Algoritmos e Programação B

Ponteiros

Estruturas, Funções e Alocação Dinâmica de Memória

Estruturas e Funções

- Passagem de estruturas por referência a funções
  - Devemos utilizar ponteiros
- Seja a estrutura:

```
struct aluno {
        char nome[80];
        char curso[60];
        int anoIngresso;
        int anoFormatura;
};
```

Estruturas e Funções

- E seja a função *main*, com a declaração da variável `a`, do tipo `struct`.

```c
int main(){
    struct aluno a;

    lerAluno(&a);

    printf("Nome: %s\n", a.nome);
    printf("Curso: %s\n", a.curso);
    printf("Ano de ingresso: %d\n", a.anoIngresso);
    printf("Previsão de formatura: %d\n", a.anoFormatura);

    return 0;
}
```

Estruturas e Funções

- A função `lerAluno` recebe um argumento que é o endereço da variável `a` (do tipo `struct`).

```
int main(){
    struct aluno a;

    lerAluno(&a);

    printf("Nome: %s\n", a.nome);
    printf("Curso: %s\n", a.curso);
    printf("Ano de ingresso: %d\n", a.anoIngresso);
    printf("Previsão de formatura: %d\n", a.anoFormatura);

    return 0;
}
```

Estruturas e Funções

- Para acessar cada elemento da struct, na função `lerAluno` (ou em qualquer função que tem como parâmetro, um ponteiro para `struct`), usamos **->**

```
void lerAluno(struct aluno *p){
    printf("Nome: ");
    gets(p->nome);
    printf("Curso: ");
    fflush(stdin);
    gets(p->curso);
    printf("Ano de ingresso: ");
    scanf("%d", &p->anoIngresso);
    printf("Previsão de formatura em: ");
    scanf("%d", &p->anoFormatura);
    return ;
}
```

Estruturas e Funções

- Para acessar cada elemento da struct, na função `lerAluno` (ou em qualquer função que tem como parâmetro, um ponteiro para `struct`), usamos **->**

```
void lerAluno(struct aluno *p){
    printf("Nome: ");
    gets(p->nome);
    printf("Curso: ");
    fflush(stdin);
    gets(p->curso);
    printf("Ano de ingresso: ");
    scanf("%d", &p->anoIngresso);
    printf("Previsão de formatura em: ");
    scanf("%d", &p->anoFormatura);
    return ;
}
```

Vetor de Estruturas e Funções

Vetor de Estruturas e Funções

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

struct aluno {
    char nome[80];
    char curso[60];
    int anoIngresso;
    int anoFormatura;
};

void lerAluno(struct aluno *p){
    printf("Nome: ");
    fflush(stdin);
    gets(p->nome);
    printf("Curso: ");
    fflush(stdin);
    gets(p->curso);
    printf("Ano de ingresso: ");
    scanf("%d", &p->anoIngresso);
    printf("Previsão de formatura em: ");
    scanf("%d", &p->anoFormatura);
    return ;
}

```

Vetor de Estruturas e Funções

```
25  int main(){
26      struct aluno a[3];
27      int i;
28
29      for(i=0; i<3; i++){
30          lerAluno(&a[i]);
31      }
32
33      for(i=0; i<3; i++){
34         printf("----------------------------------------\n");
35         printf("Nome: %s\n", a[i].nome);
36         printf("Curso: %s\n", a[i].curso);
37         printf("Ano de ingresso: %d\n", a[i].anoIngresso);
38         printf("Previsão de formatura: %d\n", a[i].anoFormatura);
39      }
40
41      return 0;
42  }
43
```

Estruturas e Alocação Dinâmica de Memória

Estruturas e Alocação Dinâmica

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

struct aluno {
    char nome[80];
    char curso[60];
    int anoIngresso;
    int anoFormatura;
};

void lerAluno(struct aluno *p){
    printf("Nome: ");
    fflush(stdin);
    gets(p->nome);
    printf("Curso: ");
    fflush(stdin);
    gets(p->curso);
    printf("Ano de ingresso: ");
    scanf("%d", &p->anoIngresso);
    printf("Previsão de formatura em: ");
    scanf("%d", &p->anoFormatura);
    return ;
}
```

Estruturas e Alocação Dinâmica

```
25  int main(){
26      struct aluno a, *ptr;
27
28      ptr = (struct aluno *) malloc(sizeof(struct aluno));
29
30      if(!ptr) {
31          printf("Problemas Alocação\n");
32          exit(1);
33      }
34
35      lerAluno(ptr);
36
37      printf("Nome: %s\n", ptr->nome);
38      printf("Curso: %s\n", ptr->curso);
39      printf("Ano de ingresso: %d\n", ptr->anoIngresso);
40      printf("Previsão de formatura: %d\n", ptr->anoFormatura);
41
42      return 0;
43  }
44
```

Lista Encadeada

Exemplo: Lista Encadeada

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

struct aluno {
    char nome[80];
    char curso[60];
    int anoIngresso;
    int anoFormatura;
    struct aluno *prox;
};

void lerAluno(struct aluno *p){
    printf("Nome: ");
    fflush(stdin);
    gets(p->nome);
    printf("Curso: ");
    fflush(stdin);
    gets(p->curso);
    printf("Ano de ingresso: ");
    scanf("%d", &p->anoIngresso);
    printf("Previsão de formatura em: ");
    scanf("%d", &p->anoFormatura);
    return ;
}
```

Exemplo: Lista Encadeada

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

struct aluno {
    char nome[80];
    char curso[60];
    int anoIngresso;
    int anoFormatura;
    struct aluno *prox;
};

void lerAluno(struct aluno *p){
    printf("Nome: ");
    fflush(stdin);
    gets(p->nome);
    printf("Curso: ");
    fflush(stdin);
    gets(p->curso);
    printf("Ano de ingresso: ");
    scanf("%d", &p->anoIngresso);
    printf("Previsão de formatura em: ");
    scanf("%d", &p->anoFormatura);
    return ;
}
```

Exemplo: Lista Encadeada

```c
int main(){
    struct aluno *pa, *pinicio, *pnew;
    char letra = 's';
    int i = 0;

    while (letra == 's' || letra == 'S'){
       pnew = (struct aluno *) malloc(sizeof(struct aluno));

       lerAluno(pnew);

       pnew->prox = NULL;

       if (i==0){
                 pinicio = pnew;
                 pa = pnew;
       }
       else pa->prox = pnew;

       pa = pnew;
       i++;

       printf("Digitar dados de outro aluno (S/N)?\n");
       fflush(stdin);
       scanf("%c", &letra);
    }
```

Exemplo: Lista Encadeada

```
71
72          printf("Digitar dados de outro aluno (S/N)?\n");
73          fflush(stdin);
74          scanf("%c", &letra);
75      }
76
77      printf("Foram inseridos %d alunos:\n", i);
78      listarAlunos(pinicio);
79      printf("-----------------------------------\n", i);
80      printf("Destruindo a lista...\n");
81      destruirLista(pinicio);
82      return 0;
83  }
84
```

Exemplo: Lista Encadeada

```
26
27  void listarAlunos(struct aluno *p){
28      struct aluno *ptr;
29      ptr = p;
30
31      while(ptr != NULL){
32          printf("\n\n%s cursa %s e ingressou no ano %d\n", ptr->nome, ptr->curso, ptr->anoIngresso);
33          ptr = ptr->prox;
34      }
35      return ;
36  }
```

Exemplo: Lista Encadeada

```
26
27  void listarAlunos(struct aluno *p){
28      struct aluno *ptr;
29      ptr = p;
30
31      while(ptr != NULL){
32          printf("\n\n%s cursa %s e ingressou no ano %d\n", ptr->nome, ptr->curso, ptr->anoIngresso);
33          ptr = ptr->prox;
34      }
35      return ;
36  }
```

Exemplo: Lista Encadeada

```
38  void destruirLista(struct aluno *p){
39      struct aluno *aux;
40
41      while(p != NULL){
42          aux = p;
43          p = p->prox;
44          free(aux);
45      }
46      printf("Lista destruída\n");
47      return ;
48  }
49
```

Exercício: utilize Alocação Dinâmica, Ponteiros, Funções e Estruturas

Considere que em uma estética automotiva, é necessário implementar um programa que controle a entrada e saída de veículos. Ao chegar um veículo, são registrados a placa, o nome do proprietário, o telefone do proprietário, o tipo de serviço, o valor a ser pago e a situação (pago ou não pago).
Construa o código para este programa usando uma função para obter os dados do veículo, outra para registrar o pagamento e outra para mostrar todos os dados do veículo.