

Algoritmos II



STRUCTS (ESTRUTURAS)

Conceito



- “Tipo” criado pelo programador que permite a armazenagem de dados de tipos diferentes.
- As structs consistem em criar apenas um dado que contém vários membros, que nada mais são do que outras variáveis. De uma forma mais simplificada, é como se uma variável tivesse outras variáveis dentro dela.
- Struct, ou estrutura, é um bloco que armazena diversas informações.

Vantagem de Utilização



- Pode-se agrupar de forma organizada vários tipos de dados diferentes. Por exemplo, dentro de uma estrutura de dados pode-se ter juntos um tipo float, um inteiro, um char ou um double.

```
1  #include <stdio.h>
2
3  struct Jogador {
4      char nomeDoJogador[20];
5      int forca, destreza, inteligencia;
6      //E os outros itens que fossem necessários
7  };
8
9  int main() {
10
11      struct Jogador j1, j2; //E quantos jogadores fossem necessários
12
13      return 0;
14
15  }
```

Utilizando struct é possível agrupar as informações. Caso não fossem utilizadas, seria necessário replicar as variáveis criadas para cada jogador (nomeDoJogador1, forca1, destreza1, inteligencia, nomeDoJogador2, ...). Utilizando struct são agrupadas as informações comuns, e utilizada somente a struct na chamada.

Onde utilizar as Estruturas



- Structs são muito usadas quando tem-se elementos nos programas que precisam e fazem uso de vários tipos de variáveis e características.
- Usando struct, pode-se trabalhar com vários tipos de informações de uma maneira mais fácil, rápida e organizada, uma vez que não é necessário se preocupar em declarar e decorar o nome de cada elemento da struct.

Onde utilizar as Estruturas



- **Exemplo:**

- Vamos supor que você foi contratado para criar um aplicativo de uma escola. As structs servem para organizar as informações de uma maneira mais otimizada. Para isso, basta colocar as informações comuns na estrutura.
 - ✦ Quais seriam os elementos comuns que deveriam ser colocados nessa estrutura?
 - Como vai-se trabalhar com alunos, então é necessário colocar elementos na struct que representem os alunos: nome, notas, mensalidade, se esta mensalidade foi paga ou não etc. Assim, pode-se criar uma variável struct para cada aluno, ou um vetor de struct e, automaticamente, esse aluno terá as variáveis acima citadas.

Sintaxe



- Para criar uma estrutura de dados é utilizada a palavra reservada `struct`.
- Toda estrutura deve ser criada antes de qualquer função ou mesmo da função principal `main`.
- Toda estrutura tem nome e seus membros são declarados dentro de um bloco de dados.
- Após a definição de seus membros no bloco de dados, termina-se a linha com um ponto-e-vírgula(`;`).

Sintaxe



Nome da Estrutura

Componentes da Estrutura

```
struct <nome> {  
    membro;  
    ...  
};
```

```
struct produto{  
    int cod;  
    char nome[100];  
};
```

Sintaxe



```
1  #include <stdio.h>
2
3  struct Mercado {
4      int codigo;
5      char nomeDoProduto[10];
6      float preco;
7  };
8
9  int main() {
10
11      struct Mercado prateleira1;
12
13      return 0;
14
15  }
```

A struct Mercado contém todas as variáveis referentes as informações do Mercado.

Para utilizar é necessário criar uma variável do "tipo" Mercado (No C a criação do tipo deve vir acompanhada da palavra reservada struct).

Sintaxe



```
1  #include <stdio.h>
2
3  struct Mercado {
4      int codigo;
5      char nomeDoProduto[10];
6      float preco;
7  };
8
9  int main() {
10
11      struct Mercado prateleira1;
12
13      printf("Código: ");
14      scanf("%d", &prateleira1.codigo);
15      printf("Nome do Produto: ");
16      scanf("%s", &prateleira1.nomeDoProduto);
17      printf("Preço: ");
18      scanf("%f", &prateleira1.preco);
19
20      return 0;
21 }
```

Para acessar um membro da struct chama-se pelo nome da variável criada, ponto, o nome da variável a ser acessada na struct.

variavelCriada.variavelStruct
prateleira1.preco

Estruturas



- Uma struct pode ser tanto global quanto local. A struct local será válida somente na função onde foi declarada, e a struct global por todas as funções abaixo de sua declaração.
- Uma struct pode ser atribuída a outra do mesmo tipo.

```
1  #include <stdio.h>
2
3  struct Mercado {
4      int codigo;
5      char nomeDoProduto[10];
6      float preco;
7  };
8
9  int main() {
10
11      struct Mercado prateleira1 = {1, "Biscoito", 4.50};
12
13      struct Mercado prateleira2 = prateleira1;
14
15      printf("%s", prateleira2.nomeDoProduto);
16
17      return 0;
18  }
```

Vetores



- É possível criar vetores cujos elementos são estruturas.

```
1
2  #include <stdio.h>
3
4  #define TAM 5
5
6  struct Mercado {
7      int codigo;
8      char nomeDoProduto[10];
9      float preco;
10 };
11
12 int main() {
13
14     struct Mercado produtos[TAM];
15
16     for(int i=0; i<TAM; i++) {
17         printf("Código: ");
18         scanf("%d", &produtos[i].codigo);
19         printf("Produto: ");
20         scanf("%s", &produtos[i].nomeDoProduto);
21         printf("Preço: ");
22         scanf("%f", &produtos[i].preco);
23     }
24     return 0;
25 }
```

- Quando se cria um vetor do tipo de uma estrutura, em cada índice do vetor pode-se encontrar todos os dados da estrutura. Por exemplo, na estrutura abaixo:

<ul style="list-style-type: none">▪ codigo▪ nomeDo Produto▪ preco	<ul style="list-style-type: none">▪ codigo▪ nomeDo Produto▪ preco	<ul style="list-style-type: none">▪ codigo▪ nomeDo Produto▪ preco	<ul style="list-style-type: none">▪ codigo▪ nomeDo Produto▪ preco	<ul style="list-style-type: none">▪ codigo▪ nomeDo Produto▪ preco
0	1	2	3	4

Vetor do tipo estrutura x Vetor dentro da estrutura

- Quando se cria um vetor do tipo estrutura em cada índice do vetor estão as informações da estrutura (no caso, em cada índice tem-se uma informação de código, nomeDoProduto e preço).

```
1  #include <stdio.h>
2
3  #define TAM 5
4
5  struct Mercado {
6      int codigo;
7      char nomeDoProduto[10];
8      float preco;
9  };
10
11 int main() {
12
13     struct Mercado produtos[TAM];
14
15     return 0;
16 }
```

- Quando se cria um vetor dentro da estrutura é como uma variável da estrutura. No caso, tem-se uma variável produtos, e dentro dela será cadastrado um código, um nome e 5 preços desse produto.

```
1  #include <stdio.h>
2
3  #define TAM 5
4
5  struct Mercado {
6      int codigo;
7      char nomeDoProduto[10];
8      float preco[TAM];
9  };
10
11 int main() {
12
13     struct Mercado produtos;
14
15     return 0;
16 }
```

Estruturas Aninhadas

- Podem ser criadas estruturas aninhadas, ou seja, uma estrutura dentro da outra.

Foram criadas duas variáveis do tipo data. No formato normal seriam criadas 3 variáveis para entrega (dia, mes e ano) e 3 variáveis para validade (dia, mês e ano).

Em caso de uma estrutura aninhada a outra, a forma de chamada é:

variável.variavelStruct1.variavelStruct2

mercado1.entrega.dia
Mercado1.validade.dia

```
1  #include <stdio.h>
2
3  struct data {
4      int dia, mes, ano;
5  };
6
7  struct Produto {
8      int codigo;
9      struct data entrega;
10     struct data validade;
11 };
12
13 int main() {
14
15     struct Produto mercado1;
16     printf("Código: ");
17     scanf("%d", &mercado1.codigo);
18
19     printf("Data de Entrega: ");
20     scanf("%d", &mercado1.entrega.dia);
21     scanf("%d", &mercado1.entrega.mes);
22     scanf("%d", &mercado1.entrega.ano);
23
24     printf("Data de Validade: ");
25     scanf("%d", &mercado1.validade.dia);
26     scanf("%d", &mercado1.validade.mes);
27     scanf("%d", &mercado1.validade.ano);
28
29     return 0;
30 }
```

Estruturas



- Uma das desvantagens na utilização de estruturas em C está na declaração das variáveis, que tem sempre que ser precedidas da palavra reservada "struct", seguida do nome da estrutura.
- O ideal seria podermos representar a estrutura diretamente através de uma palavra como ocorre com os tipos primitivos da linguagem.
- Isso é possível através da palavra reservada "typedef" que segue a seguinte sintaxe:
 - typedef tipo_existente sinônimo;
 - A palavra typedef não cria um novo tipo, apenas permite que um determinado tipo possa ser denominado de forma diferente.
 - Exemplo:
 - ✦ typedef int inteiro
 - ✦ Permite que o tipo int possa ser representado pela palavra inteiro, permitindo a declaração utilizando ambas as palavras:
 - int a, b, c;
 - ou
 - inteiro a,b,c;

Passagem de Parâmetro e Retorno de Função



```
1  #include <stdio.h>
2
3  typedef struct {
4      int a, b, res;
5  } teste;
6
7
8  teste passagem_referencia(teste *y) {
9      (*y).a = (*y).a + (*y).b;
10     return (*y);
11 }
12
13
14 void imprime (teste y) {
15     printf("%d\t%d", y.a, y.b);
16 }
17
18 int main()
19 {
20     teste x;
21     x.a = 2;
22     x.b = 4;
23     imprime(passagem_referencia(&x));
24     return 0;
25 }
```

- Pode-se passar uma variável do tipo estrutura por parâmetro, bem como uma função pode retornar uma estrutura.