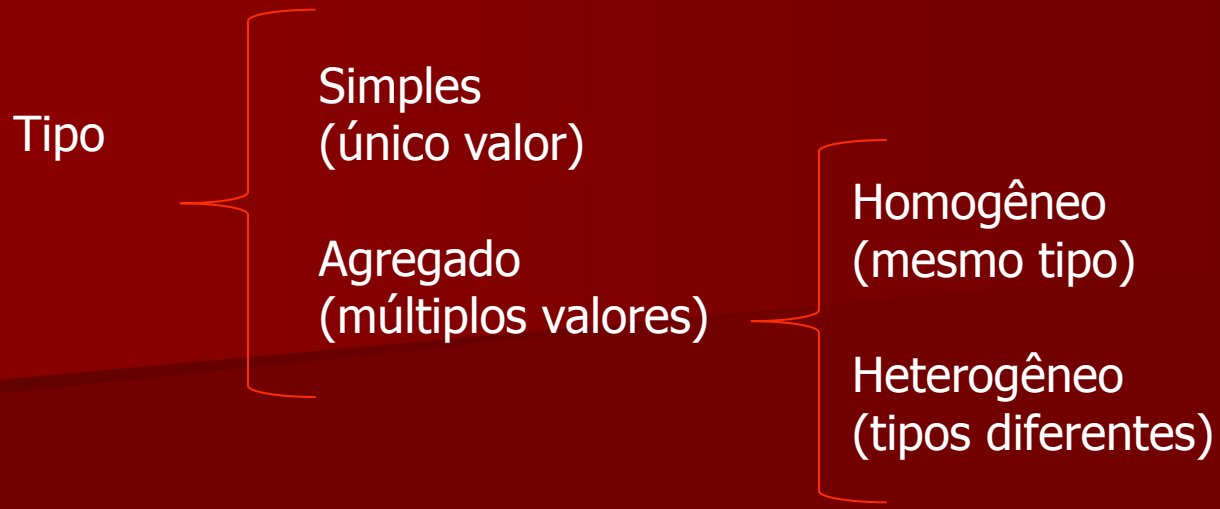
Registros

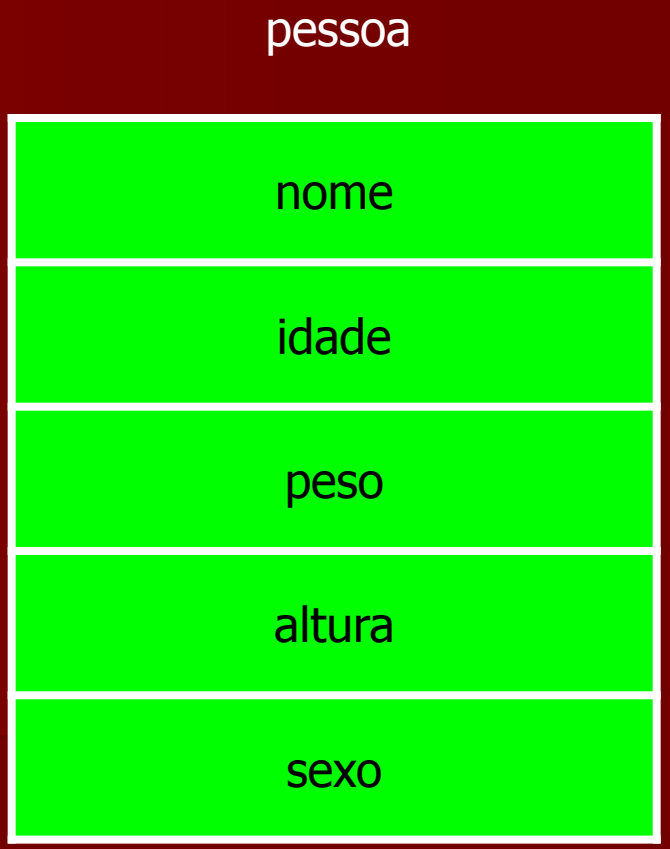
Conceitos

- Agregado heterogêno;
- Designação única para um conjunto de valores de tipos variados;
- Na linguagem C, são conhecidos como “structs” (estruturas);



Declaração

```
struct {
   char nome [30];
   int idade;
   float peso;
   float altura;
   char sexo;
   } pessoa;
```



Declaração

- Permite agrupar diversas variáveis (inclusive de tipos diferentes) sob um mesmo nome comum;
- Contribui para a lebigilidade dos programas, pois explora e evidência a coesão entre informações relacionadas;
- Confere grande flexibilidade no projeto de implementação de estruturas de dados sofisticadas.

Referência

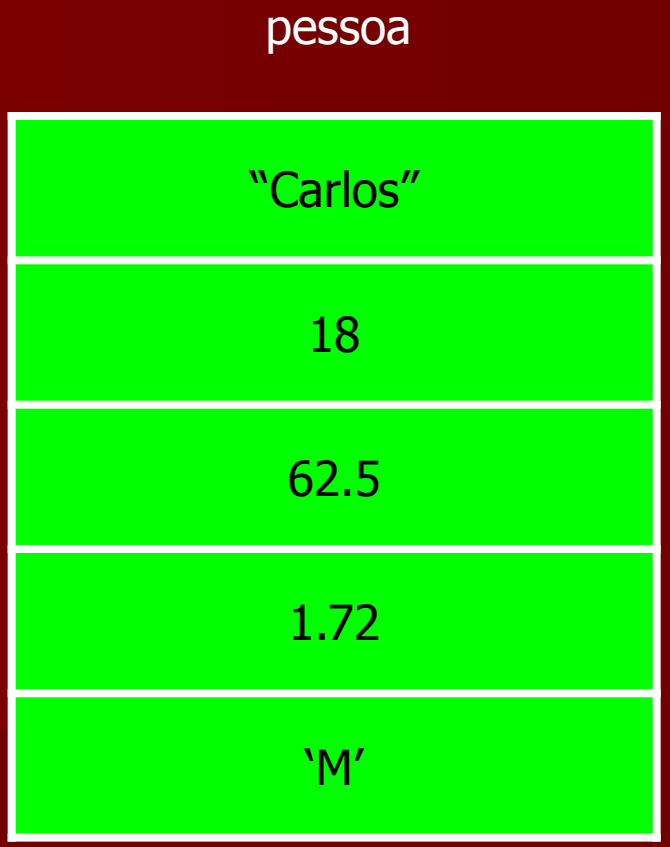
- `<variável struct> . <campo>`

- O operador “.” (seleção de campo) é usado para especificar um campo dentro de uma struct;
- Do lado esquerdo deve ser usado o nome da variável struct, do lado direito o nome do campo desejado.

- `pessoa.nome`

Referência

```
strcpy (pessoa.nome,
”Carlos”);
pessoa.idade=18;
pessoa.peso=62.5;
pessoa.altura=1.72;
pessoa.sexo=‘M’;
```



Declaração

- Quando um nome é usado depois da palavra “struct”, e antes do símbolo “{“, essa palavra denota o tipo de dados representado pela estrutura e pode ser usado na declaração de variáveis desse tipo;
- A palavra-chave “struct” deve ser usada antes do nome do tipo.

Declaração

```
struct perfil {
  char nome [30];
  int idade;
  float peso;
  float altura;
  char sexo;
  };

struct perfil pessoa;
struct perfil grupo [100];
struct perfil *p;
```

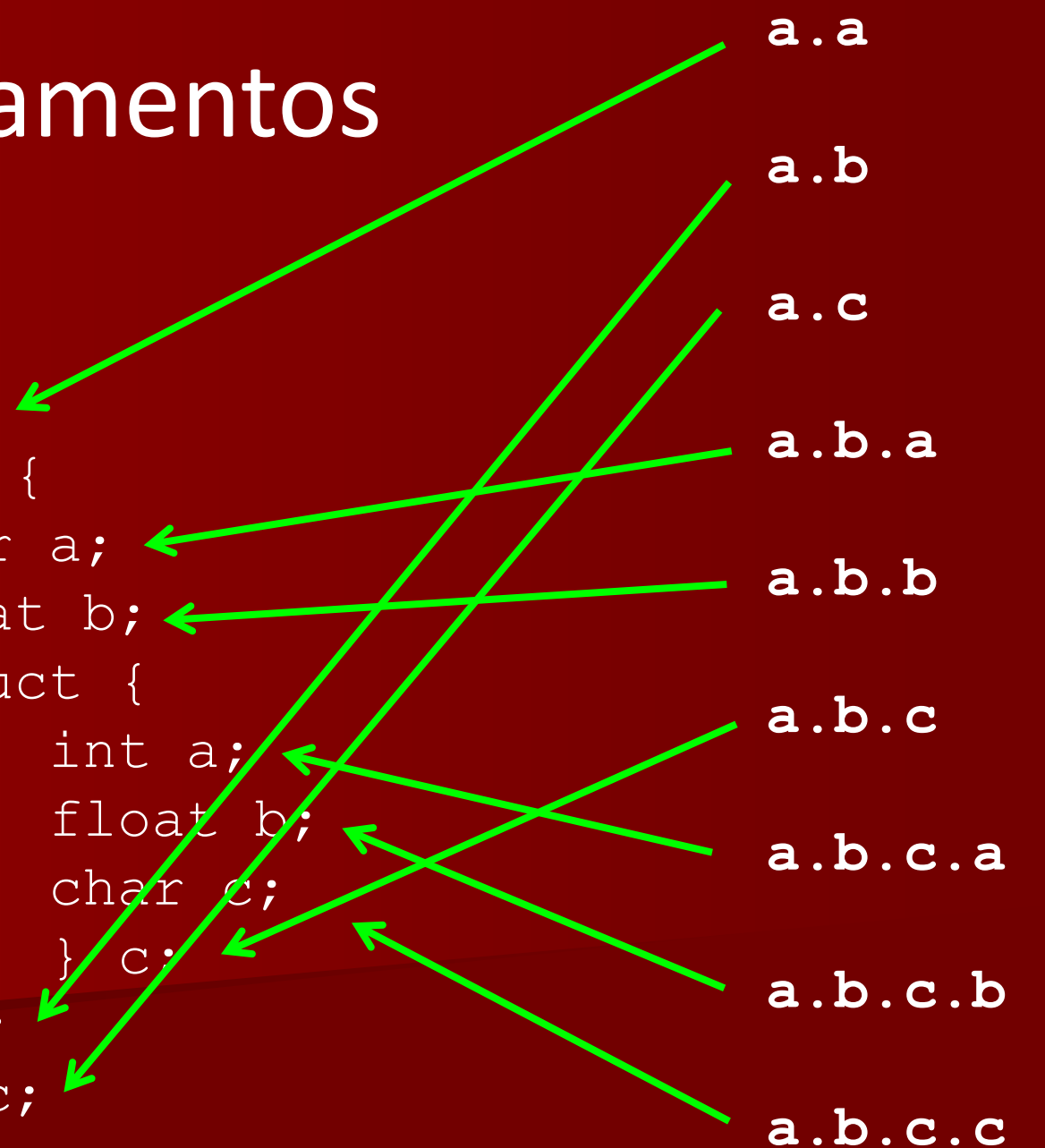
Declaração

- É possível, também, introduzir o nome do tipo na mesma declaração da variável, como em:

```
struct perfil {
   char nome [30];
   int idade;
   float peso;
   float altura;
   char sexo;
   } pessoa;
```

Aninhamentos

```
struct {
  int a;
  struct {
     char a;
     float b;
     struct {
          int a;
          float b;
          char c;
          } c;
     } b;
  float c;
  } a;
```



Vetores e structs

```
struct {int m, float n;} x [10];
```

(vetor de struct)

```
struct {
   int m;
   int x [10];
   float n;
   } a;
```

(struct com campo vetor)

```
x[2].m=0;
a.x[2]=0;
```

Tipos definidos pelo usuário

Conceitos

- Através do comando "`typedef`" é possível declarar nomes que representam tipos de dados definidos pelo usuário, e depois usar esses nomes para declarar variáveis, tipos de parâmetros e tipo de valor retornado por função, entre outros;
- Permite construir e trabalhar com abstrações que sejam mais aderentes ao domínio da aplicação que é objeto do programa;
- Do ponto de vista sintático, seu uso é idêntico às declarações de variáveis, porém com o nome typedef na frente; do ponto de vista semântico, os nomes referem-se aos respectivos tipos e podem ser usados para declarar outros nomes.

Exemplos

```
typedef int a,b,c;
typedef float d, e[10], f[10][5];
typedef char g, *h, i[40];

a p;        /* int */
b q;        /* int */
c r;        /* int */
d s;        /* float */
e t;        /* float [10] */
f u;        /* float [10][5] */
g v;        /* char */
h x;        /* char * */
i y;        /* char [40] */
```

Exemplos

```
typedef int t_idade;
typedef float t_peso, t_altura;
typedef char t_nome [30], sexo;

struct {
   t_nome nome;
   t_idade idade;
   t_peso peso;
   t_altura altura;
   t_sexo sexo;
   } pessoa;
```

Exemplos

```
typedef struct {
  t_nome nome;
  t_idade idade;
  t_peso peso;
  t_altura altura;
  t_sexo sexo;
  } t_pessoa;

t_pessoa pessoa;
```

(observe que nesse caso não se usa a palavra "`struct`" antes de `t_pessoa`)

Exemplos

```
int main () {
struct s {
   int a;
   };
struct s x;
x.a=0;
typedef struct {
   int a;
   } t;
t y;
y.a=0;
}
```