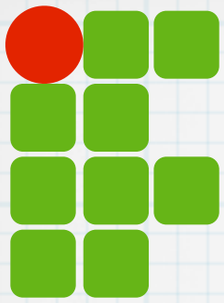
INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE

Algoritmos

ANSI C - Struct

Copyright © 2014 IFRN



Agenda

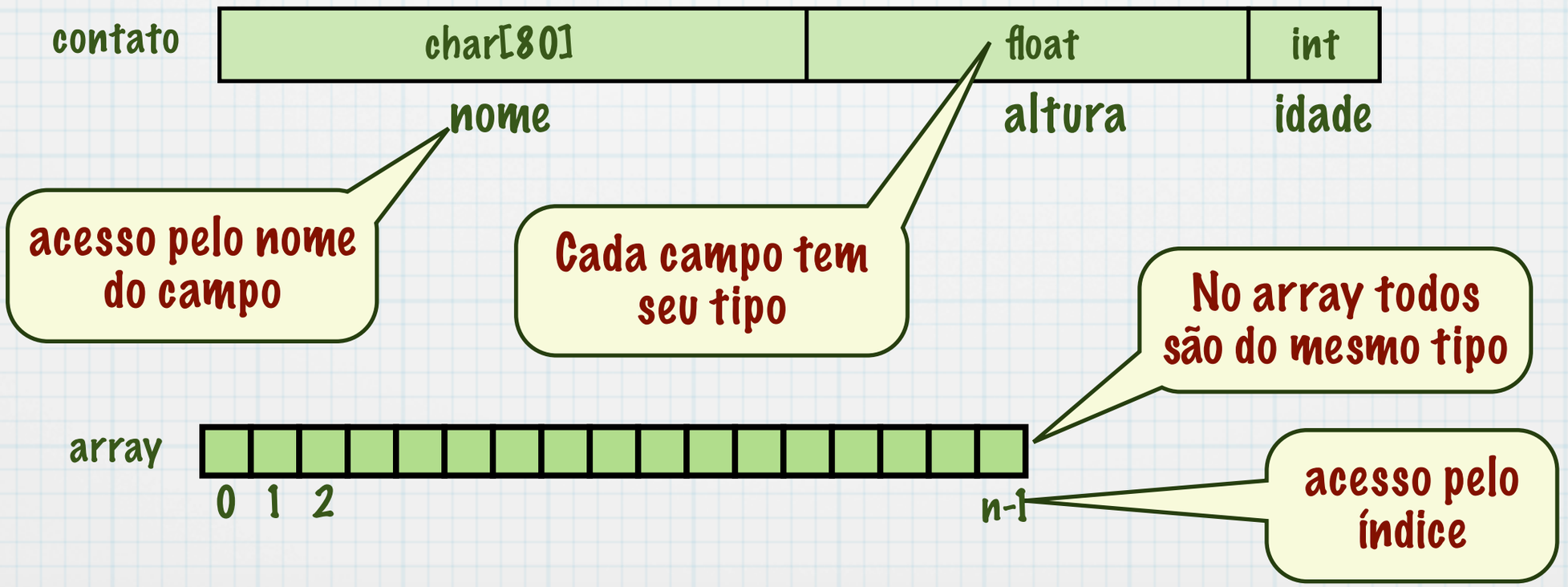
* Introdução
* Struct
* Declaração de variáveis
* Uso
* Arrays e struct
* Ponteiros para struct
* Dicas
* Exercícios

Introdução

* Array, em C, armazenam dados do mesmo tipo
* Comum agrupar dados relacionados de diferentes tipos
  * Agenda, conta bancária, aluno
    * Strings de tamanhos diferentes, inteiros, reais
* Organiza o programa
* Permite relacionar dados correlatos

Struct

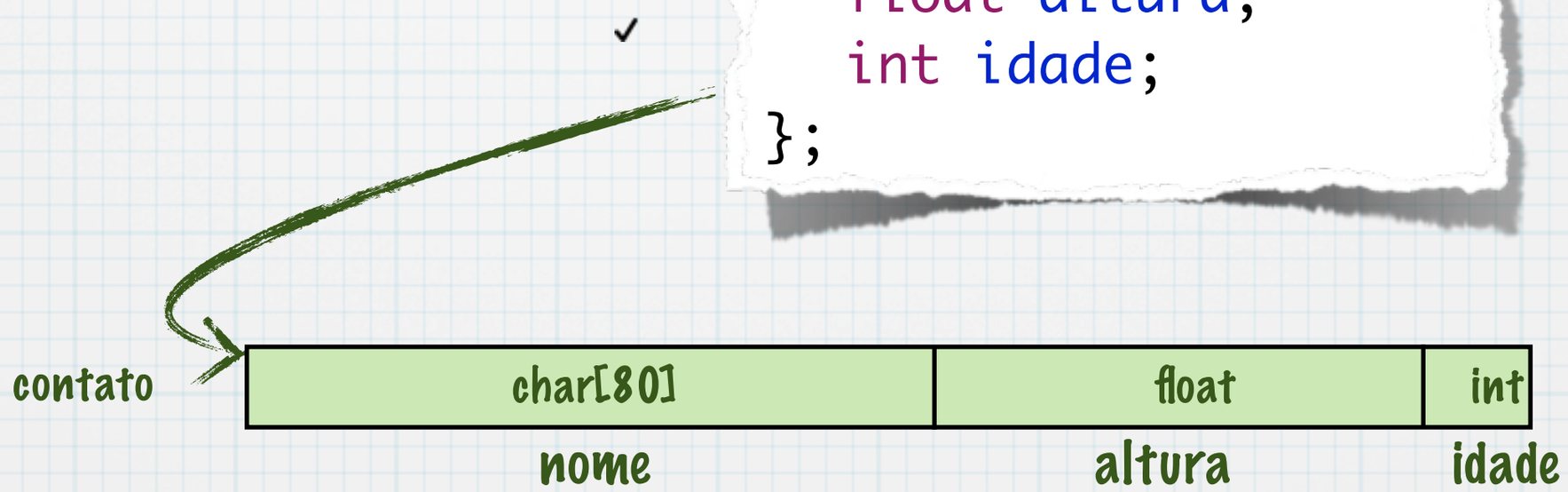
* Estrutura de dados heterogêneas
  * Armazena dados de diferentes tipos
  * Possui membros/campos
    * Dados é acessado pelo nome do campo



Em C

```c
struct nome{
   tipo1 x,y;
   tipo2 z,w;
   tipo3 a,b;
};
```

```c
struct contato{
   char nome[80];
   float altura;
   int idade;
};
```



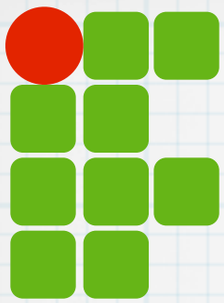
Exemplo

```c
struct data{
   int dia, mes, ano;
};
```

```c
struct hora{
   int hora,minuto,segundo;
};
```

```c
struct alunoTurma{
   char nome[80];
   int matricula;
   float nota1,nota2,nota3;
};
```

```c
struct contaCorrente{
   char nomeTitular[80];
   char cpf[11];
   double saldo;
   double limite;
};
```



Variáveis struct

* Definir struct cria um novo tipo

```
struct data{
   int dia, mes, ano;
};
```

```
struct data hoje;
```

Tipo

variáv

A variável hoje é do tipo struct data

Acesso aos campos

* O ponto (.) é usado para indicar o campo a ser acessado

```
struct data hoje,amanha;
...
hoje.ano = 2011;
amanha.ano = hoje.ano;
amanha.dia = hoje.dia+1;
```

Atribuição

* Compatibilidade de variáveis
  * mesmo tipo
  * Membros pode ser usados, respeitando os tipos

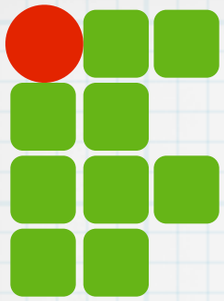
```
struct data hoje,amanha;
...
amanha = hoje;
amanha.dia = amanha.dia+1;
```

Struct e arrays

* Membros de struct podem ser arrays

```
struct alunoTurma{
   char nome[80];
   int matricula;
   float notas[3];
};
```

```
struct alunoTurma a1;
a1.matricula = 20112123321;
a1.notas[0] = 7.4;
 media = (a1.notas[0]+a1.notas[1])/2;
```

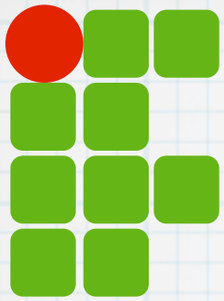


Struct e arrays

* Arrays de structs

```
struct contaCorrente contas[100];
```

```
struct contaCorrente contas[100];
contas[0].saldo  = 1000.00;
contas[0].limite = 500.00;
contas[1].saldo  = 500.00;
...
contas[x].saldo = contas[x].saldo - retirada;
```



Structs como membros

```
struct dataCompleta{
   struct data d;
   struct hora h;
};
```

```
struct dataCompleta tm;

tm.d.ano = 2011;
```

Ponteiro para struct

```c
struct Contato{
    char nome[80];
    char cpf[12];
    char telefone[20];
    char email[40];
    int idade;
};
```

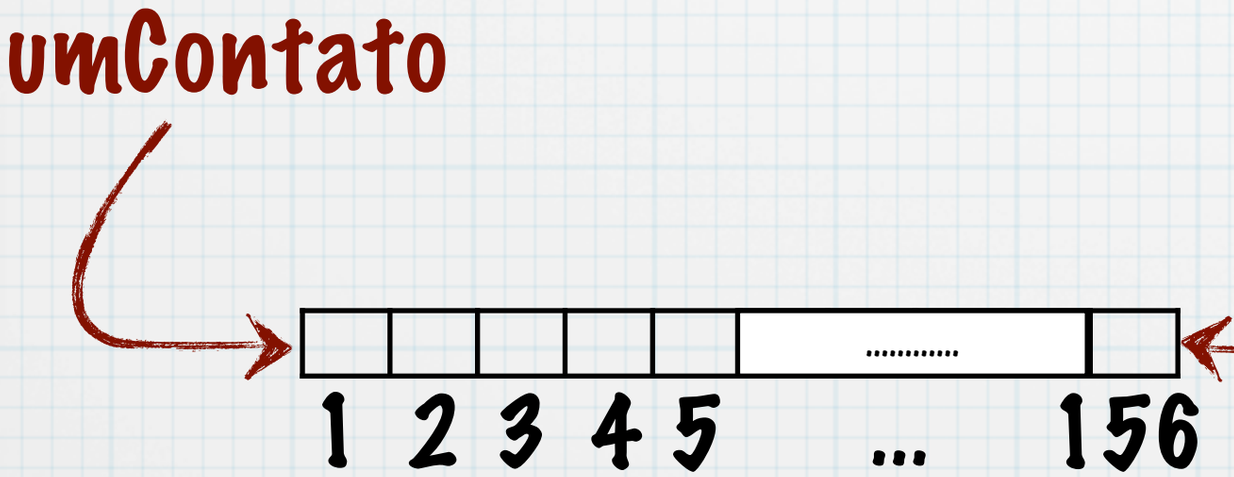
```c
struct Contato * umContato;
```

Ponteiro para struct

Ponteiro para struct

```
umContato = (struct Contato*) malloc(sizeof(struct Contato));
```

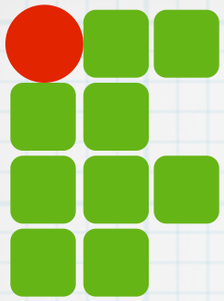
Tamanho da memória
a ser alocada
(156 bytes)



Ponteiro para struct

* Acesso
  * Quanto temos um ponteiro para struct (endereço de memória), usamos uma seta (->) ao invés do ponto (.)

```
printf("%s", umContato->nome);
umContato->idade = umContato->idade + 1;
```



Dicas

* Declare as estruturas depois dos "includes", antes da função main
  * Agrupe dados correspondentes ao mesmo elemento
    * Dados de conta corrente
    * Dados de aluno
    * Dados de Turma
    * etc

```c
//includes
#include <stdio.h>

//structs
struct contaCorrente{
    char nomeTitular[80];
    char cpf[11];
    double saldo;
    double limite;
};

//main
int main(int argc, char **argv) {
    ...
    struct contaCorrente contas[100];
    ...
    return 0;
}
```

Dúvidas?