Algoritmos e Estruturas de dados

Bibliotecas

Prof. Dr. Fábio Rodrigues de la Rocha

UTFPR

Bibliotecas - Introdução

Definição

Bibliotecas são conjuntos de funções "empacotadas" na forma de um arquivo. Se desejarmos utilizar uma destas funções, basta informar que o seu fonte C utilizará uma determinada biblioteca. Mas vamos aos detalhes.

Bibliotecas - Introdução

O GCC (compilador utilizado internamente pelo codeblocks) é na verdade um conjunto de ferramentas que são executadas sem que o programador saida. De fato, existem vários estágios na compilação de um fonte C até a geração de um executável.

Bibliotecas - Introdução

Pré-processador (cpp) - Etapa inicial que trata as diretrizes `#define`, `#include`, `#ifndef`, etc.
veja o que ocorre ao executarmos o `cpp teste.c` onde teste.c é:

```
1 int main (void) {
2     printf("teste");
3     return(1);
4 }
```

É claro que este fonte .C precisa do `#include<stdio.h>` então execute novamente o cppcom o fonte .c que faz um `include` e veja a diferença.

```c
#include <stdio.h>
#ifdef USA_MENSAGEM
void mostra (void) {
    printf("MSG\n");
    printf("Oi");
}
#else
void mostra (void) {
    printf("NAO MSG\n");
    printf("Oi");
}
#endif
int main (void) {
    printf("teste\n");
    mostra();
    return(1);
}
```

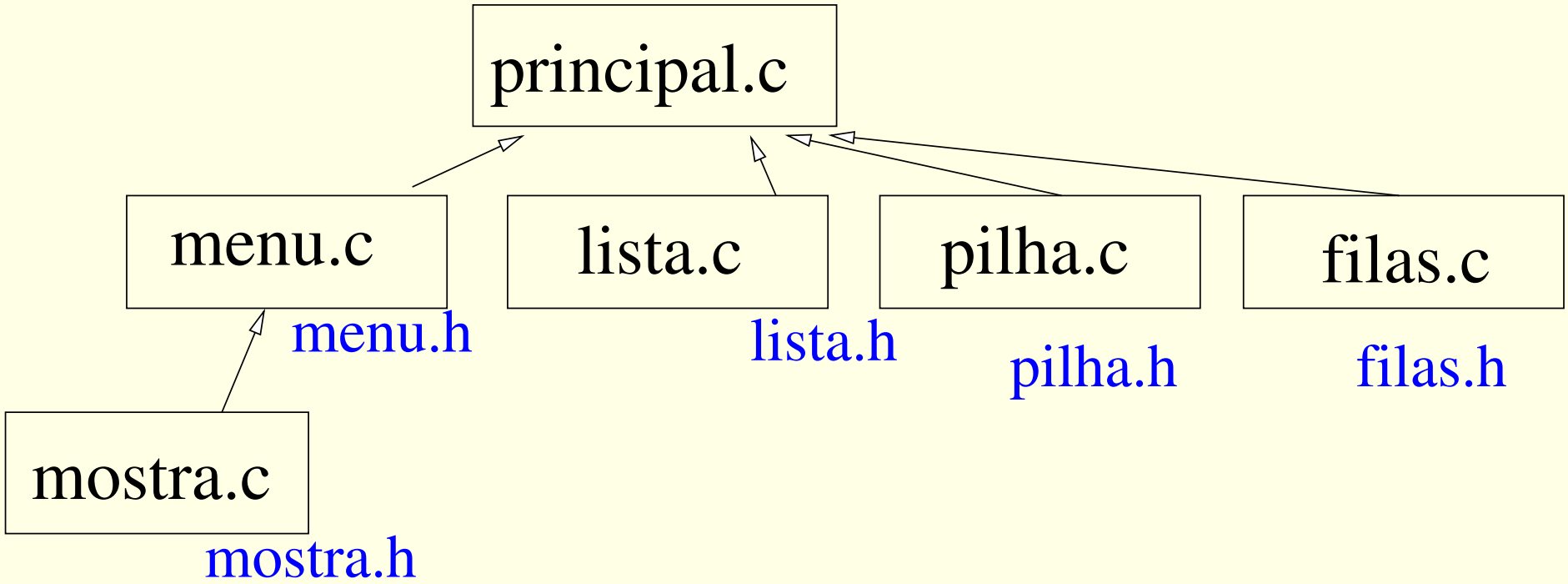
Bibliotecas - Introdução

Compilador (cc) - Transforma as instruções C em instruções assembly, tipicamente gera um arquivo .s (teste chamar o `gcc teste.c -S`

Montador - transforma o arquivo com instruções assembly em um arquivo compilado `.o`, utiliza a ferramente `gas`(gnu assembler) ou as caso o gnu assembler não exista.

Ligador Os módulos `.o`gerados pelo montador são aglutinadas para gerar o arquivo executável. Nesta etapa é que as bibliotecas são também são adionadas.

Trabalhando com múltiplos arquivos .c



```c
// PRINCIPAL.C
#include "lista.h"
#include "filas.h"
#include "pilha.h"
#include "menu.h"

int main (void)
{
      Tipo_Lista a;       Tipo_Pilha b;       Tipo_Fila  c;

      inicializa_lista(&a);
      inicializa_pilha(&b);
      inicializa_fila(&c);

      mostra_menu("Operacoes com listas");
      insere_lista(&a, 123);
      insere_lista(&a, 23);
      insere_lista(&a, 435);

      insere_pilha (&b, 234);
      mostra_lista(a);
      return 0;
}
```

Código:principal.c

```c
// MENU.C
#include "mostra.h"
#include "menu.h"

void mostra_menu (char st[] )
{
          printf("**************************************** \n");
          escreve_centralizado(st, 40);
          printf("**************************************** \n\n");
}
```

Código:menu.c

```c
// MENU.H
#ifndef _MENU_
#define _MENU_
#include <stdio.h>

void mostra_menu (char st[] );

#endif
```

Código:menu.h

```c
// MOSTRA.C
#include "mostra.h"
void escreve_centralizado (char palavra[], int tamanho_maximo) {
        int x, y;
        x=strlen(palavra);
        x = (tamanho_maximo-x)/2;

        for (y=1;y<x;y++) printf("#");    printf("%s",palavra);
        for (y=1;y<x;y++) printf("#");    printf("\n");
}
```

Código:mostra.c

```c
// MOSTRA.H
#ifndef _MOSTRA_
#define _MOSTRA_
#include <string.h>
#include <stdio.h>

void escreve_centralizado (char palavra[], int tamanho_maximo);
#endif
```

Código:mostra.h

```c
// LISTA.H
#ifndef _LISTA_
#define _LISTA_
#include <stdlib.h>
#include <stdio.h>

struct tipo_elemento{
   int valor;
   struct tipo_elemento *proximo;
   struct tipo_elemento *anterior;
};
typedef struct {
   int qtd;
   struct tipo_elemento *fim, *inicio;
} Tipo_Lista;
void inicializa_lista (Tipo_Lista *a);
void insere_lista (Tipo_Lista *a, int v);
void mostra_lista (Tipo_Lista m);
void mostra_elemento (struct tipo_elemento *tmp);
struct tipo_elemento * pesquisa_lista (Tipo_Lista a, int alvo );
void desaloca_todos_lista (Tipo_Lista *a);
void elimina (Tipo_Lista *a, struct tipo_elemento *ponteiro_alvo);
#endif
```

Código:lista.h **Código:**lista.c

```c
// PILHA.H
#ifndef _PILHA_
#define _PILHA_

#include <stdio.h>
#include <stdlib.h>

typedef struct {
      struct tipo_elemento *topo;
      int qtd;
}Tipo_Pilha;


void inicializa_pilha (Tipo_Pilha *p);
void insere_pilha (Tipo_Pilha *p, int v);
int remove_pilha (Tipo_Pilha *p);
int eh_vazia_pilha ( Tipo_Pilha p);

#endif
```

Código:pilha.h **Código:**pilha.c

```c
// FILAS.H
#ifndef _FILAS_
#define _FILAS_

#include <stdio.h>
#include <stdlib.h>

#define VERDADE 1
#define FALSO 0


typedef struct {
          struct tipo_elemento *inicio;
          struct tipo_elemento *fim;
          int qtd;
}Tipo_Fila;

void inicializa_fila (Tipo_Fila *f);
void insere_fila (Tipo_Fila *d, int x);
int retira_fila (Tipo_Fila *f);
int eh_vazia_fila (Tipo_Fila f);

#endif
```

Código:filas.h **Código:**filas.c

Compilando e gerando arquivos objeto

```
1 gcc -c menu.c
2 gcc -c principal.c
3 gcc -c mostra.c
4 gcc -c lista.c
5 gcc -c filas.c
6 gcc -c pilha.c
7 // Agora junta tudo num executavel "saida"
8 gcc -o saida.exe menu.o principal.o mostra.o
    lista.o filas.o pilha.o
```

Código:compilando

Criando Bibliotecas

```
1  gcc -c menu.c
2  gcc -c mostra.c
3  gcc -c lista.c
4  gcc -c pilha.c
5  gcc -c filas.c
6
7  // Cria uma biblioteca com o nome "libestruturas_dados.a"
8  ar rs libestruturas_dados.a lista.o pilha.o filas.o
9
10 // mostra quais arquivos objeto existem dentro de uma biblioteca
11 ar -t libestruturas_dados.a
12
13 // vamos ver o que existe dentro da biblioteca matematica do C ?
14 ar -t /usr/lib/i386-linux-gnu/libm.a
15
16 // compila o programa principal, junta com a biblioteca e gera o executavel
17 // saida
18 gcc principal.c -lestruturas_dados  menu.o mostra.o -o saida
19 // Como as bibliotecas s o procuradas num diret rio padr o a linha
20 // acima gerar  um erro. Use   gcc principal.c -L./ -lestruturas_dados  menu.o mostra.o
       -o saida para procurar no diretorio corrente
```

Código:criando_bibliotecas

Makefile

Definição

O Makefile é um arquivo texto que diz como um conjunto de arquivos fonte (aplicação) deve ser compilado. O Makefile deve ser criado no mesmo diretório onde esttão os arquivos .c e .h. e será lido por um programa chamado **make** que deve ser digitado na linha de comando. Perceba que não é passado qual arquivo deve ser lido, o programa make já sabe que será o Makefile que existir no diretório corrente.

```
1 frr@frr-desktop:~/UTFPR/Disciplinas/2011-2/AED/fontes/multiplos$ make
2 gcc -c menu.c
3 gcc -c principal.c
4 gcc -c mostra.c
5 gcc -c lista.c
6 gcc -c filas.c
7 gcc -c pilha.c
8 gcc -o saida.exe menu.o principal.o mostra.o lista.o filas.o pilha.o
```

Makefile

```
1  COMPILADOR=gcc
2  APAGA=rm -f
3  saida.exe:   menu.o principal.o mostra.o lista.o filas.o pilha.o
4            $(COMPILADOR) -o saida.exe menu.o principal.o mostra.o lista.o filas.o pilha.o
5
6  principal.o: principal.c menu.h lista.h filas.h pilha.h
7            $(COMPILADOR) -c principal.c
8
9  menu.o:   menu.c menu.h mostra.h
10           $(COMPILADOR) -c menu.c
11
12 mostra.o: mostra.c mostra.h
13           $(COMPILADOR) -c mostra.c
14
15 lista.o: lista.c lista.h definicoes.h
16           $(COMPILADOR) -c lista.c
17
18 filas.o: filas.c filas.h definicoes.h
19           $(COMPILADOR) -c filas.c
20
21 pilha.o: pilha.c pilha.h definicoes.h
22           $(COMPILADOR) -c pilha.c
23
24 clean:
25           $(APAGA) saida.exe *.o
```

Makefile

Código: Makefile