Aula – Programação II (INF 09330)

UFES
Informática

Tipos de Dados Estruturados Definidos pelo Usuário

Prof. Thiago Oliveira dos Santos
Departamento de Informática
Universidade Federal do Espírito Santo

2015

Visão Geral da Aula

UFES
Informática

- Introdução
- Estrutura de Dados (*struct*)
- Introdução a Tipo Abstrato de Dados
- Separação de Cabeçalho e Corpo de Função
- Pré-Processador

Introdução

UFES
Informática

Tipos de Básicos de Dados

- São limitados aos dados básicos da linguagem
  - Inteiro, real, literal, etc.
- Problemas do dia a dia envolvem dados mais complexos
  - Exemplos
    - Livros de uma biblioteca (titulo, gênero, autor(es), ISBN, etc.)
    - Carros de uma concessionária (ano, modelo, cor, etc.)
    - Alunos de uma escola (matrícula, nome, endereço, etc)
- Tipos de dados simples não suportam essa abstração

Introdução

UFES
Informática

Exemplo

- Estudantes da UFES
  - Idade -> int
  - Matricula -> int
  - Coeficiente -> float
  - Periodo -> int
- Para armazenar os estudantes
  - Várias variáveis são necessárias

```
int  idade;
int  matricula;
float coeficiente;
int  periodo;
```

Introdução

UFES
Informática

Problemas

- Programador deve cuidar dos atributos de um estudante
  - Variáveis separadas dificultam a legibilidade
  - Programador deve lembrar como as variáveis se relacionam

Introdução

UFES
Informática

Solução

- Tipos de dados definidos pelo usuário

Tipos de Dados Definidos pelo Usuário

- São flexíveis quanto ao tipo de problema
- O programador é responsável por defini-lo
- É conhecido como estrutura de dados (em C, struct)
- Conceito básico em programação

Estrutura de Dados

UFES
Informática

Características

- Permitem a abstração dos dados
  - Agrupamento
- Os campos da estrutura são denominados atributos
- Os elementos são geralmente logicamente relacionados

Estrutura de Dados em C

- Definida com o comando *struct*

Estrutura de Dados

UFES
Informática

Declaração do Tipo Complexo

- Sintaxe

```
typedef struct {
    <tipo> <nome_do_atributo_1>;
    <tipo> <nome_do_atributo_2>;
    <tipo> <nome_do_atributo_3>;
            ...
} <identificador>;
```

- <identificador>
  - Identificador do tipo complexo
- <tipo>
  - Tipo básico do atributo
- <nome_do_atributo_1>
  - Identificador do atributo

Estrutura de Dados

UFES
Informática

Exemplo

- Estudantes da UFES

```
typedef struct {
   int idade;
   int matricula;
   float coeficiente;
   int periodo;
} tEstudante;
```

Estrutura de Dados

UFES
Informática

Acesso aos Membros

- Utiliza-se o operador "."
  - Sintaxe: <nome_da_variavel>.<nome_do_atributo>

```
typedef struct {
   int idade;
   int matricula;
   float coeficiente;
   int periodo;
} tEstudante;

int main()
{
   tEstudante aluno;
   scanf("%d %d %f %d", &aluno.idade,
            &aluno.matricula, &aluno.coeficiente, &aluno.periodo);

   printf("Imprimindo aluno: %d %d %f %d\n",
            aluno.idade, aluno.matricula, aluno.coeficiente, aluno.periodo);
   return 0;
}
```

Estrutura de Dados

UFES
Informática

Inicialização

- Utiliza-se {}

```
typedef struct {
   int idade;
   int matricula;
   float coeficiente;
   int periodo;
} tEstudante;

int main()
{
   tEstudante aluno = { 20, 1001, 7.0, 5};
   printf("Imprimindo aluno: %d %d %f %d\n",
            aluno.idade, aluno.matricula, aluno.coeficiente, aluno.periodo);

   return 0;
}
```

Estrutura de Dados

UFES
Informática

Estruturas com Membros Compostos

- O membro de uma estrutura também pode ser uma estrutura

```
typedef struct {
    int telefoneFixo;
    int telefoneCelular;
} tContatos;

typedef struct {
    int idade;
    int matricula;
    float coeficiente;
    int periodo;
    tContatos contato;
} tEstudante;
```

Estrutura de Dados

UFES
Informática

Atribuição de Estruturas

- Utiliza-se o operador “=“ como para variáveis comuns
- É feita uma cópia de todos os membros
- Cópia é recursiva para todos os membros *struct* (que são *struct*)

```
void main()
{
   tEstudante aluno, outroAluno;
   scanf("%d %d %f %d", &aluno.idade,
            &aluno.matricula, &aluno.coeficiente, &aluno.periodo);

   outroAluno = aluno;
   printf("Imprimindo aluno: %d %d %f %d\n",
            aluno.idade, aluno.matricula, aluno.coeficiente, aluno.periodo);
   printf("Imprimindo outro aluno: %d %d %f %d\n",
            outroAluno.idade, outroAluno.matricula,
            outroAluno.coeficiente, outroAluno.periodo);
}
```

Estrutura de Dados

UFES
Informática

Estruturas como Parâmetro de Função

- Funções facilitam reuso, modularização e agrupamento de código
- Pode-se passar variáveis complexas como argumento para função
- Simplifica a passagem de parâmetros
- Funciona como a atribuição (passagem por valor)
  - Todos os campos da estrutura passada como argumento são copiados para a estrutura declarada como parâmetro

```
void ImprimeEstudante( tEstudante a_estudante )
{
  printf("Estudante:\n");
  printf("\tIdade -> %d\n", a_estudante.idade);
  printf("\tMatricula -> %d\n", a_estudante.matricula);
  printf("\tCoeficiente -> %.2f\n", a_estudante.coeficiente);
  printf("\tPeiodo -> %d\n", a_estudante.periodo);
}
```

Estrutura de Dados

UFES
Informática

Estruturas como Parâmetro de Função

```
void ImprimeEstudante( tEstudante a_estudante )
{
   printf("Estudante:\n");
   printf("\tIdade -> %d\n", a_estudante.idade);
   printf("\tMatricula -> %d\n", a_estudante.matricula);
   printf("\tCoeficiente -> %.2f\n", a_estudante.coeficiente);
   printf("\tPeiodo -> %d\n", a_estudante.periodo);
};
int main()
{
    tEstudante aluno, outroAluno;
    scanf("%d %d %f %d",
          &aluno.idade, &aluno.matricula, &aluno.coeficiente, &aluno.periodo);
    outroAluno = aluno;
    ImprimeEstudante(aluno);
    ImprimeEstudante(outroAluno);
    return 0;
}
```

Introdução a Tipo Abstrato de Dados

UFES
Informática

Motivação

- Permitir que o programador (usuário do tipo criado)
  - Utilize o tipo como se fosse um tipo básico de dados

Até Agora

- Só definiu-se novas estruturas de dados
- Porém, Tipos Básicos de Dados definem
  - Forma de armazenamento dos dados
  - Operações que podem ser realizadas nos dados
- Portanto, um Tipos Abstrato de Dados deve definir
  - Forma de armazenamento dos dados *(struct)*
  - Operações que podem ser realizadas nesse no tipo

Introdução a Tipo Abstrato de Dados

UFES
Informática

Definição

- Quando definimos um tipo abstrato de dados
  - Estamos preocupado com “o que ele faz?”
  - E não com “como ele faz?”
- Programas que usam o tipo são chamados de
  - *Clientes*
- Programas que definem o tipo são conhecidos como
  - *Implementação do tipo*

Introdução a Tipo Abstrato de Dados

UFES
Informática

Definição

- Exemplo do tipo *tData*
  - Atributos
    - dia, mês, ano
  - Operações
    - LeData
    - InicializaDataParam
    - ImprimeData
    - EhBissexto
    - InformaQtdDiasNoMes
    - AvancaParaDiaSeguinte
    - EhIgual

Introdução a Tipo Abstrato de Dados

UFES
Informática

Uso

- Exemplo do uso do tipo *tData*

```
int main()
{
   tData data;
   tData dataIni;
   tData dataFim;

   dataIni = LeData();
   dataFim = LeData();

   for(    data = dataIni;
           !EhIgual(data, dataFim);
           data = AvancaParaDiaSeguinte(data)){
      ImprimeData(data);
      printf("\n");
   }
   return 0;
}
```

Introdução a Tipo Abstrato de Dados

UFES
Informática

Definição

- Exemplo do tipo *tData*

```
typedef struct {
    int dia;
    int mes;
    int ano;
} tData;
```

Introdução a Tipo Abstrato de Dados

UFES
Informática

Definição

- Exemplo do tipo *tData*
  - InicializaDataParam (Garante que a data seja válida)

```
tData InicializaDataParam( int a_dia, int a_mes, int a_ano ) {
   tData data;
   int qtdDiasNoMes;
   data.ano = a_ano;
   if (a_mes > 12){
      a_mes = 12;
   } else if (a_mes < 1){
      a_mes = 1;
   }
   data.mes = a_mes;
   qtdDiasNoMes = InformaQtdDiasNoMesMA(a_mes, a_ano);
   if ( a_dia > qtdDiasNoMes ){
      a_dia = qtdDiasNoMes;
   } else if (a_dia < 1){
      a_dia = 1;
   }
   data.dia = a_dia;
   return data;
}
```

Introdução a Tipo Abstrato de Dados

UFES
Informática

Definição

- Exemplo do tipo *tData*
  - LeData (Garante que a data seja válida)

```
tData LeData( )
{
   tData dataRtn;
   int d, m, a;

   if ( scanf("%d %d %d", &d, &m, &a) != 3 ){
      printf("ERRO: Formato de entrada nao compativel!\n");
      exit(1);
   }

   dataRtn = InicializaDataParam( d, m, a );

   return dataRtn;
}
```

Introdução a Tipo Abstrato de Dados

UFES
Informática

Definição

- Exemplo do tipo *tData*
  - ImprimeData

```
void ImprimeData ( tData a_data )
{
   printf("'%02d/%02d/%04d'", a_data.dia, a_data.mes, a_data.ano);
}
```

Introdução a Tipo Abstrato de Dados

UFES
Informática

Definição

- Exemplo do tipo *tData*
  - EhBissexto

```
int EhBissexto( tData a_data )
{
   if ( !(a_data.ano%400) ){
      return 1;
   }
   if ( !(a_data.ano%100) ){
      return 0;
   }
   if ( !(a_data.ano%4) ){
      return 1;
   }
   return 0;
}
```

Introdução a Tipo Abstrato de Dados

UFES
Informática

Definição

- Exemplo do tipo *tData*
  - InformaQtdDiasNoMes

```
int InformaQtdDiasNoMes( tData a_data )
{
   if ( a_data.mes == 4 || a_data.mes == 6 || a_data.mes == 9 || a_data.mes == 11){
      return 30;
   }
   if ( a_data.mes == 2 ){
      if ( EhBissexto(a_data) ){
         return 29;
      }
      return 28;
   }
   return 31;
}
```

Introdução a Tipo Abstrato de Dados

UFES
Informática

Definição

- Exemplo do tipo *tData*
  - AvancaParaDiaSeguinte

```
tData AvancaParaDiaSeguinte( tData a_data )
{
   if ( a_data.dia < InformaQtdDiasNoMes(a_data) ){
      a_data.dia++;
   } else {
      a_data.dia = 1;
      if ( a_data.mes < 12 ){
         a_data.mes++;
      } else {
         a_data.mes = 1;
         a_data.ano++;
      }
   }
   return a_data;
}
```

Introdução a Tipo Abstrato de Dados

UFES
Informática

Definição

- Exemplo do tipo *tData*
  - EhIgual

```
int EhIgual( tData a_data1, tData a_data2 ){
   return  a_data1.dia == a_data2.dia &&
           a_data1.mes == a_data2.mes &&
           a_data1.ano == a_data2.ano;
}
```

Introdução a Tipo Abstrato de Dados

UFES
Informática

Problemas no Uso

- Exemplo de mau uso do tipo *tData*

```
int main()
{
   tData data;

   data = LeData ();
   ImprimeData (data);
   printf(" Passou um dia! ");
   data.dia++;
   ImprimeData (data);
   printf("\n");

   return 0;
}
```

Utilização incorreta do tipo tData pelo programa cliente!

Gera erros difíceis de detectar!
Deveria ter chamada a função do tipo:
AvancaParaDiaSeguinte

Separação de Cabeçalho e Corpo de Função

UFES
Informática

Por que separar?

- Facilita utilização
- Isola funções desnecessárias
  - Só “publica” as que vão ser re-utilizadas
- Tira o foco da implementação
  - Alguém aqui olhou como funciona o *sqrt* do arquivo *math.h*?
  - Quantos ficaram pensando em como eles implementaram?
  - Esse é o nosso objetivo!!!

Separação de Cabeçalho e Corpo de Função

UFES
Informática

Como separar?

- Agrupar todas as funções de comportamento similar
  - Exemplo
    - Implementação de um tipo
    - Funções destinadas a resolver um tipo de problema
      - Funções de matemática
- Definir nome para o arquivo
  - Exemplo: *tData*
- Criar dois arquivos
  - *tData*.h contendo os protótipos das funções
  - *tData*.c contendo o corpo das funções

Separação de Cabeçalho e Corpo de Função

UFES
Informática

Exemplo do Arquivo .h

- Nome: *tData.h*
- Conteúdo:

```
#ifndef TDATA_H
#define       TDATA_H

typedef struct {
   int dia;
   int mes;
   int ano;
} tData;

tData InicializaDataParam( int a_dia, int a_mes, int a_ano);
tData LeData( );
void ImprimeData( tData a_data );
int EhBissexto( tData a_data );
int InformaQtdDiasNoMes( tData a_data );
tData AvancaParaDiaSeguinte( tData a_data );
int EhIgual( tData a_data1, tData a_data2 );

#endif        /* TDATA_H */
```

Separação de Cabeçalho e Corpo de Função

UFES
Informática

Exemplo do Arquivo .c

- Nome: *tData*.c
- Conteúdo:

```
#include "tData.h"
#include <stdio.h>

void ImprimeData( tData a_data )
{
   printf("'%02d/%02d/%04d'", a_data.dia, a_data.mes, a_data.ano);
}

...
... //Implementacao das outras funcoes
```

Pré-Processador

UFES
Informática

Conceito

- Realiza modificações no código antes da compilação
- Modificações são feitas de acordo com diretivas

Diretivas

- Começam com o símbolo #
- Não devem vir na mesma linha
- Principais
  - #include
  - #define
  - #ifndef
  - #endif
  - #if
  - #else

Pré-Processador

UFES
Informática

#include

- Inclui um arquivo
- #include <nome do arquivo.h>
- #include “nome do arquivo.h”

#define

- Define um símbolo que será substituído por outro
- #define ALGUM_VALOR 423

Pré-Processador

UFES
Informática

#ifndef #endif

- Habilita o código de acordo com uma condição
- #ifndef
  - Código
- #endif

#if #else #endif

- Define um símbolo que será substituído por outro
- #if
  - Código 1
- #else
  - Código 2
- #endif

Perguntas???

UFES
Informática

- Fazer exercícios da lista 4