



UNIVERSIDADE
FEDERAL DE
SERGIPE



R e g i s t r o s

PROGRAMAÇÃO IMPERATIVA



Situação Problema

Aplicação

Escrever programa em C para ler nome, notas e frequência dos 80 alunos do projeto “AprendaProgramação”, calcular a média destes e emitir em tela a situação final da turma. São três notas, e são aprovados todos os que ficam com média acima da média da turma e têm frequência acima de 65%. Deve ser exibido o nome de cada aluno, seguido das notas, média e da situação final: APROVADO ou REPROVADO.



Como manter os dados deste programa ?

Situação Problema

Aplicação

Escrever programa em C para ler nome, notas e frequência dos 80 alunos do projeto “AprendaProgramação”, calcular a média destes e emitir em tela a situação final da turma. São três notas, e são aprovados todos os que ficam com média acima da média da turma e têm frequência acima de 65%. Deve ser exibido o nome de cada aluno, seguido das notas, média e da situação final: APROVADO ou REPROVADO.

Para manter os dados de um aluno, é preciso dispor de uma variável para armazenar nome (string), nota1 (float), nota2 (float), nota3 (float) frequência (int) e possivelmente: média (float) e situação final (string ou "boolean"). Nesse caso, são úteis os registros em C, as structs.

Registro / Struct

Definição & Sintaxe

Os registros (structs em C) correspondem a estruturas heterogêneas de armazenamento de dados.

Através desses tem-se novos tipos.

```
struct <nomeDoTipo>
{
    <tipo1> <campo1>[, ...<campo2>];
    <tipo2> <campo3>[...];    [...]
} <IdentificaVariavel>[...];
```

Registro / Struct

Sintaxe

Onde:

palavra reservada

nome do tipo registro

```
struct <nomeDoTipo>
```

tipos dos
campos

```
{
```

campos que compõem registros

```
<tipo1> <campo1>[, ...<campo2>];
```

```
<tipo2> <campo3>[...];    [...]
```

```
} <IdentificaVariavel>[...];
```

variável do tipo registro

Registro / Struct

Aplicação

Uma oficina mecânica que deseja registrar diariamente os clientes atendidos.

```
struct TpCliente{  
    char Nome[21];  
    char Telefone[12];  
    float Valor;  
    char Mecanico[21];  
    int Tempo; //de atendimento, em s  
} Cliente;
```

Nome	Zé
Telefone	7932574455
Valor	170,00
Mecanico	Beto
Tempo	75000

Cliente

Registro / Struct

Declaração

```
struct TpCliente{
    char Nome[21];
    char Telefone[12];
    float Valor;
    char Mecanico[21];
    int Tempo; //de atendimento, em s
} Cliente;
```

Nome	Zé
Telefone	7932574455
Valor	170,00
Mecanico	Beto
Tempo	75000

C1

Nome	Zé
Telefone	7932574455
Valor	170,00
Mecanico	Beto
Tempo	75000

C2

Outra forma de
declarar variáveis
do tipo struct:

```
struct TpCliente{
    char Nome[21];
    char Telefone[12];
    float Valor;
    char Mecanico[21];
    int Tempo;}; //de atendimento, em s
struct TpCliente C1, C2;
```

Registro / Struct

Aplicação & Sintaxe

Onde:

palavra reservada

nome do tipo registro

tipos dos campos

campos que compoem registros

variáveis do tipo registro

```
struct TpCliente  
{
```

```
char Nome[21];  
char Telefone[12];  
float Valor;
```

```
char Mecanico[21];  
int Tempo;}; //de atendimento em s
```

```
struct TpCliente C1, C2;
```


Registro / Struct

Outra Aplicação

A secagem é um dos processos mais antigos utilizados pelo homem na conservação de alimentos, para tanto podem ser usados equipamentos com controle da temperatura, umidade e velocidade do ar. Para compor relatório de monitoramento da secagem

```
char Fruta[21];  
struct TpSecagem{  
    float Temperatura;  
    int Umidade;  
    int Hora;  
    int Minuto;  
    char Antioxidante[21];  
}RegSecagem;
```

Temperatura	104.9
Umidade	50
Hora	14
Minuto	15
Antioxidante	Vitamina C

RegSecagem

Registro / Struct

Aplicação

Temperatura	104.9
Umidade	50
Hora	14
Minuto	15
Antioxidante	Vitamina C

RegSecagem

```
struct TpSecagem{  
    float Temperatura;  
    int Umidade;  
    int Hora;  
    int Minuto;  
    char Antioxidante[21];  
}RegSecagem;
```



*Qual a variável
dessa declaração?*

Registro / Struct

Aplicação

Temperatura	104.9
Umidade	50
Hora	14
Minuto	15
Antioxidante	Vitamina C

RegSecagem

```
struct TpSecagem{  
    float Temperatura;  
    int Umidade;  
    int Hora;  
    int Minuto;  
    char Antioxidante[21];  
}RegSecagem;
```



*Qual o tipo da
variável?*

Registro / Struct

Aplicação

Temperatura	104.9
Umidade	50
Hora	14
Minuto	15
Antioxidante	Vitamina C

RegSecagem

```
struct TpSecagem{  
    float Temperatura;  
    int Umidade;  
    int Hora;  
    int Minuto;  
    char Antioxidante[21];  
}RegSecagem;
```

*Que ajustes efetuar para se ter duas
variáveis: RegSecagem1 e
RegSecagem2?*



Registro / Struct

Aplicação

Temperatura	104.9
Umidade	50
Hora	14
Minuto	15
Antioxidante	Vitamina C

RegSecagem

```
struct TpSecagem{  
    float Temperatura;  
    int Umidade;  
    int Hora;  
    int Minuto;  
    char Antioxidante[21];  
}RegSecagem;
```



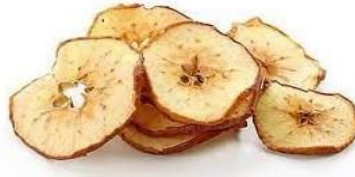
Analisando a declaração dada, o que evidencia de se tratam de variáveis heterogêneas?

Registro / Struct

Aplicação

```
struct TpCliente{  
    char Nome[21];  
    char Telefone[12];  
    float Valor;  
    char Mecanico[21];  
    int Tempo;}; //de atendimento, em s  
struct TpCliente C1, C2;
```

```
struct TpSecagem{  
    float Temperatura;  
    int Umidade;  
    int Hora;  
    int Minuto;  
    char Antioxidante[21];  
}RegSecagem;
```



*Através desses
observa-se que por
meio de struct são
propostos novos
tipos?*



Registro / Struct

Aplicação

Considerando o exemplo, temos:

```
struct TpSecagem{  
    float Temperatura;  
    int Umidade;  
    int Hora;  
    int Minuto;  
    char Antioxidante[21];  
}RegSecagem;
```

Temperatura	104.9
Umidade	50
Hora	14
Minuto	15
Antioxidante	Vitamina C

RegSecagem

Neste temos uma variável `RegSecagem`, do tipo `TpSecagem`, composta pelos campos: Temperatura, Umidade, Hora, Minuto e Antioxidante.

Registro / Struct

Manipulação/Operações

Temperatura	104.9
Umidade	50
Hora	14
Minuto	15
Antioxidante	Vitamina C

RegSecagem

```
struct TpSecagem{  
    float Temperatura;  
    int Umidade;  
    int Hora;  
    int Minuto;  
    char Antioxidante[21];  
}RegSecagem;
```

Exceto a atribuição de registros de mesmo tipo, os registros são manipulados campo a campo. Para tanto deve ser usado o nome do registro seguido de ponto e do nome do campo a manipular.

Exemplos:

```
RegSecagem.Minuto = 45;  
puts (RegSecagem.Antioxidante);
```


Registro / Struct

Manipulação/Operações

Temperatura	104.9
Umidade	50
Hora	14
Minuto	15
Antioxidante	Vitamina C

RegSecagem

A manipulação dos campos é efetuada em conformidade com os tipos desses.

Exemplos:

```
RegSecagem.Minuto = 45;
```

```
puts (RegSecagem.Antioxidante) ;
```

```
struct TpSecagem{  
    float Temperatura;  
    int Umidade;  
    int Hora;  
    int Minuto;  
    char Antioxidante[21];  
}RegSecagem;
```

```
if (RegSecagem.Umidade==35)  
...
```

```
RegSecagem.Temperatura++;
```

Registro / Struct

Manipulação/Operações

Temperatura	104.9
Umidade	50
Hora	14
Minuto	15
Antioxidante	Vitamina C

RegSecagem

A manipulação dos campos é efetuada em conformidade com os tipos desses.

Exemplos:

```
RegSecagem.Minuto = 45;
```

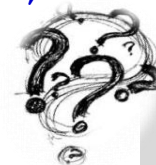
```
puts (RegSecagem.Antioxidante);
```

```
struct TpSecagem{  
    float Temperatura;  
    int Umidade;  
    int Hora;  
    int Minuto;  
    char Antioxidante[21];  
}RegSecagem;
```

```
if (RegSecagem.Umidade==35)
```

```
...
```

```
RegSecagem.Temperatura++;
```



*Como efetuar
a leitura do
campo hora?*

Registro / Struct

Inicialização

```
struct TpSecagem{  
    float Temperatura;  
    int Umidade;  
    int Hora;  
    int Minuto;  
    char Antioxidante[21];  
};
```

Temperatura	104.9
Umidade	50
Hora	14
Minuto	15
Antioxidante	Vitamina C

RegSecagem

Para inicializar:

```
struct TpSecagem RegSecagem =  
    {104.9,  
      50,  
      14,  
      15,  
      "Vitamina C"};
```

Registro / Struct

Inicialização

```
struct TpSecagem{  
    float Temperatura;  
    int Umidade;  
    int Hora;  
    int Minuto;  
    char Antioxidante[21];  
};
```

Temperatura	104.9
Umidade	50
Hora	14
Minuto	15
Antioxidante	Vitamina C

RegSecagem

Para inicializar:

```
struct TpSecagem RegSecagem =  
    {104.9,  
      50,  
      14,  
      15,  
      "Vitamina C"};
```



*A partir desta inicialização,
qual o valor mantido no
campo Antioxidante?*

Registro / Struct

Entrada de Dados

```
struct TpSecagem{  
    float Temperatura;  
    int Umidade;  
    int Hora;  
    int Minuto;  
    char Antioxidente[21];  
}RegSecagem;
```

Temperatura	104.9
Umidade	50
Hora	14
Minuto	15
Antioxidente	Vitamina C

RegSecagem

... LEITURA DOS CAMPOS

```
printf("Temperatura: ");  
scanf("%f", &RegSecagem.Temperatura);  
printf("Umidade: ");  
scanf("%d", &RegSecagem.Umidade);  
printf("Hora: ");  
scanf("%d", &RegSecagem.Hora);  
printf("Minuto: ");  
scanf("%d", &RegSecagem.Minuto);  
printf("Antioxidente: ");  
gets(RegSecagem.Antioxidente);
```

...

Registro / Struct

Saída de Dados

EXIBIÇÃO DOS CAMPOS

...
printf("Temperatura: %.1f", RegSecagem.Temperatura);

printf("Umidade: %d", RegSecagem.Umidade);

printf("Hora: %d", RegSecagem.Hora);

printf("Minuto: %d", RegSecagem.Minuto);

printf("Antioxidante: %s", RegSecagem.Antioxidante);

...

```
struct TpSecagem{  
    float Temperatura;  
    int Umidade;  
    int Hora;  
    int Minuto;  
    char Antioxidante[21];  
}RegSecagem;
```

Temperatura	104.9
Umidade	50
Hora	14
Minuto	15
Antioxidante	Vitamina C

RegSecagem

Registro / Struct

Aplicação

campo1	Valor1
campo2	Valor2
...	
campoN	ValorN

Variável

```
struct <nomeDoTipo>
{
    <tipo1> <campo1>[,...<campo2>];
    <tipo2> <campo3>[...];    [...]
} <NomeVariavel>[...];
```



*Quando aplicar
registros?*

Registro / Struct

Aplicação

Escrever programa em C para ler nome, notas e frequência dos 80 alunos do projeto

AprendaProgramação, calcular a média destes e emitir em tela a situação final da turma. São três notas, e são aprovados todos os que ficam com média acima da média da turma e têm frequência acima de 65%. Deve ser exibido o nome de cada aluno, seguido das notas, média e da situação final: APROVADO ou REPROVADO.

*Na resolução deste
pode/deve ser
aplicado registro?*



Registro / Struct

Aplicação

Escrever programa em C para ler nome, notas e frequência dos 80 alunos do projeto

AprendaProgramação, calcular a média destes e emitir em tela a situação final da turma. São três notas, e são aprovados todos os que ficam com média acima da média da turma e têm frequência acima de 65%. Deve ser exibido o nome de cada aluno, seguido das notas, média e da situação final: APROVADO ou REPROVADO.



Uma única variável do tipo registro é suficiente para o desenvolvimento deste programa?

Registro / Struct

Vetor de Registros

Nome	Belarmino
Nota 1	7.0
Nota 2	6.0
Nota 3	8.0
Frequência	50

```
struct TpAluno Vetor[80];  
struct TpAluno RegAluno;
```

```
typedef struct TpAluno TpSt;  
  
TpSt Vetor[80];  
TpSt RegAluno;
```

```
struct TpAluno{  
    char Nome[21];  
    float Nota1;  
    float Nota2;  
    float Nota3;  
    int Frequencia;};
```



*Quantos slots/células
tem o array Vetor?*

Registro / Struct

Vetor de Registros

Nome	Belarmino
Nota 1	7.0
Nota 2	6.0
Nota 3	8.0
Frequência	50

```
struct TpAluno Vetor[80];  
struct TpAluno RegAluno;
```

```
typedef struct TpAluno TpSt;  
  
TpSt Vetor[80];  
TpSt RegAluno;
```

```
struct TpAluno{  
    char Nome[21];  
    float Nota1;  
    float Nota2;  
    float Nota3;  
    int Frequencia;};
```



*Que dado(s) é/são
mantido(s) em cada
slot do Vetor?*

Registro / Struct

Vetor de Registros

Nome	Belarmino
Nota 1	7.0
Nota 2	6.0
Nota 3	8.0
Frequência	50

```
struct TpAluno Vetor[80];  
struct TpAluno RegAluno;
```

```
typedef struct TpAluno TpSt;  
  
TpSt Vetor[80];  
TpSt RegAluno;
```

```
struct TpAluno{  
    char Nome[21];  
    float Nota1;  
    float Nota2;  
    float Nota3;  
    int Frequencia;};
```



*Convém manter média e status
(aprovado ou reprovado)
como campos do vetor?*

```
struct TpAluno
```

```
{
```

```
    char Nome[21];
```

```
    float Nota1;
```

```
    float Nota2;
```

```
    float Nota3;
```

```
    int Frequencia;
```

```
} RgAluno;
```

Registro / Struct

Vetor de Registros

```
typedef struct TpAluno TpSt;  
TpSt Vetor[80];
```

0

1

2

3

...

79

Nome	Zé
Nota 1	8.5
Nota 2	6.0
Nota 3	7.0
Frequência	58

Nome	Bia
Nota 1	9.5
Nota 2	4.0
Nota 3	5.0
Frequência	40

Nome	Gil
Nota 1	3.5
Nota 2	0.0
Nota 3	0.0
Frequência	15

Nome	Gal
Nota 1	8.5
Nota 2	8.0
Nota 3	8.0
Frequência	50

Nome	Bel
Nota 1	7.5
Nota 2	8.0
Nota 3	9.0
Frequência	55



Com qual código C é possível ter acesso a 1 slot desse vetor?

```
struct TpAluno
```

```
{
```

```
    char Nome[21];
```

```
    float Nota1;
```

```
    float Nota2;
```

```
    float Nota3;
```

```
    int Frequencia;
```

```
} RgAluno;
```

Registro / Struct

Vetor de Registros

```
typedef struct TpAluno TpSt;  
TpSt Turma[80];
```

0

1

2

3

...

79

Nome	Zé
Nota 1	8.5
Nota 2	6.0
Nota 3	7.0
Frequência	58

Nome	Bia
Nota 1	9.5
Nota 2	4.0
Nota 3	5.0
Frequência	40

Nome	Gil
Nota 1	3.5
Nota 2	0.0
Nota 3	0.0
Frequência	15

Nome	Gal
Nota 1	8.5
Nota 2	8.0
Nota 3	8.0
Frequência	50

Nome	Bel
Nota 1	7.5
Nota 2	8.0
Nota 3	9.0
Frequência	55

```
strcpy(Turma[2].Nome, "Gildete");
```

```
Media=(Turma[2].Nota1+Turma[2].Nota2+Turma[2].Nota3)/3;
```

```
struct TpAluno
```

```
{
```

```
    char Nome[21];
```

```
    float Nota1;
```

```
    float Nota2;
```

```
    float Nota3;
```

```
    int Frequencia;
```

```
} RgAluno;
```

Registro / Struct

Vetor de Registros

```
typedef struct TpAluno TpSt;  
TpSt Turma[80];
```

0

1

2

3

...

79

Nome	Zé
Nota 1	8.5
Nota 2	6.0
Nota 3	7.0
Frequência	58

Nome	Bia
Nota 1	9.5
Nota 2	4.0
Nota 3	5.0
Frequência	40

Nome	Gil
Nota 1	3.5
Nota 2	0.0
Nota 3	0.0
Frequência	15

Nome	Gal
Nota 1	8.5
Nota 2	8.0
Nota 3	8.0
Frequência	50

Nome	Bel
Nota 1	7.5
Nota 2	8.0
Nota 3	9.0
Frequência	55



Considerando a necessidade de busca de um aluno, pelo nome (chave única)?

Registro / Struct

EXERCÍCIO 01

Escrever programa em C para ler nome, 3 notas e frequência dos 80 alunos do projeto

AprendaProgramação, calcular a média destes (média da turma) e emitir em tela a situação final de cada aluno.

São aprovados todos os que ficam com média (aritmética simples) acima da média da turma e têm frequência acima de 65%. Deve ser exibido o nome de cada aluno, seguido das notas, média e da situação final: APROVADO ou REPROVADO.



UNIVERSIDADE
FEDERAL DE
SERGIPE



B u s c a

PROGRAMAÇÃO IMPERATIVA



```
struct TpAluno
```

```
{  
    char Nome[21];  
    float Nota1;  
    float Nota2;  
    float Nota3;  
    int Frequencia;  
} RgAluno;
```

Registro / Struct

Vetor de Registros: BUSCA

```
typedef struct TpAluno TpSt;  
TpSt Turma[80];
```

0

1

2

3

...

79

Nome	Zé
Nota 1	8.5
Nota 2	6.0
Nota 3	7.0
Frequência	58

Nome	Bia
Nota 1	9.5
Nota 2	4.0
Nota 3	5.0
Frequência	40

Nome	Gil
Nota 1	3.5
Nota 2	0.0
Nota 3	0.0
Frequência	15

Nome	Gal
Nota 1	8.5
Nota 2	8.0
Nota 3	8.0
Frequência	50

Nome	Bel
Nota 1	7.5
Nota 2	8.0
Nota 3	9.0
Frequência	55



O que é busca?

Como efetuar?

Quando é aplicada?

```
struct TpAluno
```

```
{  
    char Nome[21];  
    float Nota1;  
    float Nota2;  
    float Nota3;  
    int Frequencia;  
} RgAluno;
```

Registro / Struct

Vetor de Registros: BUSCA

```
typedef struct TpAluno TpSt;  
TpSt Turma[80];
```

0

1

2

3

...

79

Nome	Zé
Nota 1	8.5
Nota 2	6.0
Nota 3	7.0
Frequência	58

Nome	Bia
Nota 1	9.5
Nota 2	4.0
Nota 3	5.0
Frequência	40

Nome	Gil
Nota 1	3.5
Nota 2	0.0
Nota 3	0.0
Frequência	15

Nome	Gal
Nota 1	8.5
Nota 2	8.0
Nota 3	8.0
Frequência	50

Nome	Bel
Nota 1	7.5
Nota 2	8.0
Nota 3	9.0
Frequência	55

A **BUSCA SEQUENCIAL** por um item E, tem início no primeiro elemento de um grupo de dados (vetor) e segue em sequência, elemento a elemento, verificando se cada item do conjunto (de dados) é igual a E - busca com êxito; ou até que todos os dados sejam percorridos e E não seja encontrado – busca sem sucesso.

```
struct TpAluno
```

```
{  
    char Nome[21];  
    float Nota1;  
    float Nota2;  
    float Nota3;  
    int Frequencia;  
} RgAluno;
```

Registro / Struct

Vetor de Registros: BUSCA

```
typedef struct TpAluno TpSt;  
TpSt Turma[80];
```

0

1

2

3

...

79

Nome	Zé
Nota 1	8.5
Nota 2	6.0
Nota 3	7.0
Frequência	58

Nome	Bia
Nota 1	9.5
Nota 2	4.0
Nota 3	5.0
Frequência	40

Nome	Gil
Nota 1	3.5
Nota 2	0.0
Nota 3	0.0
Frequência	15

Nome	Gal
Nota 1	8.5
Nota 2	8.0
Nota 3	8.0
Frequência	50

Nome	Bel
Nota 1	7.5
Nota 2	8.0
Nota 3	9.0
Frequência	55

A **BUSCA SEQUENCIAL** é uma operação clássica, aplicada quando, a partir de uma chave (dada, conhecida) deseja-se obter os outros dados relativos à chave (desconhecidos). É a forma trivial/básica de se efetuar uma busca.

```
struct TpAluno
```

```
{  
    char Nome[21];  
    float Nota1;  
    float Nota2;  
    float Nota3;  
    int Frequencia;  
} RgAluno;
```

Registro / Struct

Vetor de Registros: BUSCA

```
typedef struct TpAluno TpSt;  
TpSt Turma[80];
```

0

1

2

3

...

79

Nome	Zé
Nota 1	8.5
Nota 2	6.0
Nota 3	7.0
Frequência	58

Nome	Bia
Nota 1	9.5
Nota 2	4.0
Nota 3	5.0
Frequência	40

Nome	Gil
Nota 1	3.5
Nota 2	0.0
Nota 3	0.0
Frequência	15

Nome	Gal
Nota 1	8.5
Nota 2	8.0
Nota 3	8.0
Frequência	50

Nome	Bel
Nota 1	7.5
Nota 2	8.0
Nota 3	9.0
Frequência	55



*Conhecem outro tipo de busca
além da sequencial?*

```
struct TpAluno
```

```
{  
    char Nome[21];  
    float Nota1;  
    float Nota2;  
    float Nota3;  
    int Frequencia;  
} RgAluno;
```

Registro / Struct

Vetor de Registros: BUSCA

```
typedef struct TpAluno TpSt;  
TpSt Turma[80];
```

0

1

2

3

...

79

Nome	Zé
Nota 1	8.5
Nota 2	6.0
Nota 3	7.0
Frequência	58

Nome	Bia
Nota 1	9.5
Nota 2	4.0
Nota 3	5.0
Frequência	40

Nome	Gil
Nota 1	3.5
Nota 2	0.0
Nota 3	0.0
Frequência	15

Nome	Gal
Nota 1	8.5
Nota 2	8.0
Nota 3	8.0
Frequência	50

Nome	Bel
Nota 1	7.5
Nota 2	8.0
Nota 3	9.0
Frequência	55



Conhecem a busca direta?

Registro / Struct

BUSCA BINÁRIA

Vetor onde se tem o número do corredor (atleta) e o tempo gasto na corrida.

03	12	24	36	42	55	62	78
25	35	18	88	40	61	55	68

03	12	24	36	42	55	62	78
25	35	18	88	40	61	55	68

03	12	24	36	42	55	62	78
25	35	18	88	40	61	55	68

Além da busca sequencial, há a **busca ou pesquisa binária**, a qual parte do pressuposto de que o vetor está ordenado e realiza sucessivas divisões do espaço de busca comparando o elemento buscado (chave) com o elemento no meio do vetor. Se o elemento do meio do vetor for a chave, a busca termina com sucesso. Caso contrário, se o elemento do meio vier antes do elemento buscado, então a busca continua na metade posterior do vetor. E finalmente, se o elemento do meio vier depois da chave, a busca continua na metade anterior do vetor.

Registro / Struct

BUSCA BINÁRIA

03	12	24	36	42	55	62	78
25	35	18	88	40	61	55	68

03	12	24	36	42	55	62	78
25	35	18	88	40	61	55	68

03	12	24	36	42	55	62	78
25	35	18	88	40	61	55	68



Considerando um vetor onde se tem o número de um corredor (atleta) e o tempo gasto na corrida. Para buscar o atleta com inscrição 42?

Além da busca sequencial, há a busca ou pesquisa binária, a qual **parte do pressuposto de que o vetor está ordenado** e realiza sucessivas divisões do espaço de busca comparando o elemento buscado (chave) com o elemento no meio do vetor. Se o elemento do meio do vetor for a chave, a busca termina com sucesso. Caso contrário, se o elemento do meio vier antes do elemento buscado, então a busca continua na metade posterior do vetor. E finalmente, se o elemento do meio vier depois da chave, a busca continua na metade anterior do vetor.

Registro / Struct

BUSCA BINÁRIA

03	12	24	36	42	55	62	78
25	35	18	88	40	61	55	68

03	12	24	36	42	55	62	78
25	35	18	88	40	61	55	68

03	12	24	36	42	55	62	78
25	35	18	88	40	61	55	68

03	12	24	36	42	55	62	78
25	35	18	88	40	61	55	68

Considerando um vetor onde se tem o número de um corredor (atleta) e o tempo gasto na corrida. Para buscar o atleta com inscrição 78?



Além da busca sequencial, há a busca ou pesquisa binária, a qual **parte do pressuposto de que o vetor está ordenado** e realiza sucessivas divisões do espaço de busca comparando o elemento buscado (chave) com o elemento no meio do vetor. Se o elemento do meio do vetor for a chave, a busca termina com sucesso. Caso contrário, se o elemento do meio vier antes do elemento buscado, então a busca continua na metade posterior do vetor. E finalmente, se o elemento do meio vier depois da chave, a busca continua na metade anterior do vetor.

Registro / Struct

BUSCA BINÁRIA

03	12	24	36	42	55	62	78
25	35	18	88	40	61	55	68

03	12	24	36	42	55	62	78
25	35	18	88	40	61	55	68

03	12	24	36	42	55	62	78
25	35	18	88	40	61	55	68

03	12	24	36	42	55	62	78
25	35	18	88	40	61	55	68

BUSCA SEM ÊXITO

Considerando um vetor onde se tem o número de um corredor (atleta) e o tempo gasto na corrida. Para buscar o atleta com inscrição 30?

Além da busca sequencial, há a busca ou pesquisa binária, a qual **parte do pressuposto de que o vetor está ordenado** e realiza sucessivas divisões do espaço de busca comparando o elemento buscado (chave) com o elemento no meio do vetor. Se o elemento do meio do vetor for a chave, a busca termina com sucesso. Caso contrário, se o elemento do meio vier antes do elemento buscado, então a busca continua na metade posterior do vetor. E finalmente, se o elemento do meio vier depois da chave, a busca continua na metade anterior do vetor.



Registro / Struct

BUSCA BINÁRIA

03	12	24	36	42	55	62	78
25	35	18	88	40	61	55	68

03	12	24	36	42	55	62	78
25	35	18	88	40	61	55	68

03	12	24	36	42	55	62	78
25	35	18	88	40	61	55	68



Por que é necessário aplicar sobre base de dados ordenada?

Além da busca sequencial, há a busca ou pesquisa binária, a qual **parte do pressuposto de que o vetor está ordenado** e realiza sucessivas divisões do espaço de busca comparando o elemento buscado (chave) com o elemento no meio do vetor. Se o elemento do meio do vetor for a chave, a busca termina com sucesso. Caso contrário, se o elemento do meio vier antes do elemento buscado, então a busca continua na metade posterior do vetor. E finalmente, se o elemento do meio vier depois da chave, a busca continua na metade anterior do vetor.

Registro / Struct

BUSCA BINÁRIA

03	12	24	36	42	55	62	78
25	35	18	88	40	61	55	68

03	12	24	36	42	55	62	78
25	35	18	88	40	61	55	68

03	12	24	36	42	55	62	78
25	35	18	88	40	61	55	68



Nesse poderia ser aplicada a busca direta?

Além da busca sequencial, há a busca ou pesquisa binária, a qual **parte do pressuposto de que o vetor está ordenado** e realiza sucessivas divisões do espaço de busca comparando o elemento buscado (chave) com o elemento no meio do vetor. Se o elemento do meio do vetor for a chave, a busca termina com sucesso. Caso contrário, se o elemento do meio vier antes do elemento buscado, então a busca continua na metade posterior do vetor. E finalmente, se o elemento do meio vier depois da chave, a busca continua na metade anterior do vetor.

Registro / Struct

BUSCA BINÁRIA

```
int bbinaria (int chave, int inicio, int fim, int vetor[30]) {  
    //início e fim são os índices inicial e final do array  
    if (fim < inicio)  
        return -1;  
    else {  
        int meio = (inicio + fim)/2;  
        if (vetor[meio]==chave)  
            return meio;  
        else{  
            if (vetor[meio] < chave)  
                return bbinaria (chave, meio+1, fim, vetor);  
            else  
                return bbinaria (chave, inicio, meio-1, vetor);  
        }  
    }  
}
```

Aplicar sobre uma base de dados.
Observar a aplicação da recursão.

Busca Binária

EXERCÍCIO 02

Ajustar o subprograma considerando que o vetor (de inteiros) está ordenado de forma decrescente.

```
int bbinaria (int chave, int inicio, int fim, int vetor[30]) {  
    //início e fim são os índices inicial e final do array  
    if (fim < inicio)  
        return -1;  
    else {  
        int meio = (inicio + fim)/2;  
        if (vetor[meio]==chave)  
            return meio;  
        else{  
            if (vetor[meio] < chave)  
                return bbinaria (chave, meio+1, fim, vetor);  
            else  
                return bbinaria (chave, inicio, meio-1, vetor);  
        }  
    }  
}
```

Busca Binária

EXERCÍCIO 03

Ajustar o subprograma considerando que o vetor deve ser composto por strings:

```
int bbinaria (int chave, int inicio, int fim, int vetor[30]) {  
    if (fim < inicio)  
        return -1;  
    else {  
        int meio = (inicio + fim)/2;  
        if (vetor[meio]==chave)  
            return meio;  
        else{  
            if (vetor[meio] < chave)  
                return bbinaria (chave, meio+1, fim, vetor);  
            else  
                return bbinaria (chave, inicio, meio-1, vetor);  
        }  
    }  
}
```

Registro / Struct

BUSCA BINÁRIA

```
int bbinaria (int chave, int inicio, int fim, int vetor[30]) {  
    if (fim < inicio)  
        return -1;  
    else {  
        int meio = (inicio + fim)/2;  
        if (vetor[meio]==chave)  
            return meio;  
        else{  
            if (vetor[meio] < chave)  
                return bbinaria (chave, meio+1, fim, vetor);  
            else  
                return bbinaria (chave, inicio, meio-1, vetor);  
        }  
    }  
}
```



*Reconhecem o recurso utilizado por meio
do qual a função chama a si mesma?*



UNIVERSIDADE
FEDERAL DE
SERGIPE



Recursividade

PROGRAMAÇÃO IMPERATIVA



RECURSIVIDADE

Definição

A **recursividade**, ou **recursão**, corresponde a quando um módulo invoca/chama a si mesmo.

O módulo onde ocorre a recursividade, é chamado **recursivo**.



RECURSIVIDADE

Definição & Aplicação

```
int bbinaria (int chave, int inicio, int fim, int vetor[30]) {  
    if (fim < inicio)  
        return -1;  
    else {  
        int meio = (inicio + fim)/2;  
        if (vetor[meio]==chave)  
            return meio;  
        else{  
            if (vetor[meio] < chave)  
                return bbinaria (chave, meio+1, fim, vetor);  
            else  
                return bbinaria (chave, inicio, meio-1, vetor);  
        }  
    }  
}
```

A **recursividade**, ou **recursão**, corresponde à quando um módulo invoca/chama a si mesmo.

RECURSIVIDADE

Aplicação

```
int fatorial(int x) {  
    int i, aux;  
    aux=1;  
    for (i=1; i<=x; i++)  
        aux=aux * i;  
    return aux;  
}
```

O fatorial de um número é a multiplicação desse número por todos os seus antecessores maiores que zero.

RECURSIVIDADE

Aplicação

```
int fatorial (int N)
{
    if ( (N==0) || (N==1) )
        return 1;
    else
        return (N * fatorial (N-1) );
}
```

Considerando o conceito de recursão,
o fatorial de n pode ser entendido como:

$$\text{fatorial}(n) = n \cdot \text{fatorial}(n-1)$$

RECURSIVIDADE

Legibilidade

Solução Iterativa

```
int fatorial(int x) {  
    int i, aux;  
    aux=1;  
    for (i=1; i<=x; i++)  
        aux=aux * i;  
    return aux;  
}
```



Solução Recursiva

```
int fatorial(int N) {  
    if ((N==0) || (N==1))  
        return 1;  
    else  
        return (N * fatorial(N-1));  
}
```

RECURSIVIDADE

Exemplo

```
int fatorial(int N)
{
    if ( (N==0) || (N==1) )
        return 1;
    else
        return (N * fatorial(N-1));
}
```



Calcular: `R = fatorial(5);`

RECURSIVIDADE

Funcionamento

- Rastreando: `R = fatorial(5);`
- `fatorial(5) = 5 * fatorial(4);` neste instante a execução da primeira chamada da função é interrompida temporariamente, as informações referentes a esta são armazenadas e é iniciada a execução de `fatorial(4)`.
- `fatorial(4) = 4 * fatorial(3);` outra vez, a execução da chamada da função é interrompida temporariamente, as informações referentes a esta são armazenadas e é iniciada a execução de `fatorial(3)`...

RECURSIVIDADE

Aplicação

$$5! = 5 \cdot \underbrace{4 \cdot 3 \cdot 2 \cdot 1}_{4!}$$

$$4! = 4 \cdot \underbrace{3 \cdot 2 \cdot 1}_{3!}$$

$$3! = 3 \cdot \underbrace{2 \cdot 1}_{2!}$$

...



Podemos dizer que

`fatorial(n) = n * fatorial(n-1)`?

Então não para?

RECURSIVIDADE

Definição & Aplicação

```
...
int fatorial(int N){
    int R=1;
    int i;
    for (i=1; i<=N; i++)
        R=R*i;
    return R;
}

int main()
{
    int N, K;
    printf(">>> Cálculo de N!/(K!* N-K)! <<<\n\n");
    printf("Digite os valores de N e K? ");
    scanf("%d%d",&N,&K);
    if (N<0) || (K<0)
        printf("ERRO: Não é possível calcular N!/(K!* N-K)!.);");
    else{
        int R;
        R=fatorial(N)/(fatorial(K)*fatorial(N-K));
        printf("Resultado: %d\n",R);}
    return 0;
}
```



Sem aplicar a função dada, como calcular R sendo este dado por: $R = N! / (K! * (N-K)!)$?

RECURSIVIDADE

Procedimento Recursivo

```
void inverter(int N)

/*Inverte a ordem dos algarismos de um
  número inteiro.*/
{
    if (N<10)
        printf("%d",N);
    else
    {
        printf("%d",N%10);
        inverter(N/10);
    }
}
```



Rastrear
Inverter(6754);

Recursividade

EXERCÍCIO 04

Ajustar o código de forma a identificar o menor valor do vetor.

```
int maior(int v[], int n) {  
    // n é o número de elementos do vetor  
    if (n == 1) return v[0];  
    else {  
        int x;  
        x = maior(v, n-1);  
        if (x > v[n-1]) return x;  
        else return v[n-1];  
    }  
}
```

```
int main(){  
    int Vetor[20];  
    printf("Tamanho do vetor? "); int T;  
    scanf("%d",&T);  
    for (int cont=0; cont<=T-1; cont++){  
        printf("Qual o %do dado do vetor? ",cont+1);  
        scanf("%d",&Vetor[cont]);  
    }  
    printf("\nO maior eh: %d.\n",maior(Vetor,T));  
    return 0;}
```

```
struct TpAluno
```

```
{
```

```
    char Nome[21];
```

```
    float Nota1;
```

```
    float Nota2;
```

```
    float Nota3;
```

```
    int Frequencia;
```

```
} RgAluno;
```

Registro / Struct

ORDENAÇÃO

```
typedef struct TpAluno TpSt;  
TpSt Turma[80];
```

0

1

2

3

...

79

Nome	Zé
Nota 1	8.5
Nota 2	6.0
Nota 3	7.0
Frequência	58

Nome	Bia
Nota 1	9.5
Nota 2	4.0
Nota 3	5.0
Frequência	40

Nome	Gil
Nota 1	3.5
Nota 2	0.0
Nota 3	0.0
Frequência	15

Nome	Gal
Nota 1	8.5
Nota 2	8.0
Nota 3	8.0
Frequência	50

Nome	Bel
Nota 1	7.5
Nota 2	8.0
Nota 3	9.0
Frequência	55



Considerando a necessidade emissão de um relatório com dados ordenados pelo nome do aluno. Como resolver?



UNIVERSIDADE
FEDERAL DE
SERGIPE



Ordenação Classificação





PROGRAMAÇÃO IMPERATIVA



Registro / Struct

Vetor de Registros: ORDENAÇÃO / CLASSIFICAÇÃO





A ordenação por **seleção** (do inglês, *selection sort*) é um algoritmo de ordenação baseado em se passar sempre o menor valor do vetor para a primeira posição (ou o maior dependendo da ordem requerida), depois o de segundo menor valor para a segunda posição, e assim é feito sucessivamente com os $n-1$ (sendo n o tamanho do vetor) elementos restantes, até os últimos dois elementos.

12	45	32	08	23	43	23	74	11	36
				<i>menor dado</i>					
08	45	32	12	23	43	23	74	11	36
				<i>menor dado</i>					
08	11	32	12	23	43	23	74	45	36
08	11	12	32	23	43	23	74	45	36
08	11	12	23	32	43	23	74	45	36
08	11	12	23	23	43	32	74	45	36
...									
08	11	12	23	23	32	36	43	45	74

Registro / Struct

Vetor de Registros: ORDENAÇÃO / CLASSIFICAÇÃO

Tem-se um único vetor em diversos
fases (passos da ordenação)

12	45	32	08	23	43	23	74	11	36
				<i>menor dado</i>					
08	45	32	12	23	43	23	74	11	36
				<i>menor dado</i>					
08	11	32	12	23	43	23	74	45	36
08	11	12	32	23	43	23	74	45	36
08	11	12	23	32	43	23	74	45	36
08	11	12	23	23	43	32	74	45	36
...									
08	11	12	23	23	32	36	43	45	74

CLASSIFICAÇÃO DE DADOS

Método por Seleção

```
#include <stdio.h>

void selection_sort (int vetor[], int max) {
    int i, j, min, aux;
    for (i = 0; i < (max - 1); i++) {
        /* O mínimo é o inicialmente o primeiro */
        min = i;
        for (j = i+1; j < max; j++)
            /* Caso haja numero menor, faz-se a troca do índice mínimo */
            if (vetor[j] < vetor[min])
                min = j;

        /* Se o índice mínimo for diferente do índice do i-ésimo número
        não ordenado, faz-se a troca dos valores */
        if (i != min) {
            aux = vetor[i];
            vetor[i] = vetor[min];
            vetor[min] = aux;
        } //for i
        /* Imprime o vetor ordenado */
        for (i = 0; i < max; i++) {
            printf ("%d ", vetor[i]);
            printf ("\n");
        }
    }
}

main () {
    int max, i;
    /* Lê tamanho do vetor */
    scanf ("%d", &max);
    int vetor[max];
    /* Lê os algarismos do vetor */
    for (i = 0; i < max; i++) {
        scanf ("%d", &vetor[i]);
    }
    selection_sort (vetor, max);
}
```

Registro / Struct

ORDENAÇÃO: BUBBLE SORT

É um algoritmo básico para organizar uma sequência de números ou outros elementos na ordem correta. O método funciona examinando cada conjunto de elementos adjacentes num vetor, da esquerda para a direita, trocando suas posições se estiverem fora de ordem. O algoritmo então repete esse processo até que possa percorrer toda a string e não encontrar dois elementos que precisem ser trocados.

[illegible]

CLASSIFICAÇÃO DE DADOS

Método BuBble Sort

```
#include <stdio.h>

void recurbublSort(int arr[], int len) {
    int temp;
    if (len == 1) {
        return;
    }
    for (int i=0; i<len-1; i++) {
        if (arr[i] > arr[i+1]) {
            temp=arr[i];
            arr[i]=arr[i+1];
            arr[i+1]=temp;
        }
    }
    len=len-1;
    recurbublSort(arr, len);
}

int main() {
    int Arr[] = {21, 34, 20, 31, 78, 43, 66};
    int length = sizeof(Arr)/sizeof(Arr[0]);
    recurbublSort(Arr, length);
    printf("Vetor Ordenado: ");
    for(int i=0;i<length;i++){
        printf("%d ",Arr[i]);
    }
    return 0;}
```

Recursividade

EXERCÍCIO 05

Pesquisar método de classificação de dados diferente do trabalhado em sala de aula, recursivo, e implementá-lo aplicando a vetor composto por 25 palavras (previamente definidas e desordenadas).



UNIVERSIDADE
FEDERAL DE
SERGIPE



Culminância

PROGRAMAÇÃO IMPERATIVA



Registros, Busca, Recursão...

EXERCÍCIO 06

Considerando a necessidade de construir um programa em C para definir o valor a pagar por uma vaga de estacionamento, por tempo de uso.

- As vagas são numeradas.
- O cliente escolhe a vaga em que deseja estacionar seu carro.
- É registrado no sistema a hora de chegada no estacionamento. Se um carro chegou às 8:00 no estacionamento e ocupa a vaga 15. Na posição 15 do vetor `struct` composta por Placa e hora.

Registros, Busca, Recursão...

EXERCÍCIO 07 / PROJETO

Na oficina autorizada das motos “Zonda” há box para atender serviços expressos, os quais correspondem a serviços que demandam até 60 minutos de dedicação da equipe de mecânicos para serem solucionados. Quando um cliente precisa de um serviço expresso, ele dirige-se ao atendimento, fornece: nome, modelo da moto, placa e descrição do defeito. Em seguida dirige-se a uma sala de espera e aguarda o conserto de sua moto.

Sabendo que o número máximo de solicitações não passa de 50, escrever programa modularizado para gerenciar os serviços expressos “Zonda”. Disponibilizar as opções:

(1) Solicitar Serviço – quando se insere os dados supracitados, e também os campos status com valor zero sinalizando que o serviço ainda não foi feito, preço, também iniciado com zero. Este deve estar em loop. (2) Iniciar Serviço – quando o status de uma dada moto (placa) é iniciado pelo mecânico e o status passa a ser um. (3) Remover Solicitação – para o caso do proprietário desistir de esperar, o que somente é permitido se o serviço não tiver iniciado, e o status passa a ser dois, dada a placa da moto. (4) Consultar Solicitações – quando se exibe todas as solicitações ainda não feitas (não iniciadas). (5) Concluir Serviço – para indicar que o mecânico concluiu o atendimento, define o preço final e o status passa a ser três. (6) Encerrar Expediente – para exibição de todos os serviços efetuados, e exibição do valor total obtido com os serviços. E (7) Sair – para encerrar a execução do programa.

Programação Imperativa

COMPLEMENTAR AULA...

Fundamentos da Programação de Computadores

Ana Fernanda Gomes Ascencio

Edilene Aparecida Veneruchi de Campos

Capítulos

Registros

Funções



Programação Imperativa

COMPLEMENTAR AULA...

Curso de Linguagem C
UFMG

*linux.ime.usp.br/~lucas
mmg/livecd/documenta
cao/documentos/curso_
de_c/www.ppgia.pucpr.
br/_maziero/ensino/so/
projetos/curso-c/c.html*

Aula 7 Funções (Recursão)

[Aula 1: Introdução e Sumário](#)

[Aula 2 - Primeiros Passos](#)

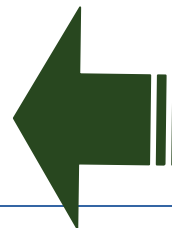
[Aula 3 - Variáveis, Constantes, Operadores e Expressões](#)

[Aula 4 - Estruturas de Controle de Fluxo](#)

[Aula 5 - Matrizes e Strings](#)

[Aula 6 - Ponteiros](#)

[Aula 7 - Funções](#)



Programação Imperativa

PRÓXIMO PASSO



Arquivos