Curso de Ciência da Computação
Estrutura de Dados

Prof. Anselmo C. de Paiva

Cap. 2 Tipos Abstratos de Dados
Funções como Tipos de Dados

Quicksort - implementação da biblioteca

- Quicksort

```
void qsort( void *base, size_t n, size_t size,
       int (*compar)( void *, void * ) );
      base        endereço de um vetor
      n           número de elementos
      size        tamanho do elemento
      compar      função de comparação
```

  - C permite passar uma função como parâmetro para uma outra função !!

Funções como tipos de dados - declaração

- Funções como tipos de dados
  - Declaração

```
int (*compar)( void *, void * );
```

Parênteses em torno do * e do nome da função
Define que é um ponrteiro para função

Diferente de:

```
int *compar( void *, void * );
```

Função retornando um inteiro

Funções como tipo de dados

  - Declaração

```
int (*compar)( void *, void * );
```

Função retorna um *int*

Possui dois argumentos void *

  - Declara um ponteiro para uma função de nome *compar* que retorna *int* e recebe dois argumentos `void *`

Funções como tipo de dados

– Declaração

```
int (*compar)( void *, void * );
```

Função retorna um *int*

Possui dois argumentos void *

– A função de comparação passada para *quicksort* retorna

-1	*arg1 < *arg2
0	*arg1 == *arg2
+1	*arg1 > *arg2

Interpretado como definindo ordem não magnitude

Funções como tipo de dados

- Uso
  - Funções de bibliotecas que precisam de uma função com objetivo especial
    - Ordenação precisa do conceito de ordem
    - função `compar` fornece isto
      - Conceito de ordem pode ser complexo
        - » ***eg*** nomes em uma lista telefônica
    - Busca também necessita de ordem

Funções como tipos de dados

- Generalizando nosso ADT collection
  - Usamos ponteiros para a estrutura(struct) que representa o elemento da coleção
    - Permite que o ADT collection armazene qualquer tipo de objetos
  - *Mas* precisamos assumir que existem duas funções externas
    - `itemkey e itemcmp`
    - Restringindo a uma coleção por programa
- Coloque a função de ordenação como atributo
  - Regra de ordenação é armazenada com os atributos da coleção
  - Quantas coleções eu quiser.

ADT Collection Generalizado

- Redefina a função de criação da coleção

```
Collection ConsCollection( int max_items,
      int (*compar)(void *, void * ) );
```

- Adicione um elemento aos atributos

```
struct t_collection {
   int max_items;
   int cnt;
   int (*compar)(void *, void * );
   void *data;
};
```

Assume que utilizamos um vetor - necessário modificar para listas, árvores, etc.

ADT Collection Generalizado

- Na função de construção trate a função parâmetro como se fosse um ponteiro qualquer:

```
Collection ConsCollection( int max_items,
      int (*compar)(void *, void * ) )
{
    ptrCollection c;
    c =(ptrCollection) malloc(sizeof(Collection));
    if ( c != NULL ) {
       c->max_items = max_items;
       c->compar = compar;
       ……
    }
    return c;
}
```

Copie o ponteiro (é um endereço) como outro tipo de dado qualquer

***Compar* é um ponteiro para uma função** - em geral o endereço do inicio do código da função

Usando um ponteiro para uma função

- Use o atributo passado como o nome de uma função:

```
void *FindInCollection( void *key )
{
    int k;
    for(k=0;k<c->cnt;k++) {
       if( *c->compar( key, c->data[k] ) == 0 )
            return c->data[k];
    }
    return NULL;
}
```

c->compar é um ponteiro para uma função - acesse o endereço nele para executar a função

… e forneça os argumentos , como se fosse o nome de uma função que vc ja esta acostumado a usar!

Funções como Tipo de Dados - Outros usos

- Códigos conduzidos por tabelas
  - ***eg*** menus
    - Cada entrada
      - String
      - Ação a ser realizada se a entrada for selecionada

```
struct menu {
    char *desc;
    int (*action)( void );
} file_menu[ ] = {
       { "Save",    save_function },
       { "Save as", save_as_function },
       …
    };
```

Ponteiro para uma função que não receb argumentos

Strings que aparecem no menu

Funções a serem chamadas

Funções como tipos de dados - Outros usos

- Usando a tabela

```
struct menu {
    char *desc;
    int (*action)( void );
} file_menu[ ] = {
                  { "Save",  save_function },
                  { "Save as", save_as_function },
                  …
                };
void file_menu_callback()
{
  int k = item_selected( file_menu );
  file_menu[k].action();
}
```

Encontra o item a ser selecionado

Chama a função associada