Linguagens de Programação 1

Francisco Sant'Anna
Sala 6020-B
francisco@ime.uerj.br

http://github.com/fsantanna-uerj/LP1

Tipos Compostos

O que é um tipo?

- “Natureza” ou “Classificação” de um dado
- Número, Texto (string), Booleano
  - Número real, inteiro, inteiro de 1 byte, …

- `v=100 ; type(v)`
- `v=100.0 ; type(v)`
- `int v = 100;`
- `float v = 100;`

Para que tipos?

- Recusar operações inválidas
- Documentar o código
- Especializar por tamanho
- Desempenho

Tipos “Básicos”

- Numéricos: **`char`**, **`short`**, **`int`**, **`float`**, *etc*.
- Ponteiros: *`tipo`*`*`
- Vetores/Arrays: *`tipo`*`[`*`n`*`]`

- ```
  int x;
  int xs[10];
  int* p = xs;
  *(p+3) = 100;
  ```
  ```

structs

structs

- *Record,* Registro, *Product Type*
- Construtor de tipos novos
  - construtor **E**
  - *campo1 (de tipo1)* **E** *campo2 (de tipo2)* **E** ...
- Exemplo:
  - Um `Personagem` é representado por um tipo composto por

    `forca(int)` **`E`** `energia(int)` **`E`** `experiencia(int)`

structs - Declaração

```
#include <stdio.h>

struct Personagem {
    int forca;
    int energia;
    int experiencia;
};

void main (void) {
    struct Personagem p1;
    p1.forca       = 10;
    p1.energia     = 100;
    p1.experiencia = 0;

    struct Personagem p2 = { 13, 150, 200 };

    printf("> %d %d\n", p1.forca, p2.forca);
}
```

structs - Leitura

```
#include <stdio.h>

struct Personagem {
    int forca;
    int energia;
    int experiencia;
};

void main (void) {
    struct Personagem p1;

    int forca;
    scanf(“%d”, &forca);
    p1.forca = forca;

    scanf(“%d”, &p1.energia);

    printf(“%d %d\n”, p1.forca, p1.energia);
}
```

structs - Ponteiros

```
#include <stdio.h>

struct Personagem {
    int forca;
    int energia;
    int experiencia;
};

void main (void) {
    struct Personagem p1;
    scanf("%d", &p1.forca);

    struct Personagem* ptr = &p1;

    printf("%d %d %d\n",
           p1.forca, (*ptr).forca, ptr->forca);
}
```

structs - Leitura

```
#include <stdio.h>

struct Personagem {
    int forca;
    int energia;
    int experiencia;
};

// preencheA?, preencheB?

void main (void) {
    struct Personagem p1 = preencheA();

    sturct Personagem p2;
    preencheB(&p2);

    printf("%d %d\n", p1.forca, p2.forca);
}
```

structs - Leitura

```
#include <stdio.h>

struct Personagem {
    int forca;
    int energia;
    int experiencia;
};

struct Personagem preencheA (void) {
    ...
}

void main (void) {
    struct Personagem p1 = preencheA();
    printf("%d\n", p1.forca);
}
```

structs - Leitura

```
#include <stdio.h>

struct Personagem {
    int forca;
    int energia;
    int experiencia;
};

void preencheB (struct Personagem* p) {
    ...
}

void main (void) {
    struct Personagem p2;
    preencheB(&p2);
    printf("%d\n", p2.forca);
}
```

Exercícios 1 - 10

- No site:
- https://github.com/fsantanna-uerj/LP1/blob/master/Exercicios/lp1-07-structs.md

unions

unions

- União, Variante, *Sum Type*
- Construtor de tipos novos
  - construtor **OU**
  - *campo1 (de tipo1)* **OU** *campo2 (de tipo2)* **OU** ...
- Exemplo:
  - Uma `Identidade` é representada por um tipo composto por

    `IFP(int)` **`OU`** `CPF(int)` **`OU`** `NOME(char[255])`

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

union Ident {
    int  ifp;
    int  cpf;
    char nome[256];
};

int main (void) {
    union Ident i1;
    i1.ifp = 117766118;

    union Ident i2;
    i2.cpf = 1688833355;

    union Ident i3;
    strcpy(i3.nome, “Francisco Sant’Anna”);

    printf("> %d %d %s\n", i1.ifp, i2.cpf, i3.nome);

    return 0;
}
```

unions

- Como fica a memória?
- O que acontece se gravar como um subtipo e ler como outro?
- C possui tipagem fraca
  - Similar ao problema do tamanho de um vetor...
- Solução?

União Discriminada

- Solução?
  - Guardar o subtipo em uso

```
struct Ident {
  int sub; // [0,1,2]
  union {
    int  ifp;          // sub=0
    int  cpf;          // sub=1
    char nome[255]; // sub=2
  };
};
```

```
void exibe (struct Ident* id) {
   switch (id->sub) {
      case 0:
        printf("IFP = %d\n", id->ifp);
        break;
      case 1:
        ...
      case 2:
        ...
      default:
        // erro
   }
}
```

structs + unions

- Maneiras de declarar e usar
  - Separado
  - Aninhado
  - Aninhado / Separado

05-struct-union.c

Exercícios 1 - 8

- No site:
- https://github.com/fsantanna-uerj/LP1/blob/master/Exercicios/lp1-07-unions.md

enums

enums

- Enumeração
- Apenas um apelido (uma constante) para inteiros
  - **É apenas um apelido para inteiros**
- Exemplos:
  - Usar `Qui` em vez de `5`
  - Usar `Professor` em vez de `1`

```
enum Dia {
  Dom = 1,
  Seg = 2,
  Ter = 3,
  Qua = 4,
  Qui = 5,
  Sex = 6,
  Sab = 7,
};
```

```
enum Classe {
  Aluno,       // 0
  Professor,   // 1
  Funcionario // 2
};
```

```
enum Dia  dia = Qui;
enum Tipo sub = Professor;

printf("%d %d\n", dia, sub);
printf("%s %s\n", dia, sub);
```

enums

- Muito usado conjunto com uniões discriminadas
  - Guardar o subtipo em uso

```
enum Sub { IFP, CPF, NOME };

struct Ident {
  enum Sub sub;
  union {
    int  ifp;
    int  cpf;
    char nome[255];
  };
};
```

```
void exibe (struct Ident* id) {
   switch (id->sub) {
      case IFP:
        printf("IFP = %d\n", id->ifp);
        break;
      case CPF:
        ...
      case NOME:
        ...
      default:
        // erro
   }
}
```

Exercícios 1 - 3

- No site:
- https://github.com/fsantanna-uerj/LP1/blob/master/Exercicios/lp1-07-enums.md