ESTRUTURAS DE DADOS

Aula 02

Dados heterogêneos e Ponteiros

Dados heterogêneos

- Armazenamento de dados de diferentes tipos, dentro de um mesmo contexto.
- Exemplo:
  - Dados de um aluno: Nome, RA, matriculado, curso, CR, etc.

Estrutura

```
struct nome{
    Tipos e nomes dos
    campos da
    estrutura;
}instância;
```

Estrutura

```
struct nome{
    Tipos e nomes dos
    campos da
    estrutura;
}instância(opcional);
```

```
struct aluno{
    char nome [35];
    long int RA;
    bool matriculado;
    char curso [3];
    float CR;
};
```

Estrutura

Instancia FILIPE

```
struct aluno{
   char nome [35];
   long int RA;
   bool matriculado;
   char curso [3];
   float CR;
}filipe;
```

Ou:

```
struct aluno filipe;
```

Estrutura

Instancia aluno, struct1

```
struct aluno{
    char nome [35];
    long int RA;
    bool matriculado;
    char curso [3];
    float CR;
};
```

```
struct {
    char nome [35];
    long int RA;
    bool matriculado;
    char curso [3];
    float CR;
}aluno;
```

Definição de tipo

- Transformação de tipo ou definição de novo tipo

typedef *tipo novoTipo*

Exemplos:

typedef int RA;

struct *aux{...};*

typedef struct aux aluno;

Typedef e Estrutura

Typedef A to aluno...

```
typedef struct A{
   char nome [35];
   long int RA;
   bool matriculado;
   char curso [3];
   float CR;
}aluno;
```

Ou:

```
typedef struct A aluno;
```

Com isso é possível:

```
aluno filipe;
```

Ao invés de:

```
struct A filipe;
```

Ou

```
struct aluno filipe;
```

Typedef e Estrutura (mais indicado)

...ou typedef "struct1" to aluno

```
typedef struct {
   char nome [35];
   long int RA;
   bool matriculado;
   char curso [3];
   float CR;
}aluno;
```

Inicialização e Atribuição

struct *estrutura instância;*

```
struct aluno filipe;
```

instância.campo = valor;

```
filipe.RA=123456;
```

Exercício

- Modelar uma estrutura para armazenar dados de alunos;
- Obter notas do usuário e calcular o CR (Coeficiente de Rendimento) de um aluno (média entre as notas);
- Imprimir na tela o nome do aluno e o CR obtido.

Possível solução (com typedef)

```
#include<stdio.h>

typedef struct {
    char nome [35];
    float n1, n2, CR;
}aluno;

void main (){
    aluno novoaluno;
    printf("Digite o nome e as duas notas do aluno: ");
    scanf(“%s %f %f",novoaluno.nome,&novoaluno.n1,&novoaluno.n2);
    novoaluno.CR=(novoaluno.n1 + novoaluno.n2) / 2;
    printf(“O CR do %s eh: %f",novoaluno.nome,novoaluno.CR);
    return;
}
```

Possível solução (sem typedef)

```
#include<stdio.h>

struct aluno {
    char nome [35];
    float n1, n2, CR;
};

void main (){
    struct aluno novoaluno;
    printf("Digite o nome e as duas notas do aluno: ");
    scanf(“%s %f %f",novoaluno.nome,&novoaluno.n1,&novoaluno.n2);
    novoaluno.CR=(novoaluno.n1 + novoaluno.n2) / 2;
    printf(“O CR do %s eh: %f",novoaluno.nome,novoaluno.CR);
    return;
}
```

Comparando Java e C

```
public class PesoAltura {
  int peso; // peso em quilogramas
  int altura; // altura em centimetros
}
```

Comparando Java e C

```
public class PesoAltura {
  int peso; // peso em quilogramas
  int altura; // altura em centimetros
}

typedef struct{
   int peso; // peso em quilogramas
   int altura; // altura em centimetros
} PesoAltura;
```

Comparando Java e C

```
public class PesoAltura {
  int peso; // peso em quilogramas
  int altura; // altura em centimetros
}
```

```
typedef struct{
   int peso; // peso em quilogramas
   int altura; // altura em centimetros
} PesoAltura;
```

```
   public static final int alturaMaxima = 225;
```

Comparando Java e C

```
public class PesoAltura {
  int peso; // peso em quilogramas
  int altura; // altura em centimetros
}
```

```
typedef struct{
   int peso; // peso em quilogramas
   int altura; // altura em centimetros
} PesoAltura;
```

```
   public static final int alturaMaxima = 225;
```

```
   #define alturaMaxima 225
```

Comparando Java e C

```
public static void main(String[] args) {
```

Comparando Java e C

```
public static void main(String[] args) {
```

```
int main() {
//...
return 0;
```

Comparando Java e C

```
public static void main(String[] args) {
```

```
int main() {
//...
return 0;
```

```
   PesoAltura pessoa1 = new PesoAltura();
   pessoa1.peso = 80;
   pessoa1.altura = 185;
```

Comparando Java e C

```
public static void main(String[] args) {
```

```
int main() {
//...
return 0;
```

```
   PesoAltura pessoa1 = new PesoAltura();
   pessoa1.peso = 80;
   pessoa1.altura = 185;
```

```
   PesoAltura pessoa1;
   pessoa1.peso = 80;
   pessoa1.altura = 185;
```

Comparando Java e C

```
System.out.print("Peso: " + pessoa1.peso + ", Altura "+pessoa1.altura+". ");
```

Comparando Java e C

```
System.out.print("Peso: " + pessoa1.peso + ", Altura "+pessoa1.altura+". ");
```

```
printf("Peso: %i, Altura %i. ", pessoa1.peso, pessoa1.altura);
```

Comparando Java e C

```
System.out.print("Peso: " + pessoa1.peso + ", Altura "+pessoa1.altura+". ");

printf("Peso: %i, Altura %i. ", pessoa1.peso, pessoa1.altura);


if (pessoa1.altura>alturaMaxima) System.out.println("Altura acima da maxima.");
else System.out.println("Altura abaixo da maxima.");
```

Comparando Java e C

```
System.out.print("Peso: " + pessoa1.peso + ", Altura "+pessoa1.altura+". ");

printf("Peso: %i, Altura %i. ", pessoa1.peso, pessoa1.altura);


if (pessoa1.altura>alturaMaxima) System.out.println("Altura acima da maxima.");
else System.out.println("Altura abaixo da maxima.");

if (pessoa1.altura>alturaMaxima) printf("Altura acima da maxima.\n");
else printf("Altura abaixo da maxima.\n");
```

Código em C

```
#include <stdio.h>
#define alturaMaxima 225

typedef struct{
   int peso; // peso em quilogramas
   int altura; // altura em centimetros
} PesoAltura;

int main() {
   PesoAltura pessoa1;
   pessoa1.peso = 80;
   pessoa1.altura = 185;
   printf("Peso: %i, Altura %i. ", pessoa1.peso, pessoa1.altura);
   if (pessoa1.altura>alturaMaxima) printf("Altura acima da maxima.\n");
   else printf("Altura abaixo da maxima.\n");
   return 0;
}
```

Código em C

```
#include <stdio.h>
#define alturaMaxima 225

typedef struct{
   int peso; // peso em quilogramas
   int altura; // altura em centimetros
} PesoAltura;

int main() {
   PesoAltura pessoa1;
   pessoa1.peso = 80;
   pessoa1.altura = 185;
   printf("Peso: %i, Altura %i. ", pessoa1.peso, pessoa1.altura);
   if (pessoa1.altura>alturaMaxima) printf("Altura acima da maxima.\n");
   else printf("Altura abaixo da maxima.\n");
   return 0;
}
```

```
$ Peso: 80, Altura 185. Altura abaixo da maxima.
```

Uso de memória

Uso de memória em Java:

```
public class PesoAltura {
  int peso; // peso em quilogramas
  int altura; // altura em centimetros
}

public class EstruturaSimples {
   public static final int alturaMaxima = 225;
   public static void main(String[] args) {
      PesoAltura pessoa1 = new PesoAltura();
      pessoa1.peso = 80;
      pessoa1.altura = 185;
      System.out.print("Peso: " + pessoa1.peso + ", Altura "+pessoa1.altura+". ");
      if (pessoa1.altura>alturaMaxima) System.out.println("Altura acima da maxima.");
      else System.out.println("Altura abaixo da maxima.");
   }
}
```

Uso de memória

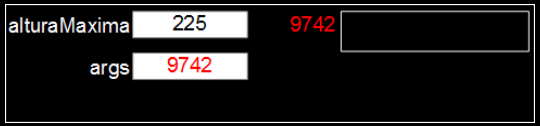
Uso de memória em Java:

```
public class PesoAltura {
  int peso; // peso em quilogramas
  int altura; // altura em centimetros
}

public class EstruturaSimples {
   public static final int alturaMaxima = 225;
   public static void main(String[] args) {
      PesoAltura pessoa1 = new PesoAltura();
      pessoa1.peso = 80;
      pessoa1.altura = 185;
      System.out.print("Peso: " + pessoa1.peso + ", Altura "+pessoa1.altura+". ");
      if (pessoa1.altura>alturaMaxima) System.out.println("Altura acima da maxima.");
      else System.out.println("Altura abaixo da maxima.");
   }
}
```

Uso de memória

Uso de memória em Java:

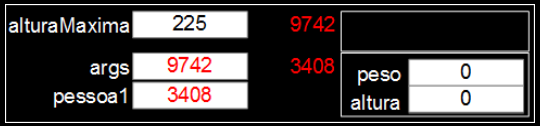
```
public class PesoAltura {
  int peso; // peso em quilogramas
  int altura; // altura em centimetros
}

public class EstruturaSimples {
   public static final int alturaMaxima = 225;
   public static void main(String[] args) {
      PesoAltura pessoa1 = new PesoAltura();
      pessoa1.peso = 80;
      pessoa1.altura = 185;
      System.out.print("Peso: " + pessoa1.peso + ", Altura "+pessoa1.altura+". ");
      if (pessoa1.altura>alturaMaxima) System.out.println("Altura acima da maxima.");
      else System.out.println("Altura abaixo da maxima.");
   }
}
```



Uso de memória

Uso de memória em Java:

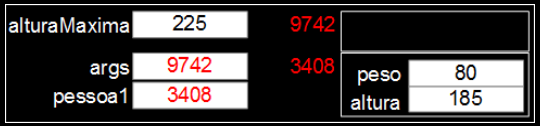
```
public class PesoAltura {
  int peso; // peso em quilogramas
  int altura; // altura em centimetros
}

public class EstruturaSimples {
   public static final int alturaMaxima = 225;
   public static void main(String[] args) {
      PesoAltura pessoa1 = new PesoAltura();
      pessoa1.peso = 80;
      pessoa1.altura = 185;
      System.out.print("Peso: " + pessoa1.peso + ", Altura "+pessoa1.altura+". ");
      if (pessoa1.altura>alturaMaxima) System.out.println("Altura acima da maxima.");
      else System.out.println("Altura abaixo da maxima.");
   }
}
```



Uso de memória

Uso de memória em Java:

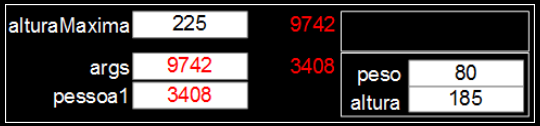
```
public class PesoAltura {
  int peso; // peso em quilogramas
  int altura; // altura em centimetros
}

public class EstruturaSimples {
   public static final int alturaMaxima = 225;
   public static void main(String[] args) {
      PesoAltura pessoa1 = new PesoAltura();
      pessoa1.peso = 80;
      pessoa1.altura = 185;
      System.out.print("Peso: " + pessoa1.peso + ", Altura "+pessoa1.altura+". ");
      if (pessoa1.altura>alturaMaxima) System.out.println("Altura acima da maxima.");
      else System.out.println("Altura abaixo da maxima.");
   }
}
```



Uso de memória

Uso de memória em Java:

```
public class PesoAltura {
  int peso; // peso em quilogramas
  int altura; // altura em centimetros
}

public class EstruturaSimples {
   public static final int alturaMaxima = 225;
   public static void main(String[] args) {
      PesoAltura pessoa1 = new PesoAltura();
      pessoa1.peso = 80;
      pessoa1.altura = 185;
      System.out.print("Peso: " + pessoa1.peso + ", Altura "+pessoa1.altura+". ");
      if (pessoa1.altura>alturaMaxima) System.out.println("Altura acima da maxima.");
      else System.out.println("Altura abaixo da maxima.");
   }
}
```




```
$ Peso: 80, Altura 185. Altura abaixo da maxima.
```

Uso de memória

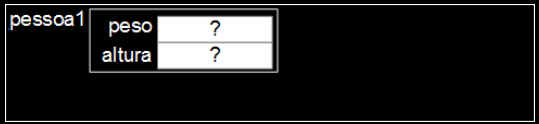
Uso de memória em C:

```
#include <stdio.h>
#define alturaMaxima 225

typedef struct{
   int peso; // peso em quilogramas
   int altura; // altura em centimetros
} PesoAltura;

int main() {
   PesoAltura pessoa1;
   pessoa1.peso = 80;
   pessoa1.altura = 185;
   printf("Peso: %i, Altura %i. ", pessoa1.peso, pessoa1.altura);
   if (pessoa1.altura>alturaMaxima) printf("Altura acima da maxima.\n");
   else printf("Altura abaixo da maxima.\n");
   return 0;
}
```

Uso de memória

Uso de memória em C:

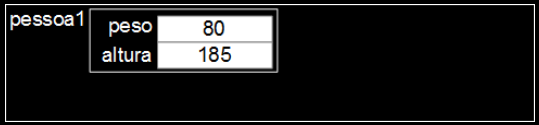
```c
#include <stdio.h>
#define alturaMaxima 225

typedef struct{
   int peso; // peso em quilogramas
   int altura; // altura em centimetros
} PesoAltura;

int main() {
   PesoAltura pessoa1;
   pessoa1.peso = 80;
   pessoa1.altura = 185;
   printf("Peso: %i, Altura %i. ", pessoa1.peso, pessoa1.altura);
   if (pessoa1.altura>alturaMaxima) printf("Altura acima da maxima.\n");
   else printf("Altura abaixo da maxima.\n");
   return 0;
}
```



Uso de memória

Uso de memória em C:

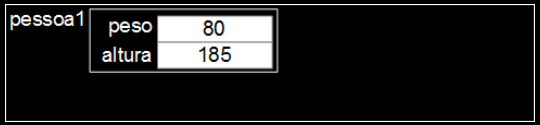
```c
#include <stdio.h>
#define alturaMaxima 225

typedef struct{
   int peso; // peso em quilogramas
   int altura; // altura em centimetros
} PesoAltura;

int main() {
   PesoAltura pessoa1;
   pessoa1.peso = 80;
   pessoa1.altura = 185;
   printf("Peso: %i, Altura %i. ", pessoa1.peso, pessoa1.altura);
   if (pessoa1.altura>alturaMaxima) printf("Altura acima da maxima.\n");
   else printf("Altura abaixo da maxima.\n");
   return 0;
}
```



Uso de memória

Uso de memória em C:

```
#include <stdio.h>
#define alturaMaxima 225

typedef struct{
   int peso; // peso em quilogramas
   int altura; // altura em centimetros
} PesoAltura;

int main() {
   PesoAltura pessoa1;
   pessoa1.peso = 80;
   pessoa1.altura = 185;
   printf("Peso: %i, Altura %i. ", pessoa1.peso, pessoa1.altura);
   if (pessoa1.altura>alturaMaxima) printf("Altura acima da maxima.\n");
   else printf("Altura abaixo da maxima.\n");
   return 0;
}
```




```
$ Peso: 80, Altura 185. Altura abaixo da maxima.
```

Uso de memória

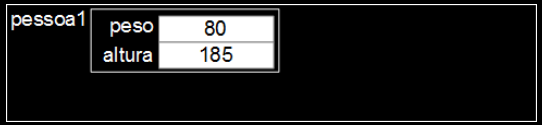
Por que o uso de memória foi tão diferente nas duas soluções?

Uso de memória

Por que o uso de memória foi tão diferente nas duas soluções?

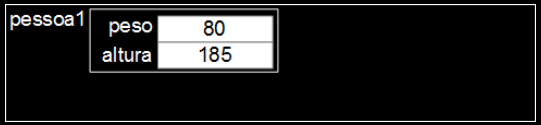
Uso de memória

Por que o uso de memória foi tão diferente nas duas soluções?



Uso de memória

Por que o uso de memória foi tão diferente nas duas soluções?



Um dos motivos é que nossa implementação em C não corresponde totalmente à implementação em Java

Ponteiros e alocação de memória

Em C há uma distinção bastante explícita entre um tipo (ou uma estrutura) e um endereço:

- ***int x;*** **significa que *x* é uma variável do tipo inteiro**
- ***int* y;*** **significa que *y* é uma variável do tipo endereço para inteiro**
- **O asterisco (*) após a palavra *int* indica que estamos falando de um endereço para inteiro e não mais de um inteiro**

Ponteiros e alocação de memória

```
#include <stdio.h>
int main(){
   int x = 25;
   int* y = &x;
   *y = 30;
   printf("Valor atual de x: %i\n", x);
   return 0;
}
```

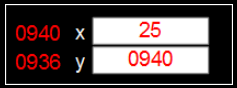
Ponteiros e alocação de memória

```
#include <stdio.h>
int main(){
   int x = 25;
   int* y = &x;
   *y = 30;
   printf("Valor atual de x: %i\n", x);
   return 0;
}
```

A variável x é inicializada com valor 25.

Ponteiros e alocação de memória

```c
#include <stdio.h>
int main(){
   int x = 25;
   int* y = &x;
   *y = 30;
   printf("Valor atual de x: %i\n", x);
   return 0;
}
```

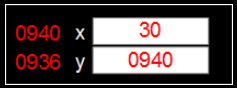


A variável x é inicializada com valor 25.

A variável y recebe o endereço onde está a variável x.

Ponteiros e alocação de memória

```c
#include <stdio.h>
int main(){
   int x = 25;
   int* y = &x;
   *y = 30;
   printf("Valor atual de x: %i\n", x);
   return 0;
}
```



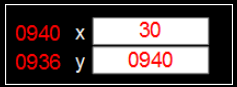
A variável x é inicializada com valor 25.

A variável y recebe o endereço onde está a variável x.

Coloca-se o valor 30 na posição de memória referenciada (apontada) pelo endereço armazenado em y.

Ponteiros e alocação de memória

```c
#include <stdio.h>
int main(){
   int x = 25;
   int* y = &x;
   *y = 30;
   printf("Valor atual de x: %i\n", x);
   return 0;
}
```

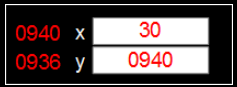



```
$ Valor atual de x: 30
```

A variável x é inicializada com valor 25.
A variável y recebe o endereço onde está a variável x.
Coloca-se o valor 30 na posição de memória referenciada (apontada) pelo endereço armazenado em y.

Ponteiros e alocação de memória

```c
#include <stdio.h>
int main(){
   int x = 25;
   int* y = &x;
   *y = 30;
   printf("Valor atual de x: %i\n", x);
   return 0;
}
```




```
$ Valor atual de x: 30
```

A variável x é inicializada com valor 25.
A variável y recebe o endereço onde está a variável x.
Coloca-se o valor 30 na posição de memória referenciada (apontada) pelo endereço armazenado em y.

Ponteiros e alocação de memória

Em C há uma função para alocação de memória: malloc (*memory allocation*)

- **recebe como parâmetro o número de bytes que se deseja alocar**
- **retorna o endereço inicial do bloco de bytes que foi alocado, porém esse retorno tem o tipo *void** (ponteiro para void)**
- **há um operador chamado sizeof que recebe como parâmetro um tipo (simples ou composto) e retorna a quantidade de bytes ocupada por esse tipo**

Função malloc

```
#include <stdio.h>
#include <malloc.h>
int main(){
   int* y = (int*) malloc(sizeof(int));
   *y = 20;
   int z = sizeof(int);
   printf("*y=%i z=%i\n", *y, z);
   return 0;
}
```

Função malloc

```
#include <stdio.h>
#include <malloc.h>
int main(){
   int* y = (int*) malloc(sizeof(int));
   *y = 20;
   int z = sizeof(int);
   printf("*y=%i z=%i\n", *y, z);
   return 0;
}
```

Função malloc

```c
#include <stdio.h>
#include <malloc.h>
int main(){
   int* y = (int*) malloc(sizeof(int));
   *y = 20;
   int z = sizeof(int);
   printf("*y=%i z=%i\n", *y, z);
   return 0;
}
```

Função malloc

```
#include <stdio.h>
#include <malloc.h>
int main(){
   int* y = (int*) malloc(sizeof(int));
   *y = 20;
   int z = sizeof(int);
   printf("*y=%i z=%i\n", *y, z);
   return 0;
}
```

Função malloc

```
#include <stdio.h>
#include <malloc.h>
int main(){
   int* y = (int*) malloc(sizeof(int));
   *y = 20;
   int z = sizeof(int);
   printf("*y=%i z=%i\n", *y, z);
   return 0;
}
```

Função malloc

```
#include <stdio.h>
#include <malloc.h>
int main(){
   int* y = (int*) malloc(sizeof(int));
   *y = 20;
   int z = sizeof(int);
   printf("*y=%i z=%i\n", *y, z);
   return 0;
}
```

Função malloc

```
#include <stdio.h>
#include <malloc.h>
int main(){
   int* y = (int*) malloc(sizeof(int));
   *y = 20;
   int z = sizeof(int);
   printf("*y=%i z=%i\n", *y, z);
   return 0;
}
```

Função malloc

```
#include <stdio.h>
#include <malloc.h>
int main(){
   int* y = (int*) malloc(sizeof(int));
   *y = 20;
   int z = sizeof(int);
   printf("*y=%i z=%i\n", *y, z);
   return 0;
}
```

Função malloc

```
#include <stdio.h>
#include <malloc.h>
int main(){
   int* y = (int*) malloc(sizeof(int));
   *y = 20;
   int z = sizeof(int);
   printf("*y=%i z=%i\n", *y, z);
   return 0;
}
```

Saída:

```
$ *y=20 z=4
```

Nossa segunda implementação

```
#include <stdio.h>
#define alturaMaxima 225
```

Nossa segunda implementação

```
#include <stdio.h>
#define alturaMaxima 225
```

```
#include <stdio.h>
#include <malloc.h>
#define alturaMaxima 225
```

Nossa segunda implementação

```
#include <stdio.h>
#define alturaMaxima 225
```

```
#include <stdio.h>
#include <malloc.h>
#define alturaMaxima 225


typedef struct{
   int peso; // peso em quilogramas
   int altura; // altura em centimetros
} PesoAltura;
```

Nossa segunda implementação

```
#include <stdio.h>
#define alturaMaxima 225

#include <stdio.h>
#include <malloc.h>
#define alturaMaxima 225


typedef struct{
   int peso; // peso em quilogramas
   int altura; // altura em centimetros
} PesoAltura;

typedef struct{
   int peso; // peso em quilogramas
   int altura; // altura em centimetros
} PesoAltura;
```

Nossa segunda implementação

```
int main() {
```

Nossa segunda implementação

```
int main() {
```

```
int main() {
```

Nossa segunda implementação

```
int main() {
```

```
int main() {


   PesoAltura pessoa1;
   pessoa1.peso = 80;
   pessoa1.altura = 185;
```

Nossa segunda implementação

```
int main() {
```

```
int main() {


   PesoAltura pessoa1;
   pessoa1.peso = 80;
   pessoa1.altura = 185;

   PesoAltura* pessoa1 = (PesoAltura*) malloc(sizeof(PesoAltura));
   pessoa1->peso = 80;
   pessoa1->altura = 185;
```

Nossa segunda implementação

```
printf("Peso: %i, Altura %i. ", pessoa1.peso, pessoa1.altura);
```

Nossa segunda implementação

```
printf("Peso: %i, Altura %i. ", pessoa1.peso, pessoa1.altura);

printf("Peso: %i, Altura %i. ", pessoa1->peso, pessoa1->altura);
```

Nossa segunda implementação

```
printf("Peso: %i, Altura %i. ", pessoa1.peso, pessoa1.altura);

printf("Peso: %i, Altura %i. ", pessoa1->peso, pessoa1->altura);


if (pessoa1.altura>alturaMaxima) printf("Altura acima da maxima.\n");
else printf("Altura abaixo da maxima.\n");
```

Nossa segunda implementação

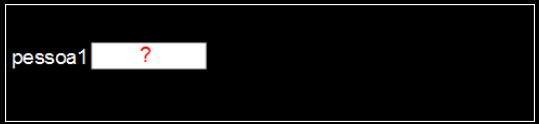
```
printf("Peso: %i, Altura %i. ", pessoa1.peso, pessoa1.altura);

printf("Peso: %i, Altura %i. ", pessoa1->peso, pessoa1->altura);


if (pessoa1.altura>alturaMaxima) printf("Altura acima da maxima.\n");
else printf("Altura abaixo da maxima.\n");

if (pessoa1->altura>alturaMaxima) printf("Altura acima da maxima.\n");
else printf("Altura abaixo da maxima.\n");
```

Nossa segunda implementação

```
#include <stdio.h>
#define alturaMaxima 225

typedef struct{
   int peso; // peso em quilogramas
   int altura; // altura em centimetros
} PesoAltura;

int main() {
   PesoAltura* pessoa1 = (PesoAltura*) malloc(sizeof(PesoAltura));
   pessoa1->peso = 80;
   pessoa1->altura = 185;
   printf("Peso: %i, Altura %i. ", pessoa1->peso, pessoa1->altura);
   if (pessoa1->altura>alturaMaxima) printf("Altura acima da maxima.\n");
   else printf("Altura abaixo da maxima.\n");
   return 0;
}
```

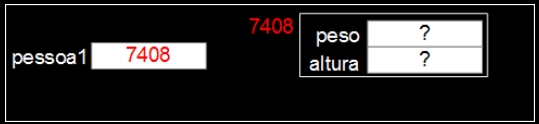
Nossa segunda implementação

```c
#include <stdio.h>
#define alturaMaxima 225

typedef struct{
   int peso; // peso em quilogramas
   int altura; // altura em centimetros
} PesoAltura;

int main() {
   PesoAltura* pessoa1 = (PesoAltura*) malloc(sizeof(PesoAltura));
   pessoa1->peso = 80;
   pessoa1->altura = 185;
   printf("Peso: %i, Altura %i. ", pessoa1->peso, pessoa1->altura);
   if (pessoa1->altura>alturaMaxima) printf("Altura acima da maxima.\n");
   else printf("Altura abaixo da maxima.\n");
   return 0;
}
```



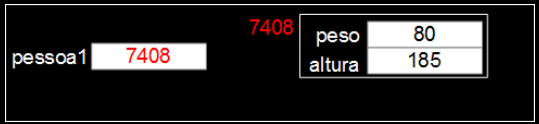
Nossa segunda implementação

```c
#include <stdio.h>
#define alturaMaxima 225

typedef struct{
   int peso; // peso em quilogramas
   int altura; // altura em centimetros
} PesoAltura;

int main() {
   PesoAltura* pessoa1 = (PesoAltura*) malloc(sizeof(PesoAltura));
   pessoa1->peso = 80;
   pessoa1->altura = 185;
   printf("Peso: %i, Altura %i. ", pessoa1->peso, pessoa1->altura);
   if (pessoa1->altura>alturaMaxima) printf("Altura acima da maxima.\n");
   else printf("Altura abaixo da maxima.\n");
   return 0;
}
```



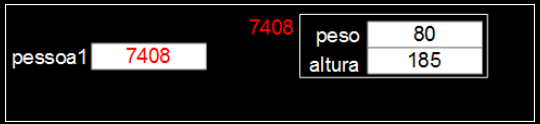
Nossa segunda implementação

```c
#include <stdio.h>
#define alturaMaxima 225

typedef struct{
   int peso; // peso em quilogramas
   int altura; // altura em centimetros
} PesoAltura;

int main() {
   PesoAltura* pessoa1 = (PesoAltura*) malloc(sizeof(PesoAltura));
   pessoa1->peso = 80;
   pessoa1->altura = 185;
   printf("Peso: %i, Altura %i. ", pessoa1->peso, pessoa1->altura);
   if (pessoa1->altura>alturaMaxima) printf("Altura acima da maxima.\n");
   else printf("Altura abaixo da maxima.\n");
   return 0;
}
```



Nossa segunda implementação

```
#include <stdio.h>
#define alturaMaxima 225

typedef struct{
   int peso; // peso em quilogramas
   int altura; // altura em centimetros
} PesoAltura;

int main() {
   PesoAltura* pessoa1 = (PesoAltura*) malloc(sizeof(PesoAltura));
   pessoa1->peso = 80;
   pessoa1->altura = 185;
   printf("Peso: %i, Altura %i. ", pessoa1->peso, pessoa1->altura);
   if (pessoa1->altura>alturaMaxima) printf("Altura acima da maxima.\n");
   else printf("Altura abaixo da maxima.\n");
   return 0;
}
```




```
$ Peso: 80, Altura 185. Altura abaixo da maxima.
```

AULA 02
ESTRUTURA DE DADOS

Criação de uma primeira estrutura

Norton T. Roman & Luciano A. Digiampietri