int *a, int *b)
{
    int *x;
    x = a;
    a = b;
    b = x;
}

int main()
{
    int a, b;

    a = 3;
    b = 5;
    swap(&a, &b);

    printf("Valor de a: %d\n",a);
    printf("Valor de b: %d\n",b);

    return 0;
}
```

Ex5

```c
#include <stdio.h>

void swap(int *a, int *b)
{
    int x;
    x = *a;
    *a = *b;
    *b = x;
}

int main()
{
    int a, b;

    a = 3;
    b = 5;
    swap(&a, &b);

    printf("Valor de a: %d\n",a);
    printf("Valor de b: %d\n",b);

    return 0;
}
```

Ex6

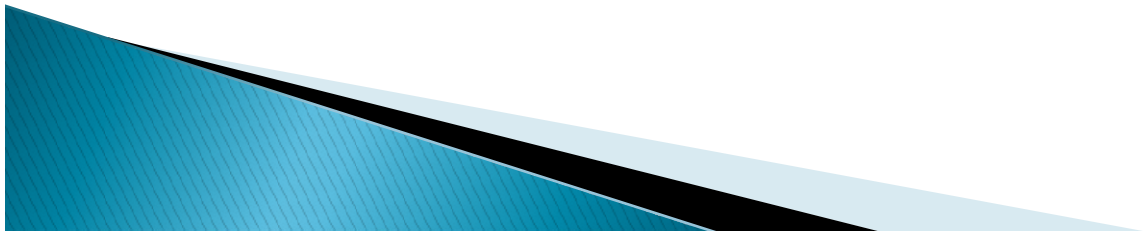
```c
#include <stdio.h>

int main()
{
    char x, y;

    x = 65;
    y = 'B';

    printf("Primeira apresentacao: %c, %c\n",x,y);
    printf("Segunda  apresentacao: %d, %d\n",x,y);

    return 0;
}
```



Ex7

```c
#include <stdio.h>

int main()
{
    char x, y;

    x = 65;
    y = 'B';

    x++;
    y--;

    printf("Primeira apresentacao: %c, %c\n",x,y);
    printf("Segunda  apresentacao: %d, %d\n",x,y);

    return 0;
}
```

Ex8

```c
#include <stdio.h>

int main()
{
    char x[] = "abacaxi";

    char *y = "abacaxi";

    char *z = malloc(100);
    strcpy(z, "abacaxi");

    printf("Sao iguais?\n");
    printf("String x: %s\n",x);
    printf("String y: %s\n",y);
    printf("String z: %s\n",z);

    free(z);

    return 0;
}
```

Ex9

```c
#include <stdio.h>

int main()
{
    char x[] = "abacaxi";

    char *y = "abacaxi";

    char *z = malloc(100);
    strcpy(z, "abacaxi");

    printf("Sao iguais?\n");
    printf("String x: %s\n",x);
    printf("String y: %s\n",y);
    printf("String z: %s\n",z);

    //posso?
    //strcpy(x, "mamao papaia");
    //strcpy(y, "mamao papaia");

    strcpy(z, "mamao papaia");
    printf("String z (depois): %s\n",z);

    free(z);

    return 0;
}
```

Ex10

```c
#define TAM 10000

int main()
{
    char *pc = malloc(TAM);
    int *pi = malloc(TAM*sizeof(int));
    float *pf = malloc(TAM*sizeof(float));
    double *pd = malloc(TAM*sizeof(double));

    printf("Tamanho pc : %d\n",sizeof(pc));
    printf("Tamanho *pc: %d\n\n",sizeof(*pc));

    printf("Tamanho pi : %d\n",sizeof(pi));
    printf("Tamanho *pi: %d\n\n",sizeof(*pi));

    printf("Tamanho pf : %d\n",sizeof(pf));
    printf("Tamanho *pf: %d\n\n",sizeof(*pf));

    printf("Tamanho pd : %d\n",sizeof(pd));
    printf("Tamanho *pd: %d\n\n",sizeof(*pd));

    free(pc);
    free(pi);
    free(pf);
    free(pd);
```

Ex11

```c
int main() {
    int i;
    char *x = malloc(10);

    for(i=0;i<10;i++)
        x[i]='.';
    strcpy(x,"abacaxi");
    printf("String: %s\n\n",x);

    printf("String ate strlen: ");
    for(i=0;i<strlen(x);i++)
        printf("%c",x[i]);
    printf("\n");
    for(i=0;i<strlen(x);i++)
        printf("%c[%d] ",x[i],x[i]);

    printf("\n\n");
    printf("String ate espaco alocado (10 bytes): ");
    for(i=0;i<10;i++)
        printf("%c",x[i]);
    printf("\n");
    for(i=0;i<10;i++)
        printf("%c[%d] ",x[i],x[i]);
    printf("\n");

    free(x);

    return 0;
```

Ex12

```c
int main() {
    char x[] = "cana brava";
    char *y;
    char *z;

    y = malloc(strlen(x)+1);
    strcpy(y,x);

    y[4] = '-';
    printf("String x: %s\n",x);
    printf("String y: %s\n",y);

    z = x;
    z[4] = '-';
    printf("String x: %s\n",x);
    printf("String z: %s\n",z);

    free(y);

    return 0;
}
```

Ex13

```c
int main() {
    char x[] = "cana brava";
    char *y, *z, *w;

    y = x;
    y[4] = '-';
    printf("String x: %s\n",x);
    printf("String y: %s\n\n",y);

    *(y + 4) = '#';
    printf("String x: %s\n",x);
    printf("String y: %s\n\n",y);

    z = x;
    z += 4;
    z[0] = '*';
    printf("String x: %s\n",x);
    printf("String z: %s\n\n",z);

    z++;
    strcpy(z,"caja");
    printf("String x: %s\n",x);
    printf("String z: %s\n\n",z);

    w = &x[4];
    w[0] = '%';
    printf("String x: %s\n",x);
    printf("String w: %s\n\n",w);

    return 0;
}
```

Ex14 – ponteiro para função

```
3   void Abrir(void) {
4       printf("\n\nAbrir");
5   }
6
7   void Ler(void) {
8       printf("\n\nLer");
9   }
10
11  int SomaInteiro(int a, int b) {
12      return(a+b);
13  }
14
15  float SomaReal(float a, float b) {
16      return(a+b);
17  }
```

```
18
19  int main() {
20      void (*ptr)(void);
21      int (*pti)(int,int);
22      float (*ptf)(float,float);
23      int x=10, y=20;
24      float z=10.5, w=20;
```

```
27          ptr=&Abrir;
28          (*ptr)();
29
30          ptr=&Ler;
31          (*ptr)();
32
33          pti=&SomaInteiro;
34          printf("\n\nA soma entre %d e %d vale: %d",x,y,(*pti)(x,y));
35
36          ptf=&SomaReal;
37          printf("\n\nA soma entre %f e %f vale: %f",z,w,(*ptf)(z,w));
```

Ex15 – ponteiro para função com passagem de parâmetro e sobrecarga de função

```
3   void Abrir(void) {
4       printf("\n\nAbrir");
5   }
6
7   void Ler(void) {
8       printf("\n\nLer");
9   }
10
11  int SomaInteiro(int a, int b) {
12      return(a+b);
13  }
14
15  float SomaReal(float a, float b) {
16      return(a+b);
17  }
```

```
18
19  void ExecutaFuncao(void (*e)(void)) {
20      (*e)();
21  }
22
23  void Soma(int a, int b, int (*s)(int, int)){
24      printf("\n\nA soma entre %d e %d vale: %d",a,b,(*s)(a,b));
25
26  }
27
28  void Soma(float a, float b, float (*s)(float, float)){
29      printf("\n\nA soma entre %f e %f vale: %f",a,b,(*s)(a,b));
30
31  }
```

```
32
33  int main() {
34
35      ExecutaFuncao(&Abrir);
36      ExecutaFuncao(&Ler);
37
38      Soma(15,20,&SomaInteiro);
39      Soma(20.5,10.8,&SomaReal);
```

Ex16 – Menu de opções usando Ponteiro para Função

```
3   void Abrir(void) {
4       printf("Abrir");
5   }
6
7   void Ler(void) {
8       printf("Ler");
9   }
10
11  void Fechar(void) {
12      printf("Fechar");
13  }
14
15  int main(void) {
16      int opcao;
17      void (*ptrFuncao[3])(void){&Abrir, &Ler, &Fechar};
18
19      do {
20          printf("\n\nMenu opcoes\n");     printf("\n1-Abrir");
21          printf("\n2-Ler");                printf("\n3-Fechar");
22          printf("\n4-Sair");               printf("\n\nQual opcao: ");
23          scanf("%d",&opcao);
24          if(opcao == 4)
25              break;
26          else {
27              if(opcao>=1 && opcao<=3) {
28                  printf("\n\nA FUNCAO ESCOLHIDA E: ");
29                  (*ptrFuncao[opcao-1])();
30              }
31          }
32      } while(true);
33      return(0);
```

Ex17 – structs

```
5   const char TRIANGULO[]="Triangulo";
6   const char QUADRADO[]="Quadrado";
7
8   typedef struct {
9       float x, y;  //coordenadas cartesianas de um ponto
10  } Ponto;
11
12  typedef struct {
13      char nome[10];
14      Ponto *pontos;    //pontos da figura
15      int np;      //numero de pontos
16  } Figura;
```

```
80  int main() {
81      Figura *triangulo, *quadrado;
82
83      triangulo = CriaFigura(TRIANGULO);
84      quadrado = CriaFigura(QUADRADO);
85
86      LeFigura(triangulo);
87      ImprimeFigura(triangulo);
88
89      LeFigura(quadrado);
90      ImprimeFigura(quadrado);
91
92      ApagaFigura(triangulo);
93      ApagaFigura(quadrado);
94
95      getchar();
96
97      return 0;
98  }
```

Ex17 - structs

```
5   const char TRIANGULO[]="Triangulo";
6   const char QUADRADO[]="Quadrado";
7
8   typedef struct {
9       float x, y;  //coordenadas cartesianas de um ponto
10  } Ponto;
11
12  typedef struct {
13      char nome[10];
14      Ponto *pontos;    //pontos da figura
15      int np;      //numero de pontos
16  } Figura;
```

```
18  //aloca espaço para uma figura de N pontos e retorna a cópia do registro da figura
19  Figura *CriaFigura(const char *nome) {
20      Figura *fig;
21
22      fig = (Figura *)calloc(1,sizeof(Figura));
23      if(fig == NULL) {
24          printf("\n\nMemoria insuficiente, alocação não realizada!");
25          return(NULL);
26      }
27
28      if(strcmp(nome,TRIANGULO)==0) {
29          fig->np = 3;
30          strcpy(fig->nome,TRIANGULO);
31      }
32      if(strcmp(nome,QUADRADO)==0) {
33          fig->np = 4;
34          strcpy(fig->nome,QUADRADO);
35      }
36
37      fig->pontos = (Ponto *)calloc(fig->np,sizeof(Ponto));
38      if(fig->pontos == NULL) {
39          printf("\n\nMemoria insuficiente, alocação não realizada!");
40          free(fig);
41          return(NULL);
42      }
43
44      return(fig);
45  }
```

Ex17 – structs

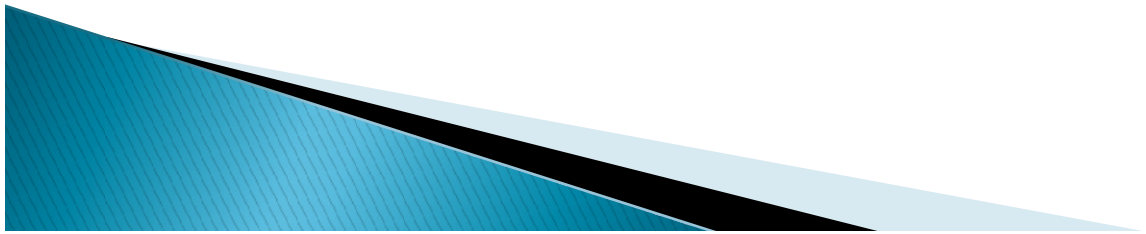
```
5    const char TRIANGULO[]="Triangulo";
6    const char QUADRADO[]="Quadrado";
7
8    typedef struct {
9        float x, y;   //coordenadas cartesianas de um ponto
10   } Ponto;
11
12   typedef struct {
13       char nome[10];
14       Ponto *pontos;    //pontos da figura
15       int np;       //numero de pontos
16   } Figura;
```

```
47   //desaloca espaço da figura
48   void ApagaFigura(Figura *f) {
49
50       if(f->pontos!=NULL) {
51           free(f->pontos);
52           if(f!=NULL)
53               free(f);
54       }
55   }
56
57   //le pontos da figura
58   void LeFigura(Figura *f) {
59       int i;
60
61       printf("\nLendo os pontos da figura %s",f->nome);
62       for(i=0;i<f->np;i++) {
63           printf("\nInforme o ponto [%d]",i+1);
64           printf("\nX: "); scanf("%f",&f->pontos[i].x);
65           printf("\nY: "); scanf("%f",&f->pontos[i].y);
66       }
67   }
68
69   //le pontos da figura
70   void ImprimeFigura(Figura *f) {
71       int i;
72
73       printf("\nImprimindo os pontos da figura %s",f->nome);
74       for(i=0;i<f->np;i++) {
75           printf("\nO ponto [%d] X:%f e Y:%f",i+1,f->pontos[i].x,f->pontos[i].y);
76       }
```

Exercícios

1-Crie uma estrutura chamada DadosAluno, que armazena a média e idade de um aluno. Na função *main*: criar uma variável que é uma estrutura para DadosAluno; ler a média e a idade de um aluno e armazenar na variável criada; exibir na tela os campos da variável criada;

2-Considerando o exercício 1, criar uma variável que é um vetor para a estrutura DadosAluno. O programa deve obter a média e idade de 10 alunos. Depois, estes dados devem ser exibidos.



Exercícios

3-Considerando o exercício 1, criar uma variável ponteiro para a estrutura DadosAluno. O programa deve usar esse ponteiro para obter a média e idade de 15 alunos. Depois, estes dados devem ser exibidos. Não esqueça que no final do programa a memória alocada deve ser liberada.

4-Considerando o exercício 3, desenvolver uma solução aplicando rotinas.