KEY POINTS

Implementação de Programação Modular

Espaço de Dados

Tipo de Dados

NAME/DATE/SUBJECT

24/04/2019

NOTES

Implementação da Programação Modular

1) Espaço de dados

São áreas de armazenamento:

- Possui um tamanho
- Possui um ou mais nomes de referência
- Alocados em um meio

exemplos

A[j] → j-ésimo elemento do vetor A

*ptAux → espaço apontado por ptAux

ptAux → espaço que contém um endereço

ptElemTabSimb* ObterElemTabSimb (char* ptsimbolo)

(*ObterElemTabSimb (char* ptsimbolo)).Id

ou

ObterElemTabSimb (char* ptsimbolo) → Id.

sub campo id do elemento retornado pela função acima

ponteiro da função

espaços de dados

2) Tipos de Dados

Determinam:

→ organização → como um binário é interpretado

→ codificação → XX XXXX, dígito do CPF, matrícula

→ tamanho em bytes → int < float

→ conjunto de valores permitidos.

↳ ENUM

SUMMARY

KEY POINTS

Espaço de Dados

NAME/DATE/SUBJECT

24/04/2019

NOTES

obs1:

Um espaço de dados precisa estar associado a um Tipo para que possa ser interpretado por um programa desenvolvido em uma linguagem tipada

↳ não permite a criação de variável sem tipo.

obs 2: Tipos de tipo

Tipos compilacionais

int, char, char*, short

Tipos Básicos

struct, union, enum, typedef

Tipos Abstratos de dados

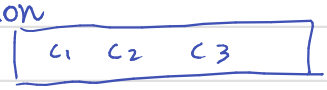
estruturas de dados encapsuladas ...

3) Tipos Básicos

- struct

mexer em c1 não afeta c2

- Union




```
1 int c1
  float c2
  char c3 }
```

type cache

SUMMARY

KEY POINTS

NAME/DATE/SUBJECT

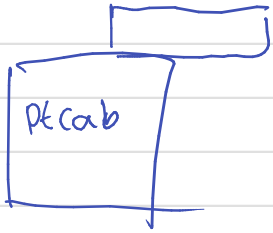
NOTES

Enum
{ a, -> 0
b, -> 1
c, -> 2
d,
e
};

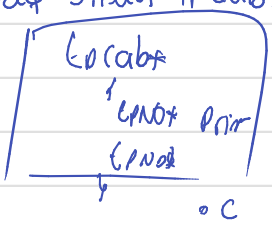
- typedef

typedef float tpVeloc,
typedef float tptempo

tptempo tempo;
tpVeloc Veloc;



typedef struct tpCab* ptCab



SUMMARY

KEY POINTS

NAME/DATE/SUBJECT

NOTES

4) Declaração e definição de elementos

definir: aloca espaço de dado e amarra o espaço ao nome (binding)

declarar: Corresponde o espaço a valores de um determinado tipo

> int x; → definir e declarar

malloc → define

tpNO → declara
struct

5) implementação e C e C++

a) declarações e definições de nomes globais exportados pelo módulo Servidor
> int a;

b) declarações externas contidas no módulo Cliente e que somente declaram o nome sem associá-lo a um espaço de dados.

> extern int a;

SUMMARY

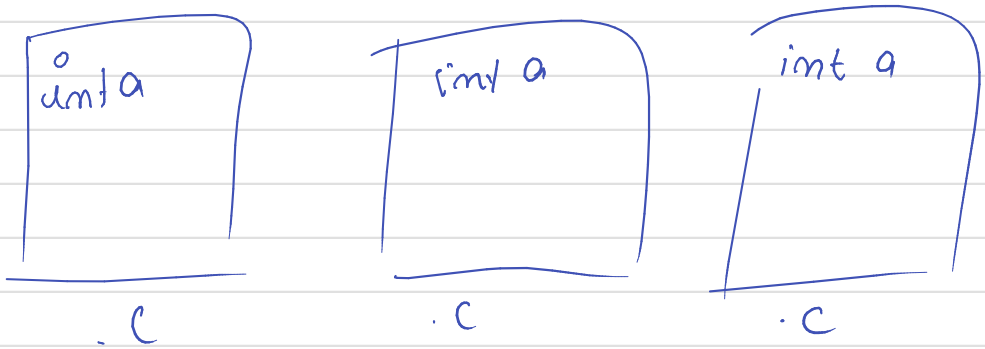
KEY POINTS

NAME/DATE/SUBJECT

NOTES

c) Declarações e definições de nomes globais encapsulados no módulo

Static int a;



a não global

a global

SUMMARY

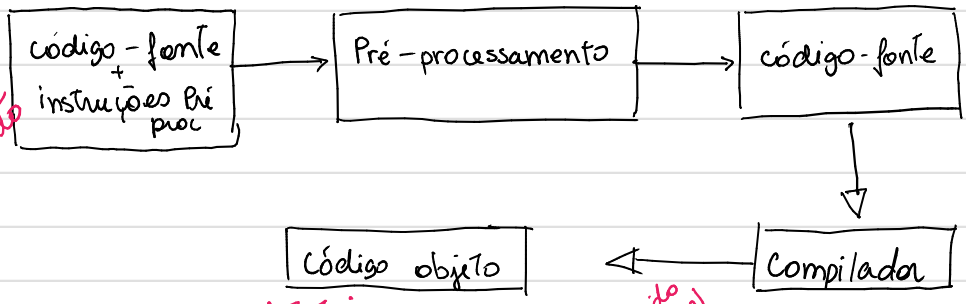
KEY POINTS

NAME/DATE/SUBJECT

29/04/2019

NOTES

6) Pré-processamento



instruções de pré-processamento

não são comandos em C

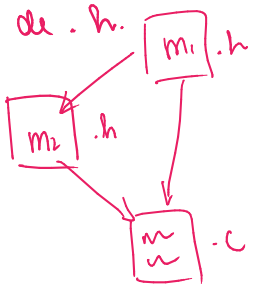
define nome valor — substitui nome por valor

include " nome arquivo " — local definido como padrão p/ include-

include < nome arquivo > — local do .C

```
# if defined (nome)        ou      # ifdef nome
      textoV
# else                             # if ! defined
      textoF                              ou
# endif                            # ifndef
```

evitar duplicidade de .h.




```
# if ! defined (Exemp_mod)
# define Exemp_mod
      corpo do .h
# endif
```

```
EXEMP_EXT int vector[7]
  # if defined (Exemplo_own)
      = {1, 2, 3, 4, 5, 6, 7};
  # else
      ;
  # endif
```

SUMMARY

```
# ifdef exemp_own          m2.h
    #define exemp_ext

# else
    #define exemp_ext extern

# endif
```

KEY POINTS

NAME/DATE/SUBJECT

NOTES

```
#define exemp_own
#include "ms.h"
#undef exemp_own

#include "ms2.h"
```

Exercícios da lista

12) Apresente uma situação de difinição sem declarar a variável.

13) É possível considerar que apenas declarar sem definir, chega a definir um espaço de dados? (certo/errado/talvez) Justifique sua resposta.

14) Como é possível personalizar interfaces para módulos cliente sem duplicar códigos? Apresente exemplo.

06/05/2019

Construtiva de Funções

1) Paradigma
   - Forma de programar → Receita de Bolo.
     - Procedural
     - Orientada a Objetos
       - POO
       - Programação Modular

SUMMARY

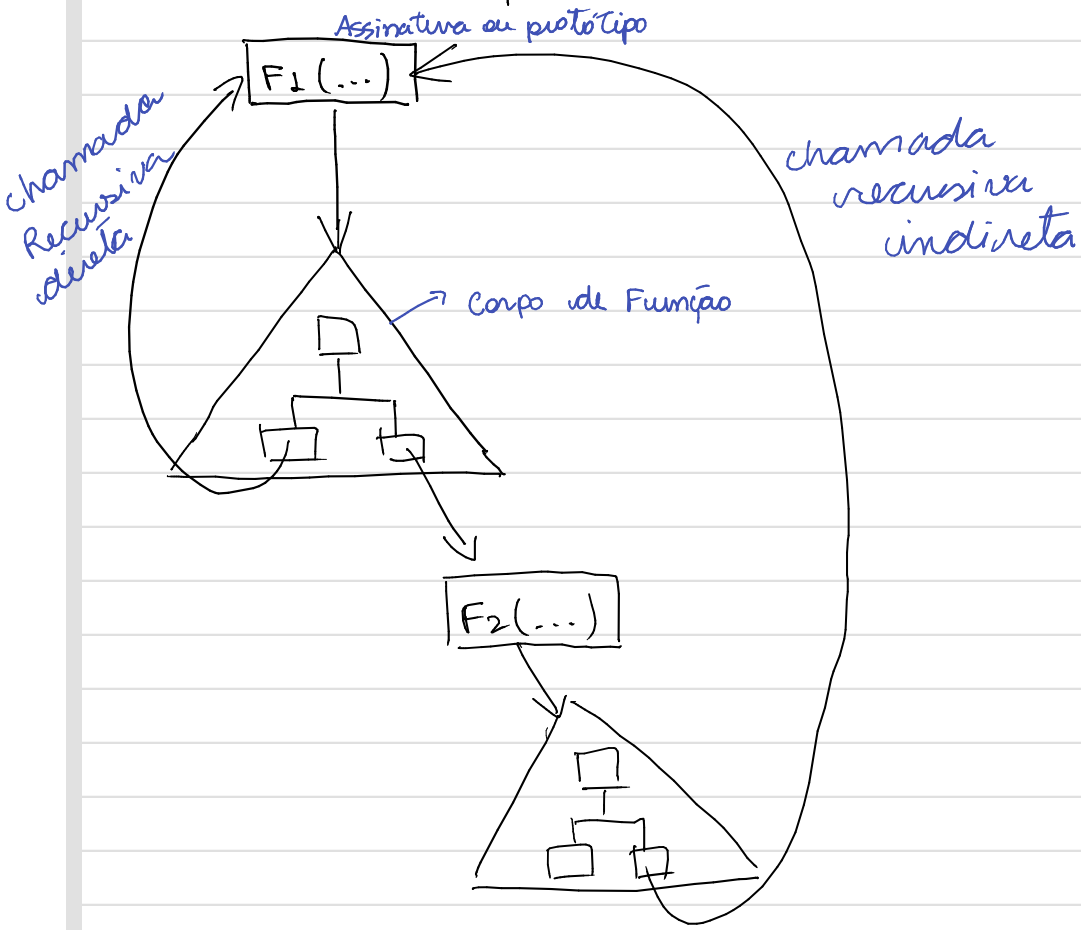
KEY POINTS

NAME/DATE/SUBJECT

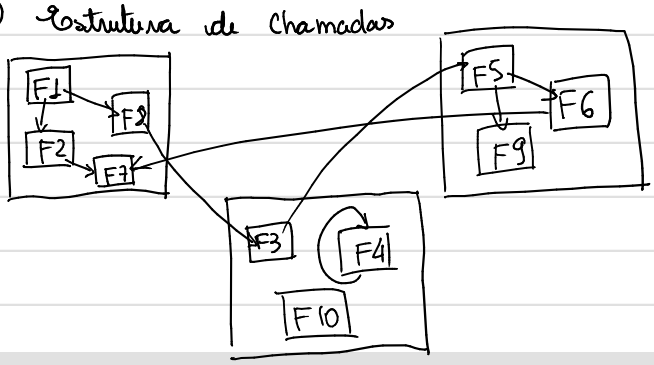
05/06/2019

NOTES

2) Estrutura de funções



3) Estrutura de Chamadas



SUMMARY

KEY POINTS

NAME/DATE/SUBJECT

06/05/2019

NOTES

Arcos de chamada

F4 → F4 chamada recursiva direta

F9 → F8 → F3 → F5 → F9 chamada recursiva indireta

F10 função morta

F8 → F3 → F5 → F6 → F7 dependência circular entre módulos

↳ não é recursiva pq não começa e termina no mesmo lugar.

F1 Origem

4) Função

é uma porção autocontida de códigos. Possui um nome, uma assinatura e um ou mais corpos de código. (ponteiro para função)

5) Especificação de Função } no .h

- Objetivo
  - Se o nome é auto explicativo esse item pode ser retirado.
- Acoplamento
  - Parâmetros e condições de retorno
- Condições de Acoplamento
  - Assertiva de entrada e saída.

SUMMARY

KEY POINTS

NAME/DATE/SUBJECT

NOTES

- Interface com o usuário
  - mensagem, mostrar informação do tabuleiro ...

- Requisitos
  - tópicos dizendo o que a função faz

- Hipóteses
  - Regra que considera válida antes do desenvolvimento da função

- Restrições
  " não pode entregar depois de terça"
  - Regras que restringem as alternativas de solução utilizadas no desenvolvimento de uma aplicação.

6) Housekeping
↳ dar free em cada malloc.
Módulo ou bloco de código responsável por liberar recursos alocados a programas, componentes ou funções ao Terminar a execução.

Asserções → módulo lista.h?
→ no doc ou nos comentários?
↳ e em funções que não tem parâmetro?
Dado ponto.

SUMMARY