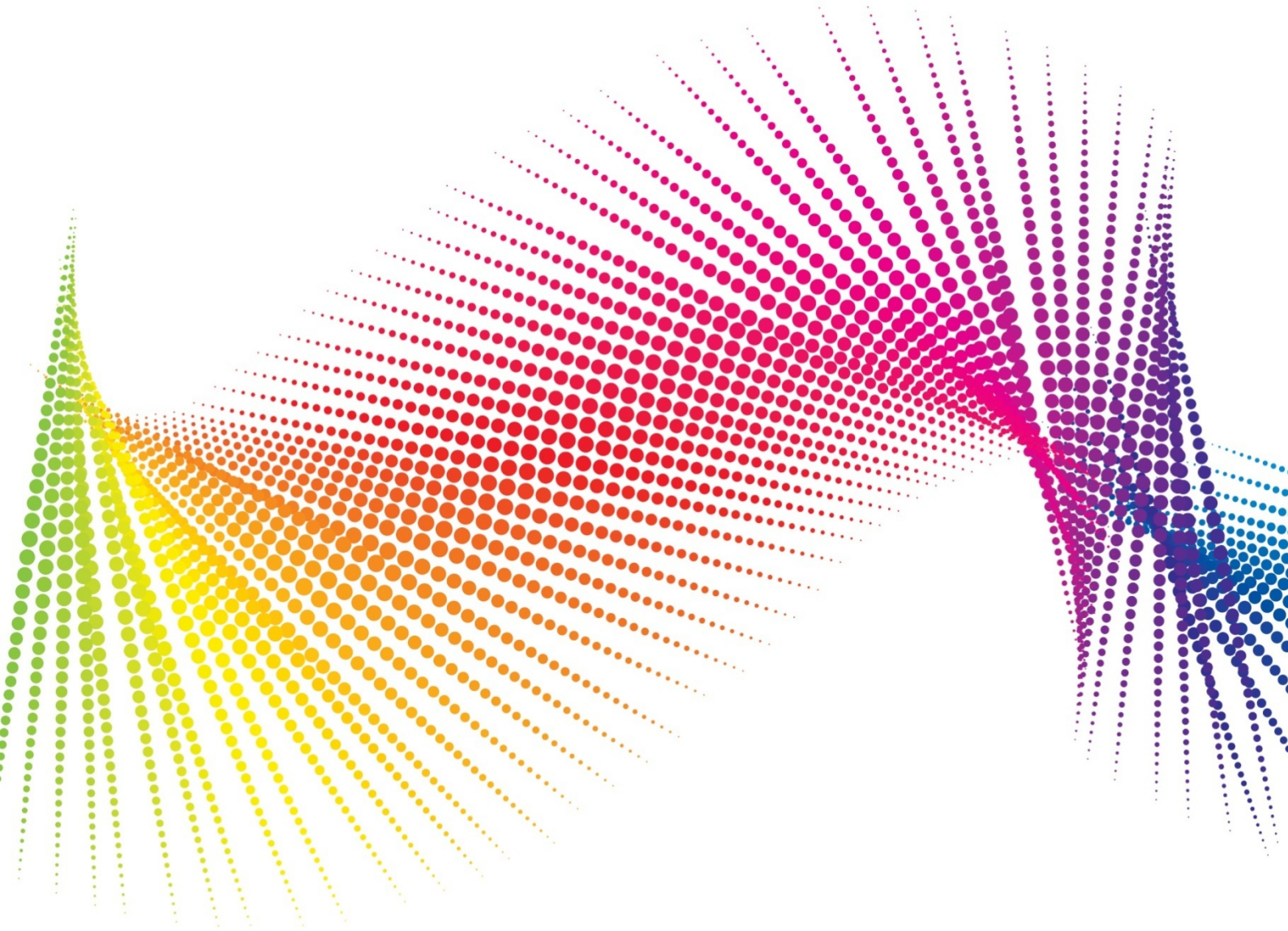
UNINOVE
Universidade Nove de Julho

Estrutura de Dados

Aula 14



Os direitos desta obra foram cedidos à Universidade Nove de Julho

Este material é parte integrante da disciplina oferecida pela UNINOVE.
O acesso às atividades, conteúdos multimídia e interativo, encontros virtuais, fóruns de discussão e a comunicação com o professor devem ser feitos diretamente no ambiente virtual de aprendizagem UNINOVE.

Uso consciente do papel.
Cause boa impressão, imprima menos.

Aula 14: Vetores de estruturas (tabelas)

Objetivo: Estudar as tabelas, ou vetores de estruturas.

Vetores de estruturas

Da mesma forma que se declara um vetor de qualquer um dos tipos básicos, podemos declarar um vetor de estrutura. Assim, formamos um tipo muito interessante e útil para armazenamento de dados: a tabela.

Exemplo:

```
struct amigos  {    char nome[50];
                    char fone[20];
                };

struct amigos agenda[5];
```

A visualização desta tabela seria:

agenda

0	nome		fone	
1	nome		fone	
2	nome		fone	
3	nome		fone	
4	nome		fone	

Para se inicializar a tabela, poderíamos usar um trecho de programa como este:

```
for (i=0; i<5; i++)
{   fflush(stdin);
```

```
    puts("\nForneca o nome da posicao %i: ",i);
    gets(agenda[i].nome);
    puts("\nForneca o telefone da posicao %i: ",i);
    gets(agenda[i].fone);
}
```

Inicialização de vetores de estruturas

A inicialização de vetores de estruturas pode ser feita na declaração ou campo a campo, seguindo as regras de inicialização de estruturas que já estudamos.

Exemplo:

```
struct informacoes  {   char nome[30];
                        int dianascimento;
                        int mesnascimento;
                        int anonascimento;
                    };
```

Inicialização na declaração:

```
struct informacoes tabela[3] = {  {“Jose”, 25, 1, 1970},
                                  {“Maria”,16, 7, 1990},
                                  {“Joao”, 11, 9, 2001},
                                };
```

Inicialização campo a campo:

```
strcpy(tabela[0].nome,“Jose”);
tabela[0].dia = 25;
tabela[0].mes = 1;
tabela[0].ano = 1970;

strcpy(tabela[1].nome,“Maria”);
tabela[1].dia = 16;
tabela[1].mes = 7;
tabela[1].ano = 1990;

strcpy(tabela[2].nome,“Joao”);
tabela[2].dia = 11;
```

```
tabela[2].mes = 9;
tabela[2].ano = 2001;
```

Visualização da tabela:

tabela

0	nome	José	dia	25	mês	1	ano	1970
1	nome	Maria	dia	16	mês	7	ano	1990
2	nome	João	dia	11	mês	9	ano	2001

Exemplo de programa usando vetor de estrutura

Caro aluno, vamos criar uma aplicação que cria uma pequena agenda com o nome e o telefone de nossos amigos.

Aproveitaremos a oportunidade e introduzir mais um recurso que a linguagem C oferece: o uso de arquivos de cabeçalho, ou **arquivos .h[1] (header).**

Neste exemplo da agenda com nomes e telefones de amigos que vamos construir, criamos um arquivo chamado **agenda.h**, em que colocamos as inclusões de bibliotecas e declaramos algumas variáveis que o programa agenda.c vai usar. Já no arquivo agenda.c basta apenas incluirmos o agenda.h que podemos usar tudo o que declaramos nele.

Olha como fica prático! Num arquivo .h podemos incluir as bibliotecas, declarar variáveis, declarar constantes, incluir protótipos e/ou códigos de funções. Você pode criar um arquivo com as bibliotecas e sempre incluí-lo em seus programas .c para evitar ter que declarar sempre o cabeçalho dos programas.

```
/* agenda.h: arquivo cabecalho de agenda.c */

#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<string.h>
#define limpatela(); system("cls");

struct amigos
       {    char nome[50];
            char fone[20];
       };
```

struct amigos agenda[100]; //agenda eh uma tabela ou vetor de estruturas

---

```
/*agenda.c: Ilustra o uso de estruturas,vetores de  estruturas
                e inclusao de arquivos .h   */
#include "agenda.h"

  void montaagenda(int *amigos);
  void mostraagenda(int amigos);

int main()
{ int opcao;
  int numero_de_amigos=0;

  system("color 1e");
  for(;;)
  {
     limpatela();
     puts("\n_____________________________________________________");
     puts("\nMonta uma agenda com nomes e telefones de amigos");
     puts("\n_____________________________________________________");
     puts("\nEscolha uma das opcoes:\n");
      puts("\n0) Sair do programa.");
      puts("\n1) Inserir dados na agenda de amigos.");
      puts("\n2) Ver as informacoes contidas na sua agenda.");
      printf("\nSua opcao: ");
      scanf("%i",&opcao);

       switch(opcao)
       {   case 0:
                 printf("\nFim do programa");
                 getch();
                 exit(0);
                 break;
           case 1:
                 montaagenda(&numero_de_amigos);
                 getch();
                 break;
           case 2:
                 mostraagenda(numero_de_amigos);
                 getch();
                 break;
           default:
                 puts("\nVoce nao escolheu uma opcao valida!");
                 getch();
        }//switch
```

```
    }//for
 }
void montaagenda(int *conta_amigo)
{  char nome[50];

    for (;;)
    { fflush(stdin);
       puts("\nDigite o nome do seu amigo ou . para finalizar: ");
      gets(nome);
      if (strcmp(nome,".") ==0)
            break;
      strcpy(agenda[*conta_amigo].nome, nome);
      puts("\nForneca o telefone do seu amigo: ");
      gets(agenda[*conta_amigo].fone);
      ++(*conta_amigo);
  }

}

void mostraagenda(int numero_de_amigos)
{  int i;
   if (numero_de_amigos == 0)
      printf("\nSua agenda de amigos esta vazia....");
   else
   {
        puts("\nInformacoes sobre seus amigos:\n");
        for(i=0; i<numero_de_amigos; i++)
        {
             printf("\nNome: %s\n", agenda[i].nome);
             printf("Telefone: %s\n", agenda[i].fone);

        }
   }

}
```

Observações sobre o programa agenda.c:

1. Usa arquivo agenda.h (header).

2. `agenda` é um vetor de estruturas.

3. A função montaagenda() recebe um inteiro como parâmetro passado por referência, por isso usamos * na frente do parâmetro dentro do código da função e usamos & na sua chamada. A variável `numero_de_amigos` precisa manter seu

valor, quando for alterada na função montaagenda(), na função main(), por isso foi passada por referência.

Saiba mais

(1) Uso de arquivos de cabeçalho (.h)

O uso de arquivos do tipo header é muito útil. Neste arquivo, que é separado do arquivo fonte com o código do programa C, podemos inserir:

- Declarações de protótipos de funções.
- Declarações de variáveis globais.
- Definições de macros.

Os arquivos de cabeçalho definidos pelo programador têm por convenção a extensão **.h**, da mesma forma que os arquivos de cabeçalho que acompanham a biblioteca C.

Devem ser incluídos no início do arquivo **.c** que tem o código fonte.

Sintaxe:

```
#include “arquivo.h”
```

(Neste caso, a procura pelo arquivo começa normalmente onde o programa fonte se encontra.)

ou

```
#include <arquivo.h>
```

(Neste caso, a procura pelo arquivo segue uma regra definida pela implementação do compilador.)

Agora, caro aluno, vamos praticar resolvendo os exercícios propostos no AVA. **Leia a lista, resolva os exercícios e verifique seu conhecimento**. Caso fique alguma dúvida, leve a questão ao Fórum e divida com seus colegas e professor.

REFERÊNCIAS

SCHILDT, H. *C Completo e Total*. São Paulo: Makron Books, 1997.

TENEMBAUM, Aaron M., et al. *Estruturas de dados usando C*. São Paulo: Pearson Makron Books, 1995.