Treinamento para a Olimpíada Brasileira de Informática

Exercícios envolvendo ordenação

Prof. Ciro Cirne Trindade

Função de Ordenação em C
(1/3)

- O stdlib.h contém funções de busca e ordenação
  - Para ordenação há a função **`qsort()`**
- **`void qsort(void * base, size_t nel, size_t width, int (*compare) (const void *, const void *));`**
  - **`base`**: endereço do 1º elemento do vetor
  - **`nel`**: número de elementos do vetor
  - **`width`**: número de bytes de cada elemento do vetor
  - Último argumento: função de comparação

Função de Ordenação em C
(2/3)

- A função de ordenação tem como parâmetros dois ponteiros para elementos com **`width`** bytes
- A função devolve:
  - Um número negativo se o 1º elemento aparece antes do 2º quando ordenados
  - Um número positivo se o 2º aparece antes do 1º
  - Zero se eles forem iguais

Função de Ordenação em C
(3/3)

- Exemplo:
  - Função que compara 2 inteiros

```
int intcompare(const void * i, const void * j)
{
    int x = *((int *) i);
    int y = *((int *) j);
    if(x < y) return -1;
    if(x > y) return 1;
    return 0;
}
```

  - Para chamar a função qsort:

```
qsort(vetor, tam, sizeof(int), intcompare);
```

Assumindo que **vetor** é um vetor de inteiros de tamanho **tam**

Exercícios

- Vice-campeão: OBI2012 – 1ª Fase – Modalidade Programação, Nível 1
  - http://olimpiada.ic.unicamp.br/pratique/programacao/nivel1/2012f1p1_vice
- Lista de Chamada: OBI2010 – 2ª Fase – Modalidade Programação, Nível 1
  - http://olimpiada.ic.unicamp.br/pratique/programacao/nivel1/2010f2p1_chamada

Estruturas

- Em alguns problemas é necessário ordenar um vetor de estruturas para obter a resposta
- Uma estrutura é um tipo estruturado heterogêneo composto por uma ou mais variáveis, possivelmente de tipos diferentes, colocadas juntas sobre um único nome
- As variáveis que fazem parte de uma estrutura são chamadas de **membros** da estrutura

Definição de uma Estrutura

- Forma geral

```
struct [etiqueta-da-estrutura] {
    tipo membro1;
    tipo membro2;
    ...
};
```

- Exemplo:

```
struct aluno {
    int matricula;
    char nome[41];
    float nota;
};
```

Declaração de uma Variável de Tipo Estrutura

- A definição da estrutura não declara nenhuma variável
- Para declarar uma variável **a** do tipo estrutura **aluno**, fazemos:

```
struct aluno a;
```

- ou

```
struct aluno {
    /* definição dos membros */
} a;
```

Acesso aos Membros da Estrutura

- Para acessar um membro individual de uma estrutura usamos o operador ponto (.)
- Sintaxe:

```
variável-estrutura.nome-do-membro
```

- Exemplos:

```
a.matricula = 201504932;
printf("%.2f", a.nota);
fgets(a.nome, 41, stdin);
```

Vetores de Estruturas (1/2)

- Para declarar um vetor de estruturas, você deve primeiro definir a estrutura, e depois declarar um vetor deste tipo
- Exemplo:

```
struct aluno alunos[100];
```

  - Cada elemento do vetor é uma estrutura do tipo **aluno**
  - O nome **alunos** não é o nome de uma estrutura e sim o nome de um vetor em que os elementos são estruturas

Vetores de Estruturas (2/2)

- Para acessar uma estrutura específica dentro do vetor **`alunos`**, use o índice no nome da variável do tipo vetor

	matricula	nome	nota
alunos[0]	alunos[0].matricula	alunos[0].nome	alunos[0].nota
alunos[1]	alunos[1].matricula	alunos[1].nome	alunos[1].nota
alunos[2]	alunos[2].matricula	alunos[2].nome	alunos[2].nota
.		.	
.		.	
.		.	
alunos[99]	alunos[99].matricula	alunos[99].nome	alunos[99].nota

typedef (1/2)

- A palavra-chave **typedef** fornece um mecanismo para a criação de sinônimos
- Sintaxe:
  - **typedef** tipo sinônimo;
- Os nomes dos tipos de estruturas são definidos frequentemente com **typedef** para criar nomes mais curtos de tipos
- Por exemplo a instrução

```
typedef struct aluno taluno;
```

  - define o novo nome de tipo **taluno** como um sinônimo do tipo **struct aluno**

typedef (2/2)

- Os programadores C usam frequentemente **typedef** para definir um tipo estrutura de modo que não é exigido uma etiqueta da estrutura
- Por exemplo:

```
typedef struct {
    int matricula;
    char nome[41];
    float nota;
} taluno;
```

Assim podemos fazer:
taluno alunos[100];

Exercícios

- Corrida: OBI2011 – 1ª Fase – Modalidade Programação, Nível 1
  - http://olimpiada.ic.unicamp.br/pratique/programacao/nivel1/2011f1p1_corrida
- Times: OBI2010 – 1ª Fase – Modalidade Programação, Nível 1
  - http://olimpiada.ic.unicamp.br/pratique/programacao/nivel1/2010f1p1_times
- Olimpíada: OBI2009 – 2ª Fase – Modalidade Programação, Nível 2
  - http://olimpiada.ic.unicamp.br/pratique/programacao/nivel2/2009f2p2_olimpiada

Referências

- UNICAMP. *Olimpíada Brasileira de Informática*. Disponível em: <http://http://olimpiada.ic.unicamp.br>.