Universidade Estadual de Goiás – UEG
Câmpus de Santa Helena
Sistemas de Informação

Lógica de programação II

Diretivas

Edmar Augusto Yokome

Santa Helena, 2019

1- Definição

- Permite que o programador modifique a compilação
- O pré-processador é um programa que examina e modifica o código-fonte antes da compilação
- As diretivas são os comandos utilizados pelo pré-processador
  - Estes comandos estarão disponíveis no código-fonte, mas não no código compilado
- As diretivas iniciam com #
  - Ex.: #include <stdio.h>

2- Diretiva #include

- Permite inserir um arquivo qualquer no código-fonte
- A diretiva include é substituída pelo conteúdo do arquivo
- Quando usamos <> para indicar o arquivo, este arquivo é procurado somente na pasta include
- Quando utilizamos “” para indicar o arquivo, este arquivo é procurado na pasta atual, e se não for encontrado é procurado na pasta include

- Exemplo:

```
int soma(int a, int b)  {
    return a+b;
 }
```

arquivo.c

```
#include<stdio.h>

#include "arquivo.c"
main(){
    int resultado = 0;
    resultado = soma(1, 2);
    printf("A soma e: %i",resultado);
   }
```

principal.c

2- Diretiva #define

- Permite definir constantes sem consumir memória durante a execução
- Não use o sinal de atribuição =

```c
#include<stdio.h>

#define PI 3.14

main(){
    float raio = 1.0;
    float area = PI * raio * raio;
    printf("Area: %.2f ",area);
}
```

2.1- Permite definir trechos fixos de código

- Permite definir trechos fixos de código

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

#define MSG printf("Mensagem!!\n"); exit(1);

main(){
   MSG;
}
```

2.2- Diretiva #define

- Permite definir trechos de código com parâmetros (macros)
- Não pode ter espaços no identificador. Ex.: SOMA (x,y)

```c
#include<stdio.h>

#define SOMA(x,y) x + y

main(){

    int a = SOMA(1, 2);
    double b = SOMA(1.0, 2.0);

    printf("Soma 1: %d\n",a);
    printf("Soma 2: %.1f",b);
}
```

2.3- Diretiva #define

- Recomenda-se usar parênteses em macros

```c
#include<stdio.h>

#define SOMA(x,y) x + y

main(){

   printf("%d\n", 10 * SOMA(1,2));

}
```

```c
#include<stdio.h>

#define SOMA(x,y) (x + y)

main(){

   printf("%d\n", 10 * SOMA(1,2));

}
```

2.3.1- Macros com quebra de linha

- Podemos escrever uma função macro usando mais de uma linha. Para tanto, basta adicionar barra (\) ano final de cada linha da macro.

```c
#include<stdio.h>

#define TROCA(a,b,c) {c t=a; a=b; b=t;}

main(){

    int x, y;
    x = 1;
    y = 2;

    TROCA(x, y, int);

    printf("Valor x: %d\n",x);
    printf("Valor y: %d",y);
}
```

```c
#include<stdio.h>

#define TROCA(a,b,c) {c t=a; \
a=b;\
b=t;}

main(){

    int x, y;
    x = 1;
    y = 2;

    TROCA(x, y, int);

    printf("Valor x: %d\n",x);
    printf("Valor y: %d",y);
}
```

3- Diretiva #undef

- Remove a definição criada com #define

```c
#include<stdio.h>

#define TAM 20

main(){

    int vetor[TAM];

    #undef TAM
    #define TAM 100

    vetor[TAM];
}
```

4- Compilação separada

- Podemos separar nosso programa em vários arquivos utilizando a diretiva #include

```
int soma(int a, int b);
```

arquivo.h

```
#include "arquivo.h"

int soma(int a, int b){
   return a+b;
}
```

arquivo.c

```c
#include<stdio.h>

#include "arquivo.c"

main(){
    float x = 2.0;
    printf("Soma: %d\n", soma(1, 2));
}
```

principal.c

Referências

- BACKES, André. Linguagem C: Completa e Descomplicada. Editora: Campus. Rio de Janeiro, 2013
- CELES, Waldemar; CERQUEIRA, Renato; RANGEL, José Lucas. Introdução a Estrutura de Dados. Editora: Campus. Rio de Janeiro, 2004.