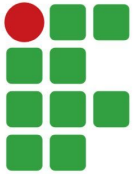


INSTITUTO FEDERAL
Bahia
Campus Santo Antônio de Jesus

http://www.portal.ifba.edu.br/santoantonio

LINGUAGEM DE PROGRAMAÇÃO

Prof. George Pacheco Pinto

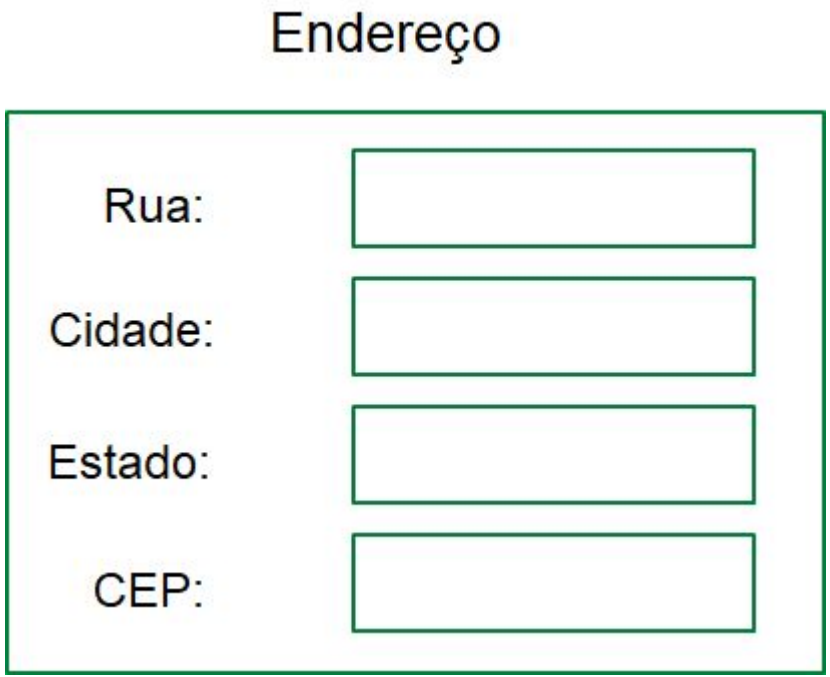
Agenda

- Linguagem C
  - Estruturas
  - Definição
  - Sintaxe
  - Exemplos
  - Exercícios

Estruturas

- Coleções de dados heterogêneos agrupados em uma mesma estrutura de dados;
- Representa uma coleção de variáveis referenciadas por um nome;
- As variáveis que compreendem a estrutura são chamados de membros da estrutura (elementos ou campos);

ESTRUTURAS



EXEMPLO

```
struct addr
{
    char street [40];
    char city[20];
    char state[3];
    int zip;
};
```

Até então só a forma dos dados foi definida. Para declarar uma variável do tipo addr:

struct addr endereco;

EXEMPLO

```
struct addr
{
    char street [40];
    char city[20];
    char state[3];
    int zip;
} addrinfo, binfo, cinfo;
```

```
struct
{
    char street [40];
    char city[20];
    char state[3];
    int zip;
} addrinfo;
```

EXEMPLO

```
typedef struct
{
    char street [40];
    char city[20];
    char state[3];
    int zip;
} ENDERECO;



ENDERECO end1, end2;
```

SINTAXE

```
struct <identificador> {
    tipo <nome_variavel>;
    tipo <nome_variavel>;
    .
    .
} <variaveis_estrutura>;
```

Identificador ou variaveis_estrutura podem ser omitidos, mas não ambos

REFERENCIANDO ELEMENTO

```
nome_da_variaval_estrutura.nome_do_elemento

addrinfo.zip = 12345
printf (“%d”, addrinfo.zip);
gets (addrinfo.street);
```

Atribuição de Estruturas

```
main() {
  struct {
    int a;
    int b;
  } x, y;

  x.a = 10;
  y = x; /* atribui uma estrutura a outra */
  printf("%d", y.a);
}
```

EXEMPLO

```
#include <stdio.h>
typedef struct {
    char rua[50];
    int num;
    char bairro[50];
    char cidade [50];
    char estado [50];
    char cep [10];
} ENDERECO;
int main(){
    ENDERECO e1;
    printf ("Digite a rua: ");
    gets (e1.rua);
    printf ("Digite o numero: ");
    scanf ("%d", &e1.num);
    fflush(stdin);
    printf ("Digite o bairro: ");
    gets (e1.bairro);
    fflush(stdin);
    printf ("Digite a cidade: ");
    gets (e1.cidade);
    fflush(stdin);
    printf ("Digite o estado: ");
    gets (e1.estado);
    fflush(stdin);
    printf ("Digite o cep: ");
    gets (e1.cep);
    printf ("Dados Digitados\n");
    printf ("%s, %d. %s\n", e1.rua,
e1.num, e1.bairro);
    printf ("%s\n", e1.cidade);
    printf ("%s\n", e1.estado);
    printf ("%s\n", e1.cep);
}
```

MATRIZES DE ESTRUTURAS

- Talvez o uso mais comum de estruturas seja em matriz de estrutura;
- Para declarar uma matriz de estrutura, primeiro deve-se definir uma estrutura e em seguida declarar uma variável matriz desse tipo;
- Ex:
  ENDERECO addrinfo[100];

printf (“%d”, addrinfo[2].zip);

EXEMPLO

```
for(i=0; i<100; i++) {
  printf("entre a rua: ");
  gets(addrinfo[i].street);
  printf("entre a cidade: ");
  gets(addrinfo[i].city);
  printf("entre o estado: ");
  gets(addrinfo[i].state);
}
```

```
for(i=0; i<100; i++) {
    printf("%s\n", addrinfo[i].street);
    printf("%s\n", addrinfo[i].city);
    printf("%s\n", addrinfo[i].state);
    printf("%lu\n", addrinfo[i].zip);
  }
```

Passando Elementos de Estruturas para Funções

```
typedef struct {
    char x;
    int y;
    float z;
    char s[10];
} PESSOA;

PESSOA mike;
```

```
Passando Valor
func (mike.x);
func2 (mike.y);
func3 (mike.z);
func4 (mike.s);
func (mike.s[2]);

Passando Endereço
func (&mike.x);
func2 (&mike.y);
func3 (&mike.z);
func4 (mike.s);
func (&mike.s[2]);
```

Passando Estruturas Inteiras para Funções

```
/* Define um tipo de
estrutura. */

typedef struct {
  int a, b;
  char ch;
} TESTE;

void f1(TESTE parm);
```

```
int main(void)
{
  TESTE arg;
  arg.a = 1000;

  f1(arg);
}

void f1(TESTE parm)
{
  printf("%d", parm.a);
}
```

Passando Estruturas Inteiras para Funções

```
typedef struct {
  int a, b;
  char ch;
} TESTE1;

typedef struct {
  int a, b;
  char ch;
} TESTE2;

void f1(TESTE2 parm);
```

```
int main(void)
{
  TESTE1 arg;
  arg.a = 1000;

  f1(arg);
}

void f1(TESTE2 parm)
{
  printf("%d", parm.a);
}
```

Passando Estruturas Inteiras para Funções

```
typedef struct {
  int a, b;
  char ch;
} TESTE1;

typedef struct {
  int a, b;
  char ch;
} TESTE2;

void f1(TESTE2 parm);
```

```
int main(void)
{
  TESTE1 arg;
  arg.a = 1000;

  f1(arg);
}

void f1(TESTE2 parm)
{
  printf("%d", parm.a);
}
```

Ponteiros para Estruturas

- Usos:
  - Fazer chamadas por referência
    - Passar estruturas inteiras para funções pode ser um gasto de tempo excessivo e desnecessário. Redução na performance;
    - Muitas vezes a função precisa referenciar o argumento real em lugar de uma cópia.
  - Criar estruturas de dados dinâmicas

```
// Declarando o ponteiro para a estrutura
ENDERECO *paddr;
```

Ponteiros para Estruturas

```
typedef struct {
    float balance;
    char name[80];
} PERSON;

PERSON *p, pessoa;
p = &pessoa;

p -> balance // acessa o campo balance
```

Matrizes e Estruturas dentro de Estruturas

```
struct c {
    int a[10][10];
    float b;
} y;

y.a[3][7] //referencia o elemento 3,7 da matriz
```

Matrizes e Estruturas dentro de Estruturas

```
struct empregado {
    struct addr ender;
    float wage;
} e;

e.ender.zip = 12345; // referenciando elementos
```

Referências

Consultar ementário.