unipampa

Universidade Federal do Pampa

Laboratório de Programação I

Assunto de Hoje
Estruturas Compostas (structs)
Parte II

Professor Luciano Brum
lucianobrum@unipampa.edu.br

unipampa
Universidade Federal do Pampa

Roteiro

- Alocação dinâmica de estruturas.
- Vetores de ponteiros para estruturas.
- Tipo União.
- Tipo Enumeração.
- Resumo.

Alocação Dinâmica

- Até o momento, vimos como criar estruturas de forma estática.

- Exemplo: struct aluno estudante1;

Alocação Dinâmica

- Deste modo, estamos utilizando o espaço de memória necessário para essa estrutura.

- Ao usar struct aluno estudantes[100], reservamos um espaço de memória para armazenar 100 estudantes, mesmo que não sejam utilizadas, de fato, todas essas estruturas.

Alocação Dinâmica

- Para evitar o desperdício de memória, podemos alocar estruturas de forma dinâmica.

- Como fazemos alocação dinâmica com variáveis de tipos básicos?

Alocação Dinâmica

```
int *variavel;
```

Alocação Dinâmica

```
int *variavel;
variavel = (int*)malloc(sizeof(int));
```

Alocação Dinâmica

```
int *variavel;
variavel = (int*)malloc(sizeof(int));
int x = 10;
```

Alocação Dinâmica

```
int *variavel;
variavel = (int*)malloc(sizeof(int));
int x = 10;
variavel = &x;
```

Alocação Dinâmica

```
int *variavel;
variavel = (int*)malloc(sizeof(int));
int x = 10;
variavel = &x;
*variável = 20;
```

Alocação Dinâmica

```
int *variavel;
variavel = (int*)malloc(sizeof(int));
int x = 10;
variavel = &x;
*variável = 20;
printf(“x = %d”, x);
```

Alocação Dinâmica

```
int *variavel;
variavel = (int*)malloc(sizeof(int));
int x = 10;
variavel = &x;
*variável = 20;
printf("x = %d", x);
```

Quanto vale x??

Alocação Dinâmica

- Como alocar uma struct dinamicamente?

Alocação Dinâmica

- Como alocar uma struct dinamicamente?

```
struct aluno{
    int mat;
    char nome[80];
    char end[120];
    char tel[20];
};
typedef struct aluno Aluno;
```

Alocação Dinâmica

- Como alocar uma struct dinamicamente?
- No main...

Alocação Dinâmica

- Como alocar uma struct dinamicamente?
- No main...

```
Aluno* alunos;
inicializa(alunos);
preenche(alunos);
retira(alunos);
```

Alocação Dinâmica

- Procedimento inicializa:

```
void inicializa(Aluno* a){
        a = NULL;
}
```

Alocação Dinâmica

- Procedimento preenche:

```
void preenche(Aluno* a){
        if(a!=NULL){
                printf("Posição já cadastrada.\n");
                return;
        }
        else{
                ...
```

Alocação Dinâmica

- Procedimento preenche:

```
void preenche(Aluno* a){
        if(a!=NULL){
                printf("Posição já cadastrada.\n");
                return;
        }
        else{
                ...
```

Alocação Dinâmica

- Procedimento preenche:

```
void preenche(Aluno* a){
        if(a[i]!=NULL){
                printf("Posição já cadastrada.\n");
                return;}
        else{
                a = (Aluno*)malloc(sizeof(Aluno));
                printf("Matrícula:");
                scanf("%d", &a->mat);
                printf("Nome:");
                scanf(" %80[^\n]",&a->nome);
                printf("Endereço:");
                scanf(" %120[^\n]",&a->end);
                printf("Telefone:");
                scanf(" %20[^\n]",&a->tel);}
}
```

Alocação Dinâmica

- Procedimento retira:

```
void retira(Aluno* a){
        if(a!=NULL){
                free(a);
                a=NULL;
        }
}
```

Alocação Dinâmica

- Procedimento retira:

```
void retira(Aluno* a){
        if(a!=NULL){
                free(a);
                a=NULL;
        }
}
```

unipampa
Universidade Federal do Pampa

Roteiro

- Alocação dinâmica de estruturas.
- Vetores de ponteiros para estruturas.
- Tipo União.
- Tipo Enumeração.
- Resumo.

Vetor de Ponteiros para Structs

- E se desejássemos cadastrar 100 alunos?

- E se o endereço fosse uma estrutura?

Vetor de Ponteiros para Structs

- E se desejássemos cadastrar 100 alunos?
- E se o endereço fosse uma estrutura?

```
struct end{
        char rua[50];
        int num;
};
typedef struct end End;
struct aluno{
    int mat;
    char nome[80];
    End endereco;
    char tel[20];
};
typedef struct aluno Aluno;
```

Vetor de Ponteiros para Structs

- Quais alterações no inicializa?
- Quais alterações no preenche?
- Quais alterações no retira?

Vetor de Ponteiros para Structs

- No main:

```
int tam = 100;
Aluno* alunos[tam];
inicializa(tam,alunos);
preenche(tam,alunos,pos);
retira(tam,alunos,pos);
```

Vetor de Ponteiros para Structs

- Procedimento inicializa:

```
void inicializa(int tam, Aluno** a){
        int i;
        for(i=0;i<tam;i++){
                a[i] = NULL;
        }
}
```

Vetor de Ponteiros para Structs

- Procedimento preenche:

```
void preenche(int n, Aluno** a, int i){
        if(i<0 || i>n){
                printf("Índice fora dos limites do vetor.\n");
                exit(1);
        }
        if(a[i]!=NULL){
                printf("Posição já cadastrada.\n");
                return;
        }
        else{
                a[i] = (Aluno*)malloc(sizeof(Aluno));
                printf("Matrícula:");
                scanf("%d", &a[i]->mat);
                printf("Nome:");
                scanf(" %80[^\n]",&a[i]->nome);
                printf("Rua:");
                scanf(" %50[^\n]",&a[i]->endereco.rua);
                printf("Numero:");
                scanf("%d",&a[i]->endereco.num);
                printf("Telefone:");
                scanf(" %20[^\n]",&a[i]->tel);}
```

Vetor de Ponteiros para Structs

- Procedimento preenche:

```
void preenche(int n, Aluno** a, int i){
        if(i<0 || i>n){
                printf("Índice fora dos limites do vetor.\n");
                exit(1);}
        if(a[i]!=NULL){
                printf("Posição já cadastrada.\n");
                return;}
        else{
                a[i] = (Aluno*)malloc(sizeof(Aluno));
                printf("Matrícula:");
                scanf("%d", &a[i]->mat);
                printf("Nome:");
                scanf(" %80[^\n]",&a[i]->nome);
                printf("Rua:");
                scanf(" %50[^\n]",&a[i]->endereco.rua);
                printf("Numero:");
                scanf("%d",&a[i]->endereco.num);
                printf("Telefone:");
                scanf(" %20[^\n]",&a[i]->tel);}}
```

Vetor de Ponteiros para Structs

- Procedimento retira:

```
void retira(int n, Aluno** a, int i){
        if(i<0 || i>n){
                printf("Índice fora dos limites do vetor.\n");
                exit(1);
        }
        if(a[i]!=NULL){
                free(a[i]);
                a[i]=NULL;
        }
}
```

Vetor de Ponteiros para Structs

- E se o endereço fosse um ponteiro para a estrutura?

```
struct end{
        char rua[50];
        int num;
};
typedef struct end End;
struct aluno{
    int mat;
    char nome[80];
    End* endereco;
    char tel[20];
};
typedef struct aluno Aluno;
```

Vetor de Ponteiros para Structs

- Procedimento inicializa:

```
void inicializa(int n, Aluno** a){
        int i;
        for(i=0;i<n;i++){
                a[i] = NULL;
        }
}
```

Vetor de Ponteiros para Structs

- Procedimento preenche:

```
void preenche(int n, Aluno** a, int i){
        if(i<0 || i>n){
                printf("Índice fora dos limites do vetor.\n");
                exit(1);}
        if(a[i]!=NULL){
                printf("Posição já cadastrada.\n");
                return;}
        else{
                a[i] = (Aluno*)malloc(sizeof(Aluno));
                a[i]->endereco = (End*)malloc(sizeof(End));
                printf("Matrícula:");
                scanf("%d", &a[i]->mat);
                printf("Nome:");
                scanf(" %80[^\n]",&a[i]->nome);
                printf("Rua:");
                scanf(" %50[^\n]",&a[i]->endereco->rua);
                printf("Numero:");
                scanf("%d",&a[i]->endereco->num);
                printf("Telefone:");
                scanf(" %20[^\n]",&a[i]->tel);}
```

Vetor de Ponteiros para Structs

- Procedimento retira:

```
void retira(int n, Aluno** a, int i){
        if(i<0 || i>n){
                printf("Índice fora dos limites do vetor.\n");
                exit(1);
        }
        if(a[i]!=NULL){
                free(a[i]->endereco);
                free(a[i]);
                a[i]=NULL;
        }
}
```

Exercício em Aula

- Exercício: Implementar um programa de cadastro de alunos com um menu com as seguintes alternativas:

1. Criar estrutura.
2. Inserir aluno.
3. Excluir aluno.
4. Imprimir todos os alunos.
5. Liberar estrutura e sair.

- Usar alocação dinâmica, modularização e structs. Use os exemplos dos slides ou crie um formato próprio.

Roteiro

- Alocação dinâmica de estruturas.
- Vetores de ponteiros para estruturas.
- Tipo União.
- Tipo Enumeração.
- Resumo.

Union

- Em C, o Union é um local de memória compartilhado por diferentes variáveis, que podem ser de diferentes tipos.

- Servem para armazenar valores heterogêneos em um mesmo espaço de memória.

Union

- Definição do tipo União:

```
union exemplo{
    int i;
    char c;
};
```

Union

- Definição do tipo União.

- Declaração de uma variável do tipo união:
  - Union exemplo v;
- Na memória, a variável v ocupa um espaço equivalente ao seu atributo de maior tamanho.

- A variável ‘v’ ocupa 4 bytes (int ocupa mais espaço que um char).

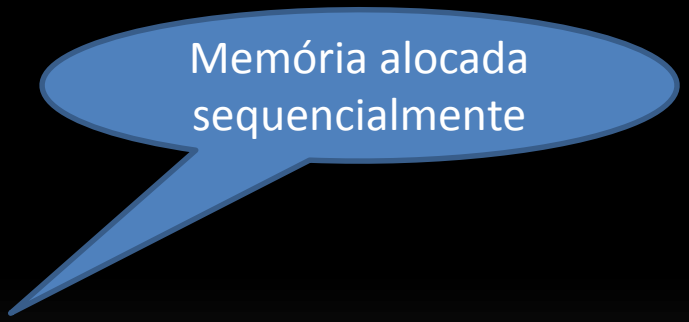
Union

- O acesso aos campos da união é similar ao acesso dos campos de uma struct:
  - v.i = 10;
  - v.c = ‘x’;
- Salienta-se que, em uma união, apenas um elemento da união pode ser armazenado por vez.
- Esta é a diferença entre unions e structs.

Union

- Exemplo:

```
struct item {
    char nome[50];
    float preco;
    float volume; //em litros
    unsigned peso; //em gramas
};
```



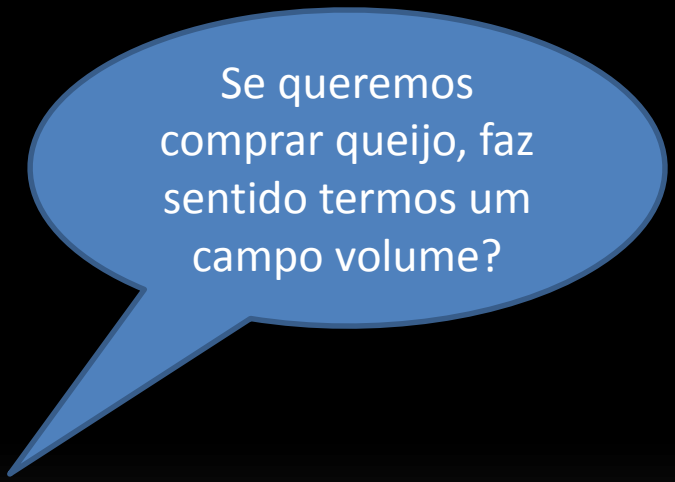
0----------49-50-------53-54--------57-58-------61

nome preco volume peso

Union

- Exemplo:

```
struct item {
    char nome[50];
    float preco;
    float volume;
    unsigned peso;
};
```



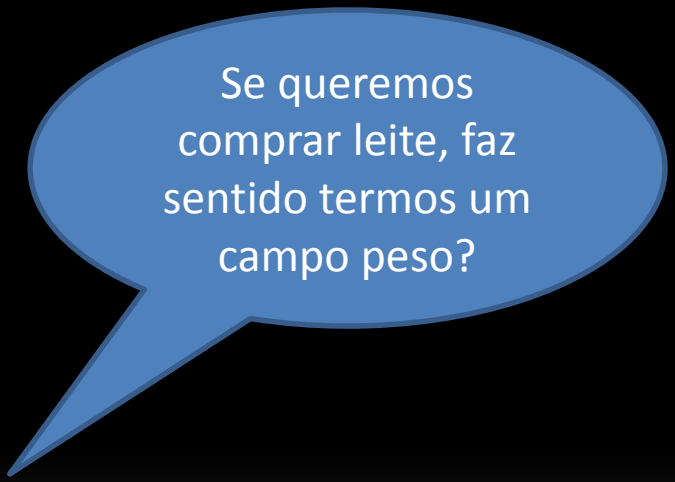
0----------49-50-------53-54--------57-58-------61

nome preco volume peso

Union

- Exemplo:

```
struct item {
    char nome[50];
    float preco;
    float volume;
    unsigned peso;
};
```



0----------49-50-------53-54--------57-58-------61

nome preco volume peso

Union

- Exemplo:

```
struct item {
    char nome[50];
    float preco;
    float volume;
    unsigned peso;
};
```

Sempre que criarmos um item, alocaríamos espaço para volume e peso, mesmo que não fossem usados.

0----------49-50-------53-54--------57-58-------61

nome preco volume peso

Union

- Exemplo:

```
union peso{
    float volume;
    unsigned int peso;
};
struct item {
    char nome[50];
    float preco;
    union peso P;
};
```

O mesmo local de memória é compartilhado pelas variáveis volume e peso.

0----------49-50-------53-54--------57

nome preco volume/peso

Union

- Exemplo:

```
struct item it;
it.P.peso = 2;
it.P.volume = 0.0f;
printf("%d", it.P.peso);
```

Don’t work.
Why?

Union

- Exemplo:

```
union peso{
    float volume;
    unsigned int peso;
};
struct item {
    char nome[50];
    float preco;
    int tipoPesagem; //0 volume, 1 peso
    union peso P;
};
```

Para cada item, criamos uma flag que indica o tipo de uso do item.

Union

- Resumo:
  - Possibilidade de armazenamento de valores de diferentes tipos em diferentes instantes do processamento.
  - Similaridade com uma struct, exceto pelo fato de todos os membros ocuparem a mesma locação de memória.
  - Possibilidade de contenção de apenas um dos seus atributos de cada vez.
  - O tamanho é em função do maior atributo da estrutura.

unipampa
Universidade Federal do Pampa

Roteiro

- Alocação dinâmica de estruturas.
- Vetores de ponteiros para estruturas.
- Tipo União.
- Tipo Enumeração.
- Resumo.

Tipo enumeração

- O tipo enum é um conjunto de constantes inteiras com nomes que especificam valores possíveis para uma variável daquele tipo.

- É uma forma mais elegante de organizar constantes.

Tipo enumeração

- Exemplo prático: boolean.

Tipo enumeração

- Exemplo prático: boolean.

- A linguagem C não possui um tipo booleano.

Tipo enumeração

- Exemplo prático: boolean.
- A linguagem C não possui um tipo booleano.
- Podemos criar um tipo booleano usando o tipo enumeração.

Tipo enumeração

```
enum bool{
    FALSE,
    TRUE
};
typedef enum bool bool;
```

Por padrão, o primeiro símbolo representa o valor 0, o segundo representa o valor 1 e assim por diante.

Tipo enumeração

```
enum bool{
    FALSE=0,
    TRUE=1
};
typedef enum bool bool;
```

Podemos também explicitar os valores dos símbolos em uma enumeração.

Tipo enumeração

- Para criar uma variável deste tipo:

  bool resultado;

- Resultado só pode assumir os valores FALSE(0) ou TRUE(1).

Tipo enumeração

```
#include<stdio.h>
int main( ){
        int     Novembro[5][7]={{0,0,1,2,3,4,5},{6,7,8,9,10,11,12},{13,14,15,16,17,
18,19},{20,21,22,23,24,25,26},{27,28,29,30,0,0,0}};
        enum dias {Domingo, Segunda, Terca, Quarta, Quinta, Sexta, Sabado};
        enum semana {semana01, semana02, semana03, semana04, semana05};
        printf (“Quinta da primeira semana de Novembro eh dia %d\n", Novembro
[semana01][Quinta]);
}
```

unipampa
Universidade Federal do Pampa

Roteiro

- Alocação dinâmica de estruturas.
- Vetores de ponteiros para estruturas.
- Tipo União.
- Tipo Enumeração.
- Resumo.

Resumo

- Vimos o que é uma estrutura (struct) e como podemos construir objetos com atributos heterogêneos na linguagem C.
- Vimos como utilizar o modificador de tipos typedef.
- Vimos como alocar memória dinamicamente para vetores de e para structs.
- Vimos o funcionamento dos tipos união (union) e enumeração (enum) na linguagem C e aplicações.

Atividade Semipresencial

1. Defina uma estrutura que irá representar bandas de música. Essa estrutura deve ter:
   I. o nome da banda;
   II. que tipo de música ela toca;
   III. o número de integrantes e;
   IV. em que posição do ranking essa banda está dentre as suas 5 bandas favoritas.

No programa principal, organize um menu com as opções relativas aos exercícios 2, 3, 4, 5 e 6. Caso o usuário digite 0, libere a estrutura alocada e encerre o programa.

Atividade Semipresencial

2. Crie uma função para preencher as 5 estruturas de bandas criadas. Faça isso com alocação dinâmica

3. Crie uma função para exibir todas as informações das bandas/estruturas.

4. Crie uma função que peça ao usuário um número de 1 até 5. Em seguida, seu programa deve exibir informações da banda cuja posição no seu ranking é a que foi solicitada pelo usuário.

Atividade Semipresencial

5. Crie uma função que peça ao usuário um tipo de música e exiba as bandas com esse tipo de música no seu ranking.

6. Crie uma função que peça o nome de uma banda ao usuário e diga se ela está entre suas bandas favoritas ou não.

7. Utilize ENUMs para a definição do gênero (tipo da música que toca):

Exemplo:
```
enum genero{
        rock=1,
        samba=2,
        reggae=3,....
};
```

Atividade Semipresencial

- Entrega: dia 11/09 pelo moodle.
- Realizar de forma INDIVIDUAL.

unipampa

Universidade Federal do Pampa

Dúvidas ?