http://www.nickgentry.com/

Algoritmos e Programação de Computadores

Disciplina 113476

Prof. Alexandre Zaghetto
http://alexandre.zaghetto.com
zaghetto@unb.com

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Módulo 10
Estrutura de Dados Heterogêneas
(*Structs* ou Registros)

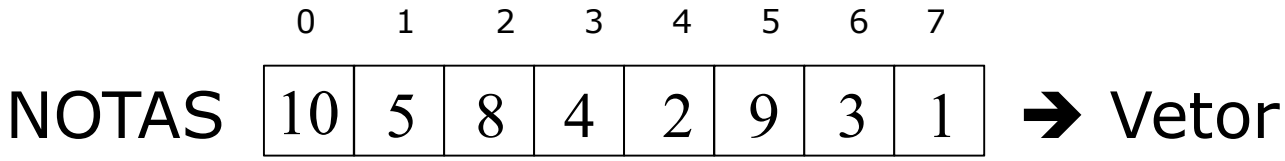
1. Structs

- Uma ***struct*** é uma coleção de campos, em que cada campo pode ser de um tipo de dado diferente. Por isso, são conhecidas como estruturas de dados heterogêneas.

1. Structs

• Uma ***struct*** é uma coleção de campos, em que cada campo pode ser de um tipo de dado diferente. Por isso, são conhecidas como estruturas de dados heterogêneas.

• Exemplo:



1. Structs

• Uma ***struct*** é uma coleção de campos, em que cada campo pode ser de um tipo de dado diferente. Por isso, são conhecidas como estruturas de dados heterogêneas.

• Exemplo:

1. Structs

- Uma ***struct*** é uma coleção de campos, em que cada campo pode ser de um tipo de dado diferente. Por isso, são conhecidas como estruturas de dados heterogêneas.

- Exemplo:

Dados_de_Funcionario					
Codigo:	9182	**Nome:**	Joseph Climber	**Sexo:**	Masculino
Endereco:	Universidade de Brasília				
Cargo:	Professor	**Salario:**	R$ 455,46		

1. Structs

Dados_de_Funcionario					
Codigo:	9182	**Nome:**	Joseph Climber	**Sexo:**	Masculino
Endereco:	Universidade de Brasília				
Cargo:	Professor	**Salario:**	R$ 455,46		

- As variaveis Nome, Sexo, Endereço e Cargo são ***strings***, Codigo é um ***int*** e Salario é um ***float***.

- Essas variáveis e seus tipos, criam um novo tipo de dado: ***Dados_de_Funcionario***.

- Da mesma forma que existem os tipos vetor de char, int e float, agora existe o tipo Dados_de_Funcionario.

- As variáveis que estão dentro da struct são chamadas de ***membros da estrutura*** (struct).

2. Declaração de uma Struct em C

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

struct  Dados_De_Funcionario{
        int codigo;
        float salario;
        char nome[50], sexo[10];
        char endereco[50], cargo[50];
};

int main() {

   struct Dados_De_Funcionario funcionario;

   return 0
}
```

3. Acessando Membros de uma Struct

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

struct  Dados_De_Funcionario{
        int codigo;
        float salario;
        char nome[50], sexo[10];
        char endereco[50], cargo[50];
};
```

3. Acessando Membros de uma Struct

```
int main(){

struct Dados_De_Funcionario funcionario;

funcionario.codigo = 2345;
funcionario.salario = 123.4;
strcpy(funcionario.nome,"Joseph Climber");
strcpy(funcionario.sexo,"Masculino");
strcpy(funcionario.endereco,"UnB");
strcpy(funcionario.cargo,"Professor");

printf("%d \n", funcionario.codigo);
printf("%f \n", funcionario.salario);

return 0;

}
```

3. Acessando Membros de uma Struct

```
int main(){

struct Dados_De_Funcionario funcionario;

printf("Digite o codigo do funcionario:");
scanf("%d", &funcionario.codigo);
printf("Digite o salario do funcionario:");
scanf("%f", &funcionario.salario);
getchar();
printf("Digite o nome:");
gets(funcionario.nome);
printf("Digite o sexo:");
gets(funcionario.sexo);
printf("Digite o endereco:");
gets(funcionario.endereco);
printf("Digite o cargo:");
gets(funcionario.cargo);
```

3. Acessando Membros de uma Struct

```
printf("\n\n=============== \n");
printf("Dados digitados \n");
printf("=============== \n\n");
printf("Codigo: %d \n", funcionario.codigo);
printf("Salario: %f \n", funcionario.salario);
printf("Nome:");
puts(funcionario.nome);
printf("Sexo:");
puts(funcionario.sexo);
printf("Endereco:");
puts(funcionario.endereco);
printf("Cargo:");
puts(funcionario.cargo);

return 0;
}
```

4. Declarando Structs Utilizando typedef:

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

typedef struct Dados_De_Funcionario {

    int codigo;
    float salario;
    char nome[50], sexo[10];
    char endereco[50], cargo[50];

}dfunc;
```

4. Declarando Structs Utilizando typedef:

```
int main(){

    dfunc funcionario;

    funcionario.codigo = 2345;
    funcionario.salario = 123.4;
    strcpy(funcionario.nome,"Joseph Climber");
    strcpy(funcionario.sexo,"Masculino");
    strcpy(funcionario.endereco,"UnB");
    strcpy(funcionario.sexo,"Professor");

    printf("%d \n", funcionario.codigo);
    printf("%f \n", funcionario.salario);

    return 0;

}
```

5. Declarando uma Única Struct

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

int main(){

    struct {

        int codigo;
        float salario;
        char nome[50], sexo[10];
        char endereco[50], cargo[50];

    }funcionario;
```

5. Declarando uma Única Struct

```
        funcionario.codigo = 2345;
        funcionario.salario = 123.4;
strcpy(funcionario.nome,"Joseph Climber");
strcpy(funcionario.sexo,"Masculino");
strcpy(funcionario.endereco,"UnB");
strcpy(funcionario.sexo,"Professor");

printf("%d \n", funcionario.codigo);
printf("%f \n", funcionario.salario);


return 0;
}
```

6. Vetor de Structs

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

struct teste{
    int a;
    float b;
};
```

6. Vetor de Structs

```
int main(){

    struct teste x[2];

    x[0].a = 10;
    x[0].b = 11.5;
    x[1].a = 12;
    x[1].b = 13.5;

    printf("%d \n", x[0].a);
    printf("%.2f \n", x[0].b);
    printf("%d \n", x[1].a);
    printf("%.2f \n", x[1].b);

    return 0;
}
```

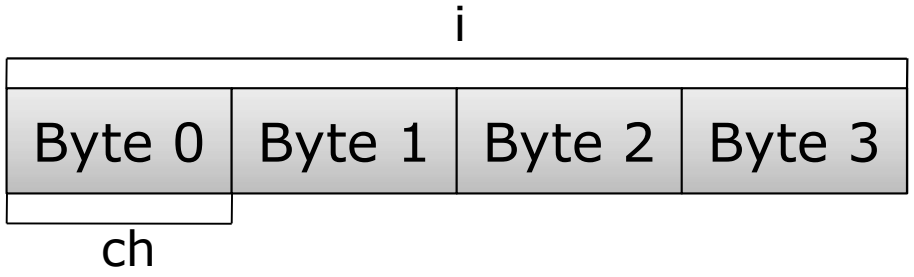
Uniões
(*Unions*)

7. Uniões

• As estruturas (*structs*) apresentam membros fixos e um único formato.

• As **uniões** (*unions*) permitem que uma variável seja interpretada de várias maneiras.

• Podem ser vistas como um região de memória que é compartilhada por duas ou mais variáveis.



7. Uniões

- Exemplo:

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

union teste {
    int i;
    char ch;
};

int main(){

    union teste x;

    system("PAUSE");
    return 0;
}
```

7. Uniões

- Exemplo:

```
int main(){

    union teste x;

    x.i = 10;
    printf("%d \n",x.i);

    x.ch = 'a';
    printf("%c \n",x.ch);

    printf("%d \n",x.i);

    system("PAUSE");
    return 0;
}
```

7. Uniões

- Um exemplo mais complexo:

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

# define LIFE 1
# define AUTO 2
# define HOME 3

struct addr {
        char street[50];
        char city[10];
        char state[2];
        char zip[5];
}; // Fim da estrutura "endereço"
```

7. Uniões

- Um exemplo mais complexo:

```
struct date {
        int month;
        int day;
        int year;
}; // Fim da estrutura "data"

struct policy {

        struct addr address; // Endereço
        int polnumber; // Número da apólice
        char name[30]; // Assergurado
        int amount; // Valor do seguro
        float premium; // Prêmio mensal
        int kind; // Tipo de apólice
```

7. Uniões

- Um exemplo mais complexo:

```
union {

        struct {
          char beneficiary[30];  //Benefic.
          struct date birthday;  //Aniv. do Asseg.
        } life; // Fim da estrtura "vida"

        struct {
          int autodeduct; // Valor dedutível
          char license[10]; // Número da licensa
          char state[2]; // Estado
          char model[15]; // Modelo
          int year; // Ano
        } automo; // Fim da estrutura "carro"
```

7. Uniões

- Um exemplo mais complexo:

```
            struct {
               int homededuct; // Valor dedutível
               int yearbuilt;  // Ano de construção
             }home; // Fim da estrutura "casa"

      } policyinfo;  // Fim da união "policyinfo"

}; // Fim da estrutura "policy"
```

7. Uniões

- Um exemplo mais complexo:

```
int main(){

    struct policy p;
    p.kind = LIFE;

    if (p.kind == LIFE){

        p.policyinfo.life.birthday.day = 14;
        printf("birthday.day: %d \n", p.policyinfo.life.birthday.day);

    }else if (p.kind == AUTO){
        p.policyinfo.automo.autodeduct = 1500;
        printf("automo.autodeduct: %d \n", p.policyinfo.automo.autodeduct);

    }else if (p.kind == HOME){
        p.policyinfo.home.yearbuilt = 1920;
        printf("home.yearbuilt: %d \n", p.policyinfo.home.yearbuilt);

    }else{
        printf("Tipo invalido!");
    }

    system("PAUSE");
    return 0;
}
```

Enumerações
(*Enumerations*)

8. Enumerações

- É um conjunto de constantes inteiras que especifica todos os valores permitidos para um determinado tipo.

- Exemplos:

```
enum coin { penny, nickel, dime, quarter,
            half_dollar, dollar};
```

```
enum semana{ domingo, segunda, terca, quarta,
             quinta, sexta, sabado};
```

8. Enumerações

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

enum coin { penny, nickel, dime, quarter,
            half_dollar, dollar};

int main(){

    enum coin money;

    money = penny;
    printf("%d \n", money);
    money = half_dollar;
    printf("%d \n", money);

    system("PAUSE");
    return 0;
}
```

8. Enumerações

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

typedef enum semana{domingo, segunda, terca, quarta,
quinta, sexta, sabado} dia;

int main(){

    dia hoje;

    hoje=quarta;

    printf("Dia: %d ",hoje);
    return 0;

}
```

"Todas as máquinas têm seu atrito, e isso possivelmente tem seu lado bom que compensa o lado ruim. De qualquer modo, seria bastante nocivo fazer muito alvoroço por causa disso. Mas quando o atrito chega ao ponto de controlar a máquina, e a opressão e o roubo se tornam organizados, digo que não devemos mais ficar presos a tal máquina."

H. D. Thoreau,
em A Desobediência Civil