

Prática de Programação



Estes *slides* são um resumo do professor para diversas aulas e não devem ser usados separadamente ou como fonte original.

Sugestões são bem-vindas: mande-as para vinicius@ufabc.edu.br

Material sob licenciamento Creative Commons Atribuição – Não Comercial – Sem Derivações 4.0 Internacional (https://creativecommons.org/licenses/by-nc-nd/4.0/deed.pt_BR).

Apresentação

Tópicos

- ▶ Dicas e sugestões para estudo e para produção.
- ▶ Quais linguagens utilizamos e porque preferimos C.
- ▶ Agradecimentos e referências.

2017 VCL

Dicas e sugestões gerais

Prefira digitar os códigos fornecidos, em vez de apenas copiar e colar, para melhor fixação, visualização de dúvidas e aprendizagem passo a passo.

Tenha lápis e papel para trabalhar em programação, não necessariamente para fazer fluxogramas, mas para estruturar dados, anotar porções de código a fazer e estudar pequenos exemplos.

Para programação, a escolha de um editor de texto é uma questão pessoal e de conforto.

Escolha uma linguagem de programação adequada para o seu objetivo: manipular objetos abstratos ou índices concretos; calcular numericamente com precisão ou trabalhar com textos ou operar arquivos etc.

Documente seu programa, para que outros possam aproveitar o código e você mesmo aperfeiçoá-lo daqui a uns anos, ou até amanhã, sem perder tempo e errar ao tentar descobrir o que cada variável armazena ou cada laço realiza.

Conheça e utilize os principais “maneirismos” dos programadores, especialmente aqueles da linguagem que você utiliza. Procedimentos corriqueiros, como selecionar o menor elemento de um vetor, são habitualmente programados com estruturas “manjadas” que não requerem documentação excessiva — somente identificação do que está sendo feito — e facilitam o entendimento e a performance do seu programa.

Por outro lado, não exagere a complexidade de um programa somente pelo prazer de usar uma ferramenta específica, o que tornaria o código ilegível.* Os compiladores modernos têm opções de otimização que ganham automaticamente a eficiência que uma tal ferramenta poderia obter.

2017 VCL

*Exemplo em C: `for(i=n=0; i<M && (a[i]<=0 || ++n); i++)` ; com o qual n conta quantos números estritamente positivos temos no vetor a de tamanho M.

Um algoritmo menos complexo pode ser de interesse em uma primeira implementação para estudo, testes e palestras, mesmo que não tenha eficiência suficiente para implementação real (isto é, não seja *industrial grade*). De fato, privilegie um algoritmo simples se ele precisa ser realizado manualmente!

Estude as eras anteriores ao computador: Alguns algoritmos extremamente eficientes têm sua origem na Grécia antiga e mesmo na Babilônia, junto com o nascimento da civilização. Boas idéias surgem da escassez de recursos e mão de obra.

Estude os programas cinquentões: Excelentes recursos sobre os quais muitos ótimos programadores já se debruçaram estão livres de erros e têm mais eficiência, de modo que não é preciso reinventar a roda.

Conheça a comunidade mundial de programação: Fóruns e tutoriais abundam na Internet e oferecem sugestões e soluções preciosas para pequenas dificuldades.

Finalmente, abra-se à totalidade do conhecimento: Outras ciências, especialmente a matemática, conhecem *shortcuts* e conexões entre os dados que não são discerníveis a partir da programação, mas permitem grandes ganhos de eficiência. Um exemplo é o uso de resultados da álgebra de corpos finitos e da geometria algébrica em criptografia. Veremos questões semelhantes no cálculo de raiz quadrada inversa e no problema da geração de números aleatórios.

Por que C e quais outras linguagens?

Escolhemos C para a maioria de nossos exemplos e dicas porque é a linguagem real que dá mais clareza aos procedimentos mais básicos de programação. Por tais paradigmas, entendemos: o controle simples de fluxo e a manipulação de dados elementares exatamente como são representados no computador digital.

Um exemplo será a discussão sobre vetores associativos: trata-se de uma abstração importante para trabalho e que realmente usaremos em alguns exemplos, importando-a de C++, mas desejamos ter o *ensejo* de discutir sua importância precisamente porque se coloca a pergunta de como poderíamos implementá-la em C.

Outros motivos são clássicos:

- ▶ C é a base e facilita o entendimento de muitas linguagens mais avançadas, como C++ e Java, ou mais modernas.
- ▶ C é a base de muito *software* em produção e uso atualmente.
- ▶ C permite a personalização de um código a um problema de modo a atingir alta eficiência.
- ▶ Finalmente, para C, o compilador gcc é um exemplo maximal de programa aberto, excelente e construído por contribuições.

Outras linguagens que usaremos

Portugol e Java: Damos exemplos, exercícios e observações nessas duas linguagens para compatibilidade com disciplinas introdutórias da UFABC.

C++: Para utilizarmos recursos não disponíveis em C, preferimos C++ por sua compatibilidade com a linguagem original e pela possibilidade de “usarmos somente o que precisamos”, sem necessidade de conhecer toda a linguagem.

bash: Para interação com o sistema operacional e manipulação de programas e arquivos, preferimos bash por ser uma linguagem apropriada para o terminal de comando.

Agradecimentos



A Daniel Morgato Martin (várias discussões sobre programação e ensino, o *benchmarking* de `iostream` e a eliminação de entradas em vetores);

2017 VCL

Variáveis

Tópicos

- ▶ O que são, tipagem e declaração.
- ▶ Entrada, operações e saída.
- ▶ Vetores e matrizes (introdução).
- ▶ Endereços e ponteiros (extra).
- ▶ Alocação dinâmica (extra).

Introdução

Variáveis são as responsáveis por armazenar os dados, valores e informações de que tratamos nos programas.

Cada variável pode ser pensada como uma “gaveta”, em que guardamos uma informação e que abrimos a toda vez que desejamos trabalhar com essa informação. Mais especificamente:

- ▶ Cada variável é uma “gavetinha” unitária, com espaço para somente uma unidade de informação, não um gavetão onde podemos apertar inúmeras camisas, meias etc.
- ▶ A variável tem um nome, como as gavetas de escritório que têm etiquetas únicas para identificar seu conteúdo.

Desse modo, as variáveis em programação funcionam muito analogamente às variáveis em matemática, que têm nomes (x, a, θ) e também assumem apenas um valor por vez ($x = 5$). (Contudo, em matemática, as variáveis podem ser incógnitas ou meros símbolos notacionais etc.)

Nesta parte, nosso propósito é verificar como ler dados, armazená-los em variáveis, modificá-los atualizando essas variáveis e retorná-los ao usuário.

Exercícios com arquivos

Arquivos de computador também são “gavetas” de dados e têm nomes, então os utilizaremos para a primeira prática com variáveis.

Usaremos fundamentalmente a noção de que operações sucessivas com dados devem ser realizadas em uma ordem determinada para que algum resultado seja alcançado: ressaltaremos essa importância em breve ao tratar de “instruções sequenciais”.

Problema 1

Um escritório tem uma política de *backup* (armazenamento de cópias anteriores de arquivos) por três dias, realizado diariamente ao fim do expediente, segundo estas normas: o arquivo original *dados.txt* deve ter o nome *dados-1.txt*, a cópia da véspera deve ter o nome *dados-2.txt* e a cópia da antevéspera deve ter o nome *dados-3.txt*. (Continua...)

Você completou o dia de trabalho e deve atualizar os nomes do arquivo, dispondo de um novo *dados.txt*. Em vez de renomear cada arquivo usando a interface gráfica do computador, a empresa impõe que se use o comando `renomear`, que é chamado assim:

```
renomear NomeOriginal NomeNovo
```

Atenção: Se algum arquivo já existe com o nome novo, ele será apagado pelo conteúdo do arquivo sendo renomeado.

Liste três chamadas do comando `renomear` na ordem correta para que a política de *backup* seja obedecida.

Na área de trabalho do Linux, crie quatro arquivos de texto com os conteúdos abaixo e utilize o comando `mv -f` para realizar a tarefa de renomear acima.

Identifique o conteúdo dos arquivos após a execução de cada uma das três chamadas exatamente como você listou na primeira questão; marque um traço se o arquivo não existir mais.*

arquivo	<i>dados.txt</i>	<i>dados-1.txt</i>	<i>dados-2.txt</i>	<i>dados-3.txt</i>
contém no início	vermelho	verde	azul	amarelo
após 1ª chamada				
após 2ª chamada				
após 3ª chamada				

*Para visualizar um arquivo alterado, poderá ser necessário fechar o editor de texto e reabri-lo. Experimente, porém, o comando `head NomeArquivo`, que exibe as primeiras linhas do arquivo no próprio terminal.

Problema 2

Seu chefe entrega-lhe dois arquivos *vendas.txt* e *clientes.txt*, para você substituir o conteúdo de um pelo outro (de modo que *vendas.txt* tenha o conteúdo original de *clientes.txt*, mas este, por sua vez, também tenha o conteúdo original de *vendas.txt*).

Liste três chamadas do comando `renomear` na ordem correta para que a determinação seja cumprida. Não vale usar “copiar e colar”, seja o conteúdo, seja o próprio arquivo. E claro, teste sua solução com arquivos reais e o comando `mv -f`.

Dica: você precisará de um 3º arquivo, digamos, *temp.txt*.

Especificações

Tipos

As diversas linguagens de programação dividem-se quanto a quais tipos de informação uma variável pode armazenar.

Algumas requerem a especificação de um “tipo”, assim como uma gaveta que é determinada apenas para meias, outra apenas para camisetas etc.

Outras admitem que as variáveis são todas de um mesmo tipo padrão (uma sequência de letras, ou um número real, dentre outras possibilidades), podendo ter ou não mecanismos que permitem especificar um tipo.

Exemplos de tipos são:

- ▶ inteiro em Portugal para números inteiros. Em C, C++ e Java, temos os tipos com `int`, `long`, `short` entre outras, com maior ou menor capacidade e com modificadores opcionais como `unsigned`.
- ▶ real em Portugal para números quebrados. Também em C, C++ e Java, temos `float` e, com maior precisão, `double`, também com opções.
- ▶ cadeia em Portugal para sequências de letras, como palavras e frases. Em C++ e Java, o tipo correspondente é `string`. Contudo, em C, não temos um tipo padronizado, sendo necessário usar vetores de tipo `char`, isto é, sequências de caracteres que precisam ter um tamanho máximo especificado ou serem alocadas manualmente.

Declaração

Linguagens diferentes também requerem, ou não, informações específicas para lidar com variáveis.

Algumas exigem a *declaração* da variável no início do programa ou procedimento, que consiste em informar o tipo e o nome da variável para que a memória correspondente seja providenciada.

Outras permitem o uso de novas variáveis ao longo do programa, bastando fazer uso de seus nomes.



Trabalharemos mais com as linguagens de variáveis tipadas e declaradas, como C, Portugol, Java e C++.

Já bash privilegia variáveis não tipadas e não declaradas. Linguagens assim requerem mais cuidado na programação e na documentação e também oferecem mecanismos para declaração e tipagem.

2017 VCL

Operações em sequência

Assim como criamos arquivos e introduzimos conteúdo neles, as variáveis são declaradas e *inicializadas*.

É um bom hábito sempre inicializar uma variável com algum valor (nulo ou vazio), para evitar algum uso inadvertido com os *bits* que por acaso estiverem acionados na sua posição de memória (o que produz resultados ininteligíveis e erros nas primeiras execuções).

A inicialização pode ser feita no momento da declaração e é um exemplo particular de atribuição:

```
int i, j, k=10, n=5;  
float x=1.0;
```

Atribuição

A operação fundamental com uma variável é a atribuição, que tem uma sintaxe própria em cada linguagem. Naquelas que consideramos, segue-se a forma:

nome da variável := expressão envolvendo variáveis

Não se trata de uma equação!

Ao contrário de nossa ordem habitual de leitura, o primeiro objeto a ser interpretado é o membro direito da atribuição, em que as variáveis são “chamadas” segundo seus nomes e os valores atuais correspondentes são substituídos na expressão aritmética (por exemplo), cujo valor final é então calculado.

Somente depois esse valor é substituído na variável nomeada no membro esquerdo, independentemente do valor original que poderá ter sido utilizado no membro direito.

Abreviaturas

Várias formas de atribuição frequentemente utilizadas têm seus próprios comandos para um código mais simples, específico e reconhecível:

- ▶ `n += 4 * i` abrevia `n = n + 4 * i;`
- ▶ `n++` abrevia `n = n + 1` etc.

Instruções sequenciais

Como vimos nos exercícios com arquivos, a ordem em que realizamos operações com as variáveis é fundamental para a obtenção dos resultados pretendidos — assim como, na fórmula de Bhaskara, primeiro se calcula o Δ , depois sua raiz quadrada, depois o restante da expressão.

2017 VCL

Exemplo da média das provas

Os próximos *slides* mostram, em Portugol, Java e C, como solicitar dois números fracionários e emitir a média deles.

Precisaremos utilizar comandos de entrada e saída, que são mais claros em Portugol e aos quais logo daremos atenção.

Tanto esses comandos como as operações correspondentes devem vir em uma sequência correta.

Média em Portugol

```
programa {  
funcao inicio() {  
    real prova1, prova2, media  
    escreva("Qual foi a primeira nota? ")  
    leia(prova1)  
    escreva("Qual foi a segunda nota? ")  
    leia(prova2)  
    media = ( prova1 + prova2 ) / 2  
    escreva("A média é: ")  
    escreva(media)  
} }
```

(Ao rodar o programa, o PortugolStudio requerirá ponto decimal para a entrada de valores quebrados.)

Média em Java

```
package javaapplication1;
import java.util.Scanner;

public class JavaApplication1 {
public static void main(String[] args) {
    float prova1, prova2, media;
    Scanner buffer = new Scanner(System.in);
    System.out.println("Qual foi a primeira nota? ");
    prova1 = buffer.nextFloat();
    System.out.println("Qual foi a segunda nota? ");
    prova2 = buffer.nextFloat();
    media = ( prova1 + prova2 ) / 2;
    System.out.printf("A média é: %f", media);
} }
```

(Ao rodar o programa, o Netbeans aceitará números com vírgula decimal.)

Média em C

```
#include <stdio.h>

int main(void) {
    float prova1, prova2, media;
    printf("Qual foi a primeira nota? ");
    scanf("%f", &prova1);
    printf("Qual foi a segunda nota? ");
    scanf("%f", &prova2);
    media = ( prova1 + prova2 ) / 2;
    printf("A média é: %f", media);
    return(0);
}
```

(A biblioteca `stdio.h` requerirá o uso de ponto decimal.)

Exercícios com o método do quadrado do meio

O “método do quadrado do meio”, devido a von Neumann, é utilizado iteradamente para produzir números pseudoaleatórios inteiros.

Ele consiste em tomar um número com, por exemplo, quatro dígitos decimais, elevar ao quadrado obtendo um número de oito dígitos e extrair os quatro dígitos centrais, para formar um novo número de quatro dígitos.

Problema 1

Faça esse processo, com auxílio da calculadora do computador, com cada número abaixo:*

(a) 3456

(b) 3600

(c) 123

(d) 2048

Problema 2

Complete o código em Portugal a seguir com as expressões necessários para ler um inteiro e, assumindo que tem quatro dígitos, substituir seu conteúdo pelo novo número, como descrito acima.

*Atenção para o que “oito dígitos” significa no resultado intermediário!
Prática de Programação ©©©© 2017 Vinicius Cifú Lopes


```
programa {
funcao inicio() {

    inteiro num
    escreva("Entre quatro dígitos: ")
    leia(num)

    num = num * -----

    num = num / -----

    num = num % -----

    escreva("Obtemos: ", num)
} }
```

Opcionais e extras

Abrevie as atribuições feitas nesse programa.

Transcreva-o na linguagem Java.

Elabore uma versão que repita os procedimentos acima 20 vezes, imprimindo a sequência de números pseudoaleatórios. O programa deve ler um número inicial para começar: digite, por exemplo, o final do seu RA.

O quanto aleatória é a sequência produzida? (Revisitaremos esse tópico.)

No próximo *slide*, vemos um código em C que utiliza o comando `for` para repetir o mesmo bloco de instruções 20 vezes.

Solução em C

```
#include <stdio.h>

int main (void) {
    int i, num;
    scanf("%d",&num);
    for(i=1;i<=20;i++) {
        num *= num;
        num /= 100;
        num %= 10000;
        printf("%d, ", num);
    }
    return(0);
}
```

Exercícios

- (I) Escreva um programa que recebe um ângulo sexagesimal puro (lê graus, minutos e segundos) e transforma para graus decimais e para radianos. (Assuma que é uma medida positiva.)
- (II) Escreva um programa que recebe um ângulo medido em graus decimais e transforma para graus sexagesimais (graus, minutos e segundos).

O item II requer chamar uma biblioteca (em Portugol) e utilizar funções de truncamento (não arredondamento) de um real para um inteiro.*

*Há muitas variantes de Portugol com diferentes nomes para essas funções.
Prática de Programação ©©©© 2017 Vinicius Cifú Lopes

Solução completa de II em Portugol

```
programa {
inclua biblioteca Tipos --> tip
funcao inicio() {
    inteiro graus, minutos
    real segundos, GrausDecimais, temp

    escreva("Digite um número positivo: ")
    leia(GrausDecimais)

    graus = tip.real_para_inteiro(GrausDecimais)
    temp = GrausDecimais - graus
    temp = temp * 60
    minutos = tip.real_para_inteiro(temp)
    temp = temp - minutos
    segundos = temp * 60

    escreva("Igual a ",
            graus, ":", minutos, ":", segundos) } }
```

O código principal de II em C e Java

Não há necessidade de incluir uma biblioteca:

```
graus = (int) GrausDecimais;  
temp = GrausDecimais - graus;  
temp = temp * 60;  
minutos = (int) temp;  
temp = temp - minutos;  
segundos = temp * 60;
```

Exemplo extra

Jogos gráficos antigos não contavam com processadores com instruções dedicadas, então os autores de “Quake III” usaram uma aproximação rápida para calcular $1/\sqrt{x}$:

```
xmeio=x*0.5F;
y=x;
n=*(long*)&y;
n=0x5f3759df-(n>>1);
y=*(float*)&n;
y=y*(1.5F-(xmeio*y*y));
// y=y*(1.5F-(xmeio*y*y));
```

Para saber como

Pesquise sobre “raiz quadrada inversa rápida”.

Entrada e saída

Procedimentos de entrada e saída de dados (ou “leitura” e “impressão”) em diferentes linguagens atendem aos diferentes propósitos dessas linguagens.

Em Portugal

Para praticidade, os dois comandos são muito simples:

- ▶ `leia(NomeVariável)` simplesmente entra o valor digitado (seguido de *enter*) na variável de nome especificado.
- ▶ `escreva` toma cada argumento, dentre um ou mais separados por vírgulas, e imprime o valor da variável em questão ou, no caso de caracteres entre aspas, imprime tais caracteres.

Em Java

Java é voltada para manipulação de dados abstratos (por exemplo, ambientes de interação, janelas e protocolos), ou “objetos”, que têm sua própria entrada e saída. Em consequência, a interface para texto simples é mais pesada, embora forneça “atalhos” para leitura de tipos específicos e métodos inspirados em C.

Em C

C é adequada para programação de sistemas, em que procedimentos de entrada e saída são adaptados conforme a necessidade. Por isso, a linguagem não tem comandos nativos para ler e escrever.

Embora isso pareça absurdo, torna-se muito natural quando se pensa que um controlador de semáforo não precisa ler números ou imprimir textos.

Para uso simples, a biblioteca padrão `stdio.h` oferece inúmeras funções, das quais convém conhecer `printf` e `scanf` (sem prejuízo a outras).

O 1º argumento de ambas é uma *cadeia de controle* entre aspas, que contém a mensagem a ser escrita ou lida. Os símbolos `%` e `\` têm função especial nessa cadeia e indicam:

- ▶ no caso de `%`, o tipo da variável a ser escrita ou lida (o nome da variável em si vem na lista de argumentos subsequentes, em ordem de chamada), além de especificações para formatação;
- ▶ no caso de `\`, um caractere especial, como a nova-linha `\n`.

Os especificadores `%d` (inteiro *decimal*), `%f` (*floating*) e `%s` (*string*) são os mais comuns.

A maioria das variáveis na lista de argumentos de `scanf` tem o nome precedido pelo *operador de endereço* `&`.

Isso porque `scanf` é uma função e, em C, funções recebem os valores das variáveis, não seus nomes.

Portanto, para registrar uma informação em uma variável, não precisamos o valor atual da mesma, e sim seu endereço de memória. (A exceção são vetores de caracteres para `%s`, que já são endereços e não comportam `&`.)

Apresentaremos endereços de memória ainda nesta parte.

Quais opções eu tenho?

Para trabalhar com facilidade em C, sugerimos as seguintes alternativas:

- (1) Aprender `printf` e `scanf` como mencionado nos *slides* acima. Requer um pouco de prática e paciência, mas é a melhor opção. Faremos uso livre neste material.
- (2) Servir-se de “macros” para criar comandos diretos, necessariamente adaptados a cada tipo. (Veja o próximo *slide*.)
- (3) Usar a biblioteca padrão de C++. (Veja os *slides* seguintes.)

Macros para C

Ponha isto no seu cabeçalho:

```
#include <stdio.h>
#define putd(A) printf("%d",A)
#define putf(A) printf("%f",A)
#define getd(A) scanf("%d",&A)
#define getf(A) scanf("%f",&A)
#define getw(A) scanf("%s",A)
```

Para escrever sequências de caracteres delimitadas por aspas, use a própria `printf` ou `puts`, já disponível em `stdio.h`.

A correspondente `gets` lê uma linha inteira.

Leitura e escrita em C++

A biblioteca padrão trabalha todos os tipos da mesma forma. Vejamos um pequeno exemplo completo:

```
#include <iostream>
using namespace std;

int main(void) {
    int idade;
    string nome;
    cout << "Qual é o seu nome? ";
    cin >> nome;
    cout << "Quantos anos você tem? ";
    cin >> n;
    cout << nome << " tem " << idade << " anos.\n";
    return(0);
}
```

Notas extras

`using namespace std` pode criar conflitos, em um programa mais complexo, com outros nomes que `std` compartilhe com outras bibliotecas ou códigos não padronizados.

Pode-se substituí-lo especificamente por

```
using std::cin;  
using std::cout;  
using std::string;
```

que identificam exclusivamente esses nomes, mas ainda há chances de conflito.

Recomenda-se, em vez disso, escrever por extenso `std::cin` e similares em todo o seu programa.

Considere pôr isto em seu cabeçalho (ainda será necessário compilar como C++):

```
#include <iostream>  
#define IN(A) std::cin >> A  
#define OUT(A) std::cout << A
```

Finalmente, deve-se notar que `cin` e `cout` são consideravelmente mais lentos que `scanf` e `printf`.

Um pouco de vetores e matrizes



(Futuramente, para uso básico com laços.)

2017 VCL

Endereços e ponteiros (extra)



(Futuramente, sobre & e *.)

2017 VCL

Alocação de memória (extra)



(Futuramente, sobre malloc e free.)

Conheça também

...os comandos `new` e `delete` em C++.

2017 VCL

Pensamento algorítmico

Tópicos

- ▶ Instruções sequenciais.
- ▶ Controle de fluxo e fluxogramas.
- ▶ Construções típicas: desvios condicionais e laços de repetição.

Condicionais

Tópicos

- ▶ Construções básicas “se–então” e “se–então–senão”.
- ▶ A construção encadeada “senão–se”.
- ▶ Combinações booleanas de testes.

Casos e encadeamento

Decodificação do RA (1ª parte)

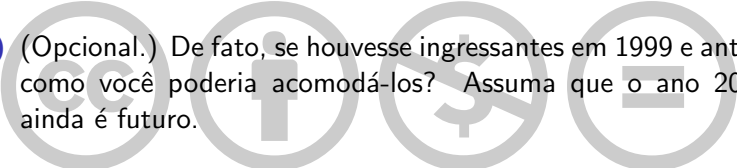
Antigamente, o “Registro do Aluno” (RA) ou número de matrícula na UFABC consistia de oito dígitos `CTXXXXAA` que codificavam diversas informações sobre o discente, assim:

- ▶ `C` identificava o *campus* de ingresso: 1 para Santo André e 2 para São Bernardo do Campo.
- ▶ `T` identificava o tipo de estudo: 1 para graduação; 2 para especialização; 3 para mestrado; 4 para doutorado; 5 para mobilidade; 6 para especial.
- ▶ `XXXX` era um número individual de cadastro.
- ▶ `AA` eram os dois últimos dígitos do ano de ingresso.

Escreva o código em Portugol para ler um número de RA e decodificá-lo conforme as especificações abaixo:

- (i) Cada bloco condicionado (simples, composto ou encadeado) deve ter início e fim identificados univocamente com comentários e deve ter corpo recuado em relação ao código circundante, conforme profundidade da condicional.
- (ii) Pode ser necessário declarar mais variáveis no início do programa, juntamente com o número dado: use somente variáveis do tipo inteiro e liste-as em separado.

- (iii) O código deve determinar cada uma das seguintes informações do aluno e indicá-las para o usuário: *campus* de ingresso, categoria, número de cadastro e ano de ingresso (escreva-o na forma 20AA).
- (iv) Caso uma informação não seja conhecida, deve-se indicá-la também; por exemplo, *campus* 3 como “*Campus inválido*”.
- (v) O seu código funciona para ingressantes entre 2007 e 2009?

- 
- (vi) (Opcional.) De fato, se houvesse ingressantes em 1999 e antes, como você poderia acomodá-los? Assuma que o ano 2050 ainda é futuro.
 - (vii) (Opcional.) Retomaremos esse exercício na seção sobre combinações booleanas e seu código deverá funcionar quando inserido nas duas versões oferecidas lá.

2017 VCL

Combinações booleanas

Decodificação do RA (conclusão)

Fizemos a decodificação do “Registro do Aluno”, anteriormente, para qualquer número informado. Podemos verificar previamente se se trata de um número de oito dígitos.

A seguir, oferecemos duas versões de código para a estrutura principal do programa. Complete cada condicional com as expressões necessárias para garantir que o RA lido esteja entre 10.000.000 e 99.999.999 (formato antigo), inclusive.

Atente para a estrutura de cada versão. Ambas estão em Português.

Versão 1

```
se (-----)
{ // início do bloco que decodifica o RA
    /* a resposta do primeiro
       exercício vem aqui */

} // fim do bloco que decodifica o RA

senao
{
    escreva("Formato inadequado.")
}
```

Versão 2

```
se (-----)
{
    escreva("Formato inadequado.")
}

senao
{ // início do bloco que decodifica o RA

    /* a resposta do primeiro
       exercício vem aqui */

} // fim do bloco que decodifica o RA
```

Exercício clássico

Leia e armazene quatro inteiros: dois pares de abscissas e coordenadas de casas em um tabuleiro de xadrez.

Verifique se duas damas posicionadas nessas casas estão uma interceptando a outra (com o restante do tabuleiro vazio).

No xadrez, as damas movem-se ao longo das linhas, das colunas e das diagonais, um número qualquer de casas.

Futuramente

Esse será um pequeno ingrediente para resolvermos o “Problema das Oito Damas”.

Exercícios clássicos

Sugestão: Leia e armazene cada inteiro em uma variável separada, por exemplo, cada número de horas ou de minutos.

- (1) Solicite e leia dois horários e determine qual vem antes (em um mesmo período entre 0:00 e 11:59).
- (2) Solicite e leia duas datas, de nascimento e atual, e determine quantos anos *completos* uma pessoa nascida na primeira data teria na segunda, ou informe um erro caso a data de nascimento seja posterior à atual.

Para saber mais

Pesquise sobre “ordem lexicográfica”.

Iterações

Tópicos

- ▶ Construções básicas “enquanto” e “para”.
- ▶ Manipulação simples de vetores e matrizes.

2017 VCL

Discussão e resumo de paradigmas

Embora múltiplas combinações sejam possíveis e ocorram na prática, temos dois modelos principais de controle de ciclos:

Índice percorre conjunto

A variável tem valores inicial, limitante e de incremento e habitualmente se usa um comando `para` ou `for`, como nestas linhas:

- ▶ `for(i=0;i<N;i++) {...}`
- ▶ `for(k=64;k>0;k%=2) {...}`
- ▶ `for(n=0;a[n]!='\0';n++) {...}`

A solução para o problema de somar 12 faturamentos mensais é emblemática desse caso.

O problema de somar números até uma entrada nula também pode ser resolvido em tal formato:

```
for(total=0;
    scanf("%f",&num) && num!=0 ;
    total+=num) ;
```

(O bloco de comandos interno ao laço é vazio.)

Veja que **while**((c=getchar())!=EOF) {...}, que é uma construção bastante comum, essencialmente segue a mesma estrutura.

Controle é alterado dentro do bloco

Uma variável é testada isoladamente ou a construção é mais complexa e melhor desdobrada com `enquanto` ou `while`:

```
while (s==TRUE) {  
    ...  
    if(...) s=FALSE;  
    ...  
}
```

Repetições explícitas

A instrução de laço impõe a repetição não apenas do bloco de comandos destacado, mas também da condicional associada e do comando de avanço.

Em casos que esse teste tem complexidade similar (ou maior) que os comandos do ciclo, pode ser interessante a repetição explícita dos comandos afinal.

Um pequeno exemplo seria, no código que vimos para $1/\sqrt{x}$, o uso do segundo passo do método de Newton.

Para saber mais

Pesquise sobre “desenrolamento de laços” e o “dispositivo de Duff” (*Duff's device*).

Vetores

Tópicos

- ▶ Vetores indexados.
- ▶ Operações com vetores.
- ▶ Vetores associativos.

2017 VCL

Manipulações

Eliminação de entradas

Dado um vetor de elementos, remova os que não satisfazem determinada condição, trazendo os remanescentes para o início da lista e diminuindo o comprimento da mesma.

Conte o número de movimentações feitas dos elementos.

Por exemplo, mantenha apenas os números estritamente positivos, completando o código no próximo *slide*.

```
#include <stdio.h>

int main(void) {
    int i, j, total=12, movimentos=0,
        a[12]={-1,2,3,-4,5,-6,-7,
              -8,9,10,-11,12};

    /* (insira código aqui) */

    printf("Restam %d positivos "
           "após %d movimentos:\n",
           total, movimentos);
    for(i=0;i<total;i++)
        printf("%d, ", a[i]);
    printf("\n");
    return(0);
}
```

Primeira resposta

Para remover um elemento indesejado (negativo ou nulo), deslocamos o restante do vetor uma posição para a frente, sobrescrevendo-o:

```
for (i=0; i<total; i++)
    if (a[i] <= 0)
    {
        total--; // (a)
        for (j=i; j<total; j++)
        {
            a[j]=a[j+1];
            movimentos++;
        }
        i--; // (b)
    }
```

Notas

- (a) Essa linha atualiza o total; pode ser colocada após o ciclo de deslocamento, porém, então ele será condicionado por $j < \text{total} - 1$.
- (b) Ponto importante: ao antecipar elementos do vetor, precisamos testá-los também.

Observe que antecipamos *todos* os elementos posteriores quando eliminamos qualquer um. Desse modo, um mesmo elemento pode ser movido diversas vezes.

Segunda resposta

Percorremos o vetor em busca dos elementos a manter, usando um segundo índice para preenchê-lo somente quando encontramos um desejado:

```
for (i=0, j=0; i<total; i++)
    if (a[i]>0) // (a)
    {
        if (i>j) // (b)
        {
            a[j]=a[i];
            movimentos++;
        }
        j++; // (c)
    }
total=j; // (d)
```

Notas

- (a) Note a condicional negada em relação à da 1ª resposta; $a[i]$ será *mantido*.
- (b) O teste dos índices distintos deve ser mais econômico que uma movimentação do elemento por sobre si mesmo, mas é opcional. A diferença só ocorre após a primeira exclusão.
- (c) j somente é atualizado para a próxima posição a preencher.
- (d) Neste momento, j está uma posição à frente do último elemento mantido e é o comprimento da nova lista.

Vale a pena testar ambas as respostas!

Para o vetor exemplificado, a 1ª realiza 35 movimentos (inclusive de elementos a remover) e a 2ª apenas 6, ou seja, apenas um por elemento a manter *após* a primeira remoção, se houver.

De fato, a 1ª solução é quadrática sobre o número de elementos, enquanto a 2ª é linear.

2017 VCL

Movimentação na memória

Suponha que a é um vetor de tipo `char`, com $(\text{int}) N$ elementos. Para especificar um subintervalo, no mesmo vetor, identificamos sua posição inicial e seu comprimento (ambos `int`). Dados dois subintervalos:

- ▶ verifique se ambos têm o mesmo comprimento e qual começa à esquerda do outro;
- ▶ verifique se os subintervalos são disjuntos;
- ▶ nesse caso, copie o primeiro para o segundo;
- ▶ ainda nesse caso, permuta o conteúdo (em ordem) de ambos, com apenas uma variável auxiliar;

- ▶ em caso de sobreposição, copie o primeiro para o segundo sem vetor ou variável auxiliar, começando, p. ex., pela extremidade direita se for copiar o esquerdo para o direito;
- ▶ ainda nesse caso, teste se é possível permutar o conteúdo (em ordem) de ambos e faça-o com apenas uma variável auxiliar.

Conheça também

...as funções `memcpy` e `memmove` e a declaração `restrict`.

Programação modular

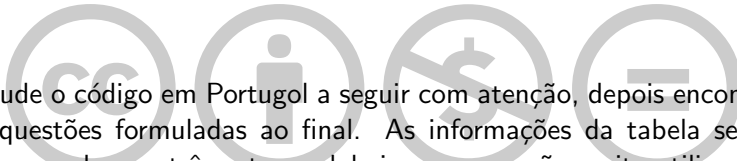
Tópicos

- ▶ Funções e subrotinas.
- ▶ Variáveis globais.
- ▶ Modularização e o paradigma de “cliente–interface–implementação”.

Minibanco de dados

O enunciado

Você participa de um diretório estudantil que está colaborando com o desenvolvimento de uma nova metodologia para as aulas de uma disciplina. A professora responsável cedeu uma tabela de alunos, suas notas nessa disciplina e seus coeficientes de rendimento, para o grupo realizar um levantamento e apresentar ao ConsEPE. Os demais integrantes do grupo optaram por criar um programa para a apresentação, em vez de uma planilha, e precisam que você o complete com a implementação de algumas funções.



Estude o código em Portugol a seguir com atenção, depois encontre as questões formuladas ao final. As informações da tabela serão armazenadas em três vetores globais, porque serão muito utilizadas; os vetores serão paralelos, ou seja, as informações sobre um mesmo aluno estarão armazenadas nas posições de mesmo índice.

2017 VCL


```
programa {
```

```
    cadeia Nome[5]
```

```
    real Nota[5]
```

```
    real CR[5]
```

```
    funcao inicio() {
```

```
        Inicializa_Dados()
```

```
        Ordena_por_Nota()
```

```
        Imprime_Dados()
```

```
        Ordena_por_CR()
```

```
        Imprime_Dados()
```

```
    }
```

```
funcao Inicializa_Dados() {  
    Nome[0]="Joana"  
    Nota[0]=9.0  
    CR[0]=3.1  
    Nome[1]="Alberto"  
    Nota[1]=4.5  
    CR[1]=2.7  
    Nome[2]="Miguel"  
    Nota[2]=7.5  
    CR[2]=2.3  
    Nome[3]="Debora"  
    Nota[3]=5.0  
    CR[3]=1.9  
    Nome[4]="Mariana"  
    Nota[4]=8.0  
    CR[4]=3.8  
}
```

```
funcao Imprime_Dados() {
    // (insira código aqui: questão 1)
}

funcao Ordena_por_Nota() {
    inteiro i, j
    // realmente começa em 1
    para(i=1;i<5;i++) {
        para(j=i;j>0;j--) {
            se(Nota[j-1]>Nota[j]) {
                Permuta_Elementos(j-1,j)
            }
        }
    }
}
```

```
funcao Permuta_Elementos(inteiro a, inteiro b) {  
    // (insira código aqui: questão 2)  
}  
  
funcao Ordena_por_CR() {  
    // (insira código aqui: questão 3)  
}  
  
} // fim do programa
```

2017 VCL

Questões

- (1) Implemente a função `Imprime_Dados`, que deve escrever as três informações de um aluno em cada linha, separadas por `\t`.
- (2) Implemente a função `Permuta_Elementos`, que inverte os dados dos alunos nas duas posições indicadas pelos argumentos.
- (3) Implemente a função `Ordena_por_CR` adaptando o código de `Ordena_por_Nota` para ordenar os alunos por seu coeficiente de rendimento.

Observação

Em Ordena_por_Nota acima, utilizamos o método de ordenação “por inserção”. Na literatura, encontra-se uma redação mais enxuta:

```
para(i=1; i<5; i++) {  
    para(j=i; j>0 e Nota[j-1]>Nota[j] ; j--) {  
        Permuta_Elementos(j-1, j)  
    }  
}
```

Essa organização se baseia no fato de muitas linguagens, inclusive C e Java, somente verificarem cada componente de uma conjunção (combinação “e”) se as anteriores já resultarem verdadeiras.

Em Portugol, esse código não funciona, porque mesmo com $j=0$ e a 1ª parte da conjunção sendo falsa, ele tenta calcular a 2ª parte.