Edwar Saliba Júnior

Dicas, Comandos e Exemplos Comparativos entre Linguagem Algorítmica e Linguagem C / C++

Belo Horizonte
2009

Sumário

1 Nota:

Os códigos em Linguagem C apresentados neste trabalho foram criados utilizando a ferramenta conhecida por "Dev-C++" versão "4.9.9.2". Esta ferramenta poderá ser adquirida gratuitamente no site abaixo:

http://www.bloodshed.net/devcpp.html

2 Comandos e Palavras Reservadas:

Como mecanismo facilitador, vejamos a tabela abaixo relativa a alguns comandos básicos e palavras reservadas existentes em algoritmos, e respectivamente em Linguagem C/C++.

Comando em Algoritmo	Comando em Linguagem C/C++
início	{
fim	}
declare	(não existe)
Real	float
inteiro	int
carácter	char
escreva	cout | printf
leia	cin | scanf
se	if
senão	else
caso	switch ... case
enquanto	while
para	for
repita ... enquanto	do ... while
←	=

Alguns sub-comandos mais utilizados:

Caracteres especiais	Significado
\n	Salto de linha.
\0	Nulo.
\r	Retorno do carro.
\\	Barra invertida.

Operadores:

Operador	Símbolo
Adição	+
Subtração	-
Divisão	/
Multiplicação	*
Módulo (resto da divisão inteira)	%

No capítulo 4 serão apresentados diversos problemas, seus algoritmos e seus respectivos programas em linguagem C/C++.

3 Dicas

3.1 Strings em Linguagem C / C++

3.1.1 Ao Imprimir tem Lixo na Variável Junto ao Valor Armazenado:

Para campos de texto, deixar sempre um carácter a mais do que o necessário.

Por exemplo: Tomemos um campo para guardar data com o seguinte formato:

dd/mm/aaaa

Podemos verificar que para armazenarmos a data com o formato acima, são necessários 10 posições de memória. Exemplo:

```
char data[10];
```

Porém, a prática nos mostrou que se criarmos a variável com somente as 10 posições necessárias, ao imprimirmos seu valor o Dev-C++ imprimirá lixo junto ao valor atribuído para a variável. Ao contrário, se criarmos a variável com 11 posições de memória, exemplo:

```
char data[11];
```

Ao imprimirmos será impresso o valor correto.

3.1.2 Em Tempo de Execução o Programa está Pulando Dois ou Mais Campos ao Mesmo Tempo:

Ao trabalharmos com ***strings*** em linguagem C/C++ devemos tomar certos cuidados. A linguagem C/C++ não é muito amigável quando se trata de ***strings***.

Ao usarmos o seguinte programa abaixo:

```
(01)#include <cstdlib>
(02)#include <iostream>

(03)using namespace std;

(04)int main(int argc, char *argv[])
(05){
(06)    int identificador;
(07)    char nome[50];
```

```
(08)     char dataNascimento[11];
(09)     char sexo;

(10)     cout << "Digite o identificador: ";
(11)     cin >> identificador;
(12)     cout << "Digite o nome: ";
(13)     gets (nome);
(14)     cout << "Digite a data de nascimento: ";
(15)     gets (dataNascimento);
(16)     cout << "Digite o sexo (M/F): ";
(17)     cin >> sexo;

(18)     cout << "\n\n\n=== Os Dados Digitados Foram ===\n\n\n";

(19)     cout << "Identificador  : " << identificador;
(20)     cout << "\nNome           : " << nome;
(21)     cout << "\nData Nascimento: " << dataNascimento;
(22)     cout << "\nSexo (M/F)     : " << sexo;

(23)     cout << "\n\n";

(24)     system("PAUSE");
(25)     return EXIT_SUCCESS;
(26)}
```

O usuário deste notará que ao apertar tecla ***<Enter>*** após ter digitado o número do "identificador" (linha 11), o programa saltará diretamente para o campo "data de nascimento" (linha 15) sem pedir os dados para o campo "nome" (linha 13).

Porém, isto não acontecerá no programa abaixo:

```
(01)#include <cstdlib>
(02)#include <iostream>

(03)using namespace std;

(04)int main(int argc, char *argv[])
(05){
(06)     int identificador;
(07)     char nome[50];
(08)     char dataNascimento[11];
(09)     char sexo;

(10)     cout << "Digite o identificador: ";
(11)     cin >> identificador;
(12)     fflush(stdin);
(13)     cout << "Digite o nome: ";
(14)     gets (nome);
(15)     cout << "Digite a data de nascimento: ";
(16)     gets (dataNascimento);
(17)     cout << "Digite o sexo (M/F): ";
(18)     cin >> sexo;

(19)     cout << "\n\n\n=== Os Dados Digitados Foram ===\n\n\n";
```

```
(20)     cout << "Identificador  : " << identificador;
(21)     cout << "\nNome           : " << nome;
(22)     cout << "\nData Nascimento: " << dataNascimento;
(23)     cout << "\nSexo (M/F)     : " << sexo;

(24)    cout << "\n\n";

(25)    system("PAUSE");
(26)    return EXIT_SUCCESS;
(27)}
```

O programa a seguir é idêntico ao anterior com apenas uma modificação, a linha 12 (em vermelho). Este comando limpa o ***buffer*** do teclado após ocorrido o evento de "apertar da tecla ***<Enter>***" do comando ***cin*** da linha 11. Este pequeno comando garantirá que nosso programa funcione corretamente.

Sempre que for usar um comando para leitura de ***strings*** via teclado (o comando ***gets***, por exemplo), antes de usá-lo escreva um comando ***fflush(stdin)***, para garantir que o ***buffer*** estará vazio no momento da execução do comando ***gets***.

Caso isto não seja feito, o comando tomará o que estiver no ***buffer*** do teclado como sendo a ***string*** digitada. Por isto acontece o salto "automático" de campo, sem nenhuma digitação, no primeiro programa (sem o comando ***fflush***).

3.1.3 Como Apagar um Campo *String*:

Uma ***string*** quando declarada como sendo uma seqüência de caracteres, tipo:

```
char nome[50];
```

A variável 'nome', nada mais é do que um vetor de caracteres.

Uma ***string*** em linguagem C/C++ deve ser finalizada pelo carácter '\0' (barra invertida zero). Que é o carácter que representa fim de ***string*** na linguagem C/C++.

Portanto, sabendo disto, para apagarmos uma ***string*** basta atribuirmos o carácter de fim de ***string*** para a primeira posição do vetor. Exemplo:

```
nome[0] = '\0';
```

Lembrando que todo vetor em linguagem C/C++ começa da posição 0 (zero) e vai até a quantidade reservada menos 1 (um). No nosso caso, o vetor "nome" vai da posição 0 (zero) até a posição 49 (quarenta e nove), o que dá 50 posições de memória.

3.1.4 Como Atribuir Valores a Uma Variável do Tipo *String*:

As variáveis do tipo 'char' têm tratamento especial pela linguagem C. Para atribuirmos valores a variáveis deste tipo, precisamos utilizar alguns comandos especiais.

Vejamos a seguir alguns exemplos de atribuição de valores a variáveis do tipo **char.**

3.1.4.1 Exemplo de leitura e escrita de um caracter:

Para ler e imprimir um único caracter, não há tratamento especial, podemos tratar a variável do tipo *char* como se fosse uma variável do tipo *int*, ou seja, os comandos são idênticos aos que usamos para trabalhar com valores do tipo inteiro.

Vejamos o exemplo a seguir, um programa onde são criadas duas variáveis do tipo *char*, Letra1 e Letra2; sendo que a variável Letra1 recebe um valor digitado pelo usuário e a variável Letra2 recebe seu valor internamente. Após receberem seus valores, ambas são impressas na tela.

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
    char Letra1, Letra2;

    Letra2 = 'E';

    cout << "Digite uma letra qualquer: ";
    cin >> Letra1;

    cout << "\nA letra digitada foi: " << Letra1;
    cout << "\nA letra atribuida a variável internamente foi: " <<
Letra2 << "\n\n";

    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Podemos observar acima que, ao atribuirmos um valor a uma variável do tipo *char*, este valor deverá estar entre aspas simples e, quando formos ler o valor do teclado, poderemos usar o comando *cin* normalmente.

3.1.4.2 Exemplo de leitura e escrita de vários caracteres:

Vejamos a seguir um programa onde foram criadas quatro variáveis do tipo ***char*** de 14 posições de memória cada uma delas. As quatro variáveis são: Palavra1, Palavra2, Palavra3 e Palavra4.

Conforme podemos observar, o programa atribui internamente através da função ***strcpy*** as palavras "Belo Horizonte" à variável Palavra1. Em seguida é solicitado que o usuário do software entre com mais três palavras via teclado. Observe que a variável Palavra2 receberá o valor digitado do teclado através do comando ***cin***. A variável Palavra3 receberá o valor digitado do teclado através do comando ***gets***. E por último, a variável Palavra4 receberá o valor digitado do teclado através do comando ***cin.getline***.

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
    char Palavra1 [14], Palavra2 [14], Palavra3 [14], Palavra4 [14];

    /* Comando de atribuição para String. */
    strcpy (Palavra1, "Belo Horizonte");

    cout << "Digite uma letra qualquer: ";
    cin >> Palavra2;
    fflush(stdin);

    cout << "Digite uma letra qualquer: ";
    gets (Palavra3);

    cout << "Digite uma letra qualquer: ";
    cin.getline (Palavra4, 14);

    cout << "\nA palavra digitada na variável Palavra1 foi: " <<
Palavra1;
    cout << "\nA palavra digitada na variável Palavra2 foi: " <<
Palavra2;
    cout << "\nA palavra digitada na variável Palavra3 foi: " <<
Palavra3;
    cout << "\nA palavra digitada na variável Palavra4 foi: " <<
Palavra4 << "\n\n";

    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Ao executarmos o programa acima, conforme podemos ver na figura 01 a seguir, foi solicitado ao usuário que digitasse três palavras quaisquer. Nas três vezes em que ocorreram essas solicitações, o usuário digitou as palavras "Belo Horizonte".

Após a digitação das palavras, o programa imprime o conteúdo de cada variável na tela.

Ao verificarmos a figura 01, podemos averiguar que:

- O comando de atribuição funcionou perfeitamente, pois, o conteúdo da variável Palavra1 foi impresso perfeitamente, ou seja, "Belo Horizonte".

- O mesmo não ocorreu com o comando ***cin*** para a variável Palavra2 que só armazenou a palavra "Belo" e descartou a palavra "Horizonte". O comando ***cin*** não reconhece "espaços", por isto a palavra "Horizonte" não foi armazenada.

- Já o comando ***gets***, armazenou perfeitamente na variável Palavra3, a palavra "Belo Horizonte".

- E por último, o comando ***cin.getline*** (que solicita como parâmetros a variável para atribuição e o número de caracteres) perdeu a letra "e" da palavra "Horizonte", armazenando somente "Belo Horizont" na variável Palavra4. Isto que dizer, que quando formos utilizar o comando ***cin.getline***, é necessário passarmos como parâmetro ao comando, a quantidade de caracteres que queremos armazenar na variável mais 1. Resumindo: se precisamos armazenar 14 letras, então passamos como parâmetro o valor 15, desta forma o comando funcionará perfeitamente.

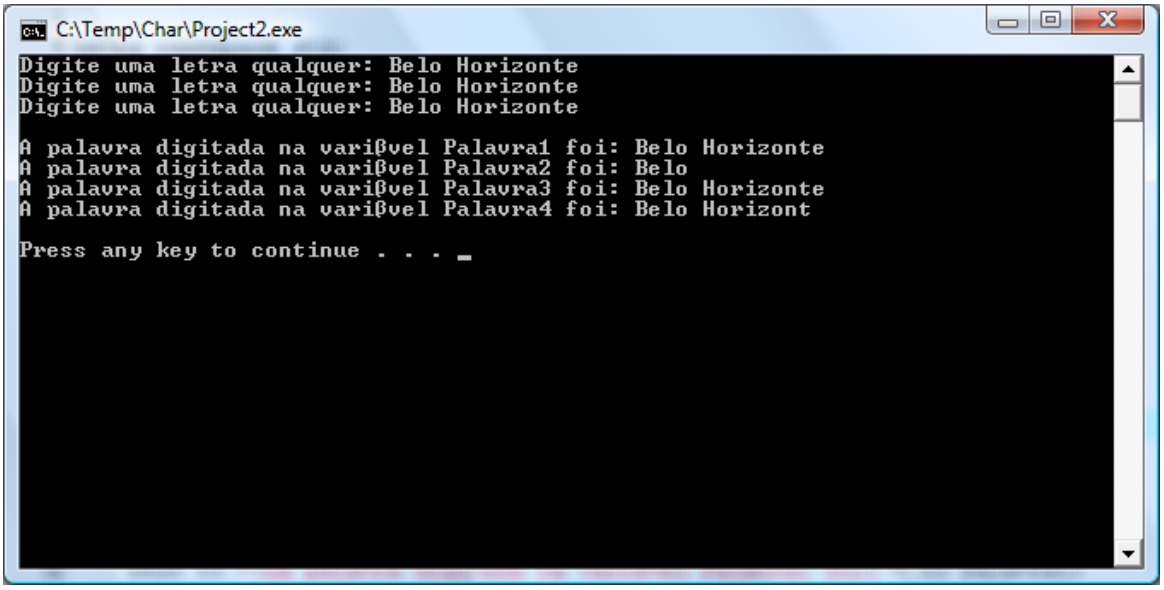

Figura 01 – Execução do Programa de Leitura de Palavras

4 Exemplos de Algoritmos em Linguagem Algorítmica e Seu Respectivo Código em Linguagem C/C++

4.1 Algoritmo usando estrutura LINEAR

```
inicio
     declare val1, val2, soma : real;

      escreva "Digite o primeiro valor: ";
      leia val1;
      escreva "Digite o segundo valor: ";
      leia val2;

      soma ← val1 + val2;

      escreva "O total da soma é: ", soma;
fim
```

O algoritmo acima descrito em linguagem C, abaixo:

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
    float val1, val2, soma;

    cout << "Digite o primeiro valor: ";
    cin >> val1;
    cout << "Digite o segundo valor: ";
    cin >> val2;

    soma = val1 + val2;

    cout <<"\n\nO total da soma e: " << soma << "\n\n";

    system("PAUSE");
    return EXIT_SUCCESS;
}
```

4.2 Algoritmo usando estrutura condicional SE

Elabore um algoritmo para ler 4 notas de um aluno (de 1 a 10). Após calcular a média das notas, apresentar a mensagem "Aprovado" se o

aluno tiver obtido média maior ou igual a 6. Caso contrário, apresentar "Reprovado".

```
início
      declare n1, n2, n3, n4, media : real;

      escreva "Digite o valor da primeira nota de 1 a 10: ";
      leia n1;
      escreva "Digite o valor da segunda nota de 1 a 10: ";
      leia n2;
      escreva "Digite o valor da terceira nota de 1 a 10: ";
      leia n3;
      escreva "Digite o valor da quarta nota de 1 a 10: ";
      leia n4;

      media ← (n1 + n2 + n3 + n4) / 4;

      se (media >= 6) então
            escreva "O aluno foi APROVADO!";
      senão
            escreva "O aluno foi REPROVADO!";
      fim se
fim
```

O algoritmo acima descrito em linguagem C, abaixo:

```
==================================================

#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
    float n1, n2, n3, n4, media;

    cout << "Digite o valor da primeira nota de 1 a 10: ";
    cin >> n1;
    cout << "Digite o valor da segunda nota de 1 a 10: ";
    cin >> n2;
    cout << "Digite o valor da terceira nota de 1 a 10: ";
    cin >> n3;
    cout << "Digite o valor da quarta nota de 1 a 10: ";
    cin >> n4;

    media = (n1 + n2 + n3 + n4) / 4;

    if (media >= 6) {
        cout <<"\n\nO aluno foi APROVADO!\n\n";
    }
    else {
        cout <<"\n\nO aluno foi REPROVADO!\n\n";
    }

    system("PAUSE");
```

```
    return EXIT_SUCCESS;
}
```

4.3 Algoritmo usando estrutura de exclusão múltipla CASE

Faça um algoritmo que mostre o menu de opções a seguir, receba a opção do usuário e os dados necessários para executar cada operação. No final, apresente o resultado da operação escolhida.

```
Menu de opções:
1 – Somar
2 – Subtrair
3 – Multiplicar
4 - Dividir

início
     declare valor1, valor2, res : real;
     declare operacao : inteiro;

     escreva "=== Calculadora de 4 Operações Básicas ===";

     escreva "Digite o 1°. valor: ";
     leia valor1;
     escreva "Digite o 2°. valor: ";
     leia valor2;

     escreva "Escolha a operação a ser realizada sobre os valores:";
     escreva "1 - Somar";
     escreva "2 - Subtrair";
     escreva "3 - Multiplicar;
     escreva "4 - Dividir;
     escreva "Escolha uma opção: ";
     leia operacao;

     caso (operacao) faça
          1 : início
                   res ← valor1 + valor2;
              fim
          2 : início
                   res ← valor1 - valor2;
              fim
          3 : início
                   res ← valor1 * valor2;
              fim
          4 : início
                  se (não (valor2 = 0)) então
                       res ← valor1 / valor2;
                   senão
                       escreva "Erro: Divisão por zero.";
                       res ← 0;
                   fim se
              fim
         exceção : início
```

```
                              escreva "Opção escolhida é inválida!";
                         fim
      fim caso

      se ((operacao >= 1) e (operacao <= 4)) então
            escreva "O resultado final é: ", res;
      fim se
fim
```

O algoritmo acima descrito em linguagem C, abaixo:
==========

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
    float valor1, valor2, res = 0;
    int operacao;

    cout << "=== Calculadora de 4 Operações Básicas ===";

    cout << "\n\nDigite 1o. valor: ";
    cin >> valor1;
    cout << "Digite 2o. valor: ";
    cin >> valor2;

    cout << "\n\nEscolha a operação a ser realizada sobre os
valores:";
    cout << "\n\n1 - Somar";
    cout << "\n2 - Subtrair";
    cout << "\n3 - Multiplicar";
    cout << "\n4 - Dividir";
    cout << "\n\nEscolha uma opção: ";
    cin >> operacao;

    switch (operacao) {
        case 1:
             res = valor1 + valor2;
             break;
        case 2:
             res = valor1 - valor2;
             break;
        case 3:
             res = valor1 * valor2;
             break;
        case 4:
             if (! (valor2 == 0)) {
                 res = valor1 / valor2;
             }
             else {
                 cout << "\n\nErro de divisão por zero!\n\n";
             }
```

```
            break;
        default:
            cout << "\n\nOpção escolhida é inválida!\n\n";
    }

    if ((operacao >= 1) && (operacao <= 4)) {
        cout << "\n\nO resultado final é: " << res << "\n\n";
    }

    system("PAUSE");
    return EXIT_SUCCESS;
}
```

4.4 *Algoritmo usando estrutura de repetição ENQUANTO*

Elabore um algoritmo que solicite que o usuário entre com dois números (inicial e final). Ao final o algoritmo deverá apresentar o valor total da soma de todos os números do intervalo digitado pelo usuário.

```
início
     declare valini, valfin, soma : real;

     escreva "Digite o valor inicial: ";
     leia valini;
     escreva "Digite o valor final: ";
     leia valfin;

     soma ← valini;

     enquanto (valini <= valfin) faça
          valini ← valini + 1;
          soma ← soma + valini;
     fim enquanto

     escreva "A soma total é: ", soma;
fim
```

O algoritmo acima descrito em linguagem C, abaixo:

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
    float valini, valfin, soma;

    cout << "Digite o valor inicial: ";
    cin >> valini;
    cout << "Digite o valor final: ";
```

```
    cin >> valfin;

    soma = valini;

    while (valini < valfin) {
          valini = valini + 1;
          soma = soma + valini;
    }

    cout << "\n\nA soma total é: " << soma << "\n\n";

    system("PAUSE");
    return EXIT_SUCCESS;
}
```

4.5 *Algoritmo usando estrutura de repetição REPITA ... ENQUANTO*

Elabore um algoritmo que mostre ao usuário um menu com 4 opções: 1 – Inclusão, 2 – Exclusão, 3 – Alteração e 4 – Sair. Obrigue o usuário a escolher uma das opções. Por fim, emita uma mensagem notificando o usuário da opção escolhida.

```
início
     declare opcao : inteiro;

     repita
          escreva "Opções";
          escreva "1 - Inclusão";
          escreva "2 - Exclusão";
          escreva "3 - Alteração;
          escreva "4 - Sair;
          escreva "Escolha uma opção: ";
          leia opcao;
     enquanto ((opcao < 1) ou (opção > 4));

     caso (opcao) faça
          1 : início
                   escreva "A opção escolhida foi: Inclusão.";
              fim
          2 : início
                   escreva "A opção escolhida foi: Exclusão.";
              fim
          3 : início
                   escreva "A opção escolhida foi: Alteração.";
              fim
          4 : início
                   escreva "A opção escolhida foi: Sair.";
              fim
     fim caso
fim
```

O algoritmo acima descrito em linguagem C, abaixo:

```
================================================

#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
    int opcao;

    do {
        cout << "\n\nOpções";
        cout << "\n\n1 - Inclusão";
        cout << "\n2 - Exclusão";
        cout << "\n3 - Alteração";
        cout << "\n4 - Sair";
        cout << "\n\nEscolha uma opção: ";
        cin >> opcao;
    } while ((opcao < 1) || (opcao > 4));

    switch (opcao) {
        case 1:
             cout << "\n\nA opção escolhida foi: Inclusão.\n\n";
             break;
        case 2:
             cout << "\n\nA opção escolhida foi: Exclusão.\n\n";
             break;
        case 3:
             cout << "\n\nA opção escolhida foi: Alteração.\n\n";
             break;
        case 4:
             cout << "\n\nA opção escolhida foi: Sair.\n\n";
    }

    system("PAUSE");
    return EXIT_SUCCESS;
}
```

4.6 Algoritmo usando estrutura de repetição PARA

Elabore um algoritmo que solicite que o usuário entre com um número inteiro qualquer. Após a entrada do número, o algoritmo deverá imprimir na tela os 30 números subseqüentes ao número digitado.

```
início
     declare num, cont : inteiro;

     escreva "Digite um número inteiro: ";
     leia num;

     para cont de 1 até 30 passo 1 faça
          escreva num + cont;
```

```
      fim para
fim
```

O algoritmo acima descrito em linguagem C, abaixo:

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
    int num, cont;

    cout << "Digite um número inteiro: ";
    cin >> num;

    for (cont = 1; cont <= 30; cont++) {
        cout << "\n" << num + cont;
    }
    cout << "\n\n";

    system("PAUSE");
    return EXIT_SUCCESS;
}
```

4.7 Algoritmo usando VETOR

Elabore um algoritmo que solicite que o usuário entre com 10 números inteiros quaisquer. Após a entrada de dados, imprimir os números digitados na ordem inversa a da digitação.

```
início
      declare num[10], i : inteiro;

      para i de 0 até 9 passo 1 faça
            escreva "Digite um número: ";
            leia num[i];
      fim para

      escreva "Números digitados, na ordem inversa a da digitação:";

      para i de 9 até 0 passo -1 faça
            escreva "O número digitado foi: ", num[i];
      fim para
fim
```

O algoritmo acima descrito em linguagem C, abaixo:

```
#include <cstdlib>
#include <iostream>
```

```
using namespace std;

int main(int argc, char *argv[])
{
    int num[10], i;

    for (i = 0; i < 10; i++) {
        cout << "Digite o " << i + 1 << "o. valor: ";
        cin >> num[i];
    }

    cout << "\n\nNúmeros digitados, na ordem inversa a da
digitação:\n";

    for (i = 9; i >= 0; i--) {
        cout << "\nO " << i + 1 << "o. valor digitado foi: " <<
num[i];
    }

    system("PAUSE");
    return EXIT_SUCCESS;
}
```

4.8 Algoritmo usando MATRIZ

Elabore um algoritmo que solicite que o usuário entre com 16 números inteiros quaisquer. Após a entrada de dados, imprimir o quadrado dos números digitados na ordem inversa a da digitação.

```
início
     declare num[4][4], i, j : inteiro;

     para i de 0 até 3 passo 1 faça
          para j de 0 até 3 passo 1 faça
               escreva "Digite um número: ";
               leia num[i][j];
          fim para
     fim para

     escreva "O quadrado dos números digitados, na ordem inversa a da
digitação:";

     para i de 3 até 0 passo -1 faça
          para j de 3 até 0 passo -1 faça
               escreva (num[i][j] * num[i][j]);
          fim para
     fim para
fim
```

O algoritmo acima descrito em linguagem C, abaixo:
==========

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
    int num[4][4], i, j;

    for (i = 0; i < 4; i++) {
        for (j = 0; j < 4; j++) {
            cout << "Digite o um número: ";
            cin >> num[i][j];
        }
    }

    cout << "\n\nO quadrado dos números digitados, na ordem inversa a
da digitação:\n";

    for (i = 3; i >= 0; i--) {
        for (j = 3; j >= 0; j--) {
            cout << "\n" << (num[i][j] * num[i][j]);
        }
    }

    cout << "\n\n" ;

    system("PAUSE");
    return EXIT_SUCCESS;
}
```

4.9 Algoritmo usando REGISTRO (tipo)

Elabore um algoritmo para cadastrar um funcionário. Para o funcionário devem-se cadastrar os seguintes dados: código, nome, data de admissão, data de nascimento e sexo. Após cadastrar o funcionário, mostrar os dados.

```
tipo funcionario
     codigo : inteiro;
     nome : texto;
     dataAdmissao : texto;
     dataNascimento : texto;
     sexo : carácter;
fim tipo;

início
     declare func : funcionario;
     declare resp : carácter;

     escreva "Digite o código do funcionário: ";
     leia func.codigo;
     escreva "Digite o nome do funcionário: ";
```

```
      leia func.nome;
      escreva "Digite a data de admissão do funcionário: ";
      leia func.dataAdmissao;
      escreva "Digite a data de nascimento do funcionário: ";
      leia func.dataNascimento;
      escreva "Digite o sexo do funcionário (M/F): ";
      leia func.sexo;

      escreva "Deseja visualizar os dados cadastrados? (S/N): ";
      leia resp;

      se (resp = 's') então
          escreva "=== Dados do funcionário ===";
          escreva "Código: ", func.codigo;
          escreva "Nome : ", func.nome;
          escreva "Data de Admissão: ", func.dataAdmissao;
          escreva "Data de Nascimento: ", func.dataNascimento;
          escreva "Sexo : ", func.sexo;
      fim se
fim
```

O algoritmo acima descrito em linguagem C, abaixo:

```
#include <cstdlib>
#include <iostream>

using namespace std;

/* Para campos de texto, deixar sempre um carácter a mais do que o
   necessário. */

struct funcionario {
       int codigo;
       char nome[50];
       char dataAdmissao[11];
       char dataNascimento[11];
       char sexo;
};

int main(int argc, char *argv[])
{
      struct funcionario func;
      char resp;

      cout << "Digite o código do funcionário: ";
      cin >> func.codigo;
      fflush(stdin);  /* fflush - Limpa o buffer do teclado. */
      cout << "Digite o nome do funcionário: ";
      gets (func.nome);
      cout << "Digite a data de admissão do funcionário: ";
      gets (func.dataAdmissao);
      cout << "Digite a data de nascimento do funcionário: ";
      gets (func.dataNascimento);
      cout << "Digite o sexo do funcionário (M/F): ";
```

```
      cin >> func.sexo;
      fflush(stdin);

      cout << "\n\nDeseja visualizar os dados cadastrados? (S/N): ";
      cin >> resp;
      fflush(stdin);

      if ((resp == 's') || (resp == 'S')) {
          cout << "\n\n=== Dados do funcionário ===";
          cout << "\nCódigo: " << func.codigo;
          cout << "\nNome : " << func.nome;
          cout << "\nData de Admissão: " << func.dataAdmissao;
          cout << "\nData de Nascimento: " << func.dataNascimento;
          cout << "\nSexo : " << func.sexo << "\n\n";
     }

     system("PAUSE");
     return EXIT_SUCCESS;
}
```

4.10 Algoritmo usando REGISTRO em VETOR

Elabore um algoritmo para cadastrar 1000 funcionários. Para cada funcionário devem-se cadastrar os seguintes dados: código, nome, data de admissão, data de nascimento e sexo. Após cadastrar o(s) funcionário(s), mostrar os dados.

```
tipo funcionario
      codigo : inteiro;
      nome : texto;
      dataAdmissao : texto;
      dataNascimento : texto;
      sexo : carácter;
fim tipo;

início
      declare func[1000] : funcionario;
      declare resp : carácter;
      declare cont, aux_cont : inteiro;

      cont ← 0;
      resp ← 's';

      enquanto ((cont < 1000) e (resp = 's')) faça
            escreva "Digite o código do funcionário: ";
            leia func[cont].codigo;
            escreva "Digite o nome do funcionário: ";
            leia func[cont].nome;
            escreva "Digite a data de admissão do funcionário: ";
            leia func[cont].dataAdmissao;
            escreva "Digite a data de nascimento do funcionário: ";
            leia func[cont].dataNascimento;
            escreva "Digite o sexo do funcionário (M/F): ";
```

```
            leia func[cont].sexo;

            cont ← cont + 1;

            escreva "Deseja cadastrar outro funcionário? (S/N): ";
            leia resp;
      fim enquanto

      resp ← 'n';
      escreva "Deseja visualizar o(s) cadastro(s) realizado(s)? (S/N):
";
      leia resp;

      se (resp = 's') então
            aux_cont ← 0;

            enquanto (aux_cont < cont) faça
                  escreva "=== Dados do funcionário ===";
                  escreva "Código: ", func[aux_cont].codigo;
                  escreva "Nome : ", func[aux_cont].nome;
                  escreva "Data de Admissão: ",
func[aux_cont].dataAdmissao;
                  escreva "Data de Nascimento: ",
func[aux_cont].dataNascimento;
                  escreva "Sexo : ", func[aux_cont].sexo;

                  aux_cont ← aux_cont + 1;
            fim enquanto
      fim se
fim
```

O algoritmo acima descrito em linguagem C, abaixo:

```
#include <cstdlib>
#include <iostream>

using namespace std;

struct funcionario {
       int codigo;
       char nome[50];
       char dataAdmissao[11];
       char dataNascimento[11];
       char sexo;
};

int main(int argc, char *argv[])
{
    struct funcionario func[1000];
    char resp;
     int cont, aux_cont;

    cont = 0;
    resp = 's';
```

```
    while ((cont < 1000) && ((resp == 's') || (resp == 'S'))) {
            cout << "\n\nDigite o código do funcionário: ";
            cin >> func[cont].codigo;
            fflush(stdin);
            cout << "Digite o nome do funcionário: ";
            gets (func[cont].nome);
            cout << "Digite a data de admissão do funcionário: ";
            gets (func[cont].dataAdmissao);
            cout << "Digite a data de nascimento do funcionário: ";
            gets (func[cont].dataNascimento);
            cout << "Digite o sexo do funcionário (M/F): ";
            cin >> func[cont].sexo;
            fflush(stdin);

            cont++;   /* cont = cont + 1; */

          cout << "\n\nDeseja cadastrar outro funcionário? (S/N): ";
            cin >> resp;
    }

    resp = 'n';
    cout << "\n\nDeseja visualizar os dados cadastrados? (S/N): ";
    cin >> resp;
    fflush(stdin);

    if ((resp == 's') || (resp == 'S')) {
        aux_cont = 0;

        while (aux_cont < cont) {
                cout << "\n\n=== Dados do funcionário ===";
                cout << "\nCódigo: " << func[aux_cont].codigo;
                cout << "\nNome : " << func[aux_cont].nome;
                cout << "\nData de Admissão: " <<
func[aux_cont].dataAdmissao;
                cout << "\nData de Nascimento: " <<
func[aux_cont].dataNascimento;
                cout << "\nSexo : " << func[aux_cont].sexo << "\n\n";

                aux_cont++;
        }
    }

    system("PAUSE");
    return EXIT_SUCCESS;
}
```

4.11 Algoritmo usando REGISTRO em MATRIZ

Elabore um algoritmo para cadastrar 1200 funcionários numa matriz 30 x 40. Para cada funcionário devem-se cadastrar os seguintes dados: código, nome, data de admissão, data de nascimento e sexo. Após cadastrar o(s) funcionário(s), mostrar os dados.

```
tipo funcionario
      codigo : inteiro;
      nome : texto;
      dataAdmissao : texto;
      dataNascimento : texto;
      sexo : carácter;
fim tipo;

início
      declare func[30][40] : funcionario;
      declare resp : carácter;
      declare x, y, aux_x, aux_y : inteiro;

      resp ← 's';
      y ← 0;
      enquanto ((y < 30) e (resp = 's')) então
            x ← 0;
            enquanto ((x < 40) e (resp = 's')) faça
                  escreva "Digite o código do funcionário: ";
                  leia func[y][x].codigo;
                  escreva "Digite o nome do funcionário: ";
                  leia func[y][x].nome;
                  escreva "Digite a data de admissão do funcionário: ";
                  leia func[y][x].dataAdmissao;
                  escreva "Digite a data de nascimento do funcionário:
";
                  leia func[y][x].dataNascimento;
                  escreva "Digite o sexo do funcionário (M/F): ";
                  leia func[y][x].sexo;

                  x ← x + 1;

                  escreva "Deseja cadastrar outro funcionário? (S/N):
";
                  leia resp;
            fim enquanto

            y ← y + 1;
      fim enquanto

      resp ← 'n';
      escreva "Deseja visualizar o(s) cadastro(s) realizado(s)? (S/N):
";
      leia resp;

      se (resp = 's') então
            aux_y ← 0;
            enquanto (aux_y < y) faça
                  aux_x ← 0;
                  enquanto (aux_x < x) faça
                        escreva "=== Dados do funcionário ===";
                        escreva "Código: ", func[aux_y][aux_x].codigo;
                        escreva "Nome : ", func[aux_y][aux_x].nome;
```

```
                        escreva "Data de Admissão: ",
func[aux_y][aux_x].dataAdmissao;
                        escreva "Data de Nascimento: ",
func[aux_y][aux_x].dataNascimento;
                        escreva "Sexo : ", func[aux_y][aux_x].sexo;

                        aux_x ← aux_x + 1;
                  fim enquanto

                  aux_y ← aux_y + 1;
             fim enquanto
      fim se
fim
```

O algoritmo acima descrito em linguagem C, abaixo:

```
#include <cstdlib>
#include <iostream>

using namespace std;

struct funcionario {
       int codigo;
       char nome[50];
       char dataAdmissao[11];
       char dataNascimento[11];
       char sexo;
};

int main(int argc, char *argv[])
{
    struct funcionario func[30][40];
    char resp;
    int x, y, aux_x, aux_y;

    resp = 's';
    y = 0;
    while ((y  < 30) && ((resp == 's') || (resp == 'S'))) {
          x = 0;
          while ((x < 40) && ((resp == 's') || (resp == 'S'))) {
                 cout << "\n\nDigite o código do funcionário: ";
                 cin >> func[y][x].codigo;
                 fflush(stdin);
                 cout << "Digite o nome do funcionário: ";
                 gets (func[y][x].nome);
                 cout << "Digite a data de admissão do funcionário:
";
                 gets (func[y][x].dataAdmissao);
                 cout << "Digite a data de nascimento do funcionário:
";
                 gets (func[y][x].dataNascimento);
                 cout << "Digite o sexo do funcionário (M/F): ";
                 cin >> func[y][x].sexo;
                 fflush(stdin);
```

```
                x++;   /* x = x + 1; */

               cout << "\n\nDeseja cadastrar outro funcionário?
(S/N): ";
                cin >> resp;
          }

          y++;
    }

    resp = 'n';
     cout << "\n\nDeseja visualizar os dados cadastrados? (S/N): ";
     cin >> resp;
     fflush(stdin);

     if ((resp == 's') || (resp == 'S')) {
        aux_y = 0;
        while (aux_y < y) {
              aux_x = 0;
              while (aux_x < x) {
                     cout << "\n\n=== Dados do funcionário ===";
                     cout << "\nCódigo: " <<
func[aux_y][aux_x].codigo;
                     cout << "\nNome : " << func[aux_y][aux_x].nome;
                     cout << "\nData de Admissão: " <<
func[aux_y][aux_x].dataAdmissao;
                     cout << "\nData de Nascimento: " <<
func[aux_y][aux_x].dataNascimento;
                     cout << "\nSexo : " << func[aux_y][aux_x].sexo
<< "\n\n";

                     aux_x++;
              }

              aux_y++;
        }
    }

    system("PAUSE");
    return EXIT_SUCCESS;
}
```

4.12 Algoritmo usando FUNÇÃO

Elabore um algoritmo para simular uma calculadora de 4 operações básicas.

```
função adicao (v1, v2 : real) : real;
     declare total : real;

     total ← v1 + v2;
```

```
     retorne (total);
fim função

função subtracao (v1, v2 : real) : real;
     declare total : real;

     total ← v1 - v2;

     retorne (total);
fim função

função multiplicacao (v1, v2 : real) : real;
     declare total : real;

     total ← v1 * v2;

     retorne (total);
fim função

função divisao (v1, v2 : real) : real;
     declare total : real;

     total ← 0;

     se (v2 <> 0) então
          total ← v1 / v2;
     senão
          escreva "Erro de divisão por zero!";
     fim se

     retorne (total);
fim função

início
     declare valor1, valor2, res : real;
     declare operacao : inteiro;

     escreva "=== Calculadora de 4 Operações Básicas ===";

     escreva "Digite o 1°. valor: ";
     leia valor1;
     escreva "Digite o 2°. valor: ";
     leia valor2;

     escreva "Operação a ser realizada sobre os valores:";
     escreva "1 - Somar";
     escreva "2 - Subtrair";
     escreva "3 - Multiplicar;
     escreva "4 - Dividir;
     escreva "Escolha uma opção: ";
     leia operacao;

     caso (operacao) faça
          1 : início
                    res ← adicao (valor1,valor2);
```

```
                fim
            2 : início
                    res ← subtracao (valor1,valor2);
                fim
            3 : início
                    res ← multiplicacao (valor1,valor2);
                fim
            4 : início
                    res ← divisao (valor1,valor2);
                fim
            exceção : início
                            escreva "Opção escolhida é inválida!";
                      fim
      fim caso

      se ((operacao >= 1) e (operacao <= 4)) então
           escreva "O resultado final é: ", res;
      fim se
fim
```

O algoritmo acima descrito em linguagem C, abaixo:

```
#include <cstdlib>
#include <iostream>

using namespace std;

float adicao (float v1, float v2){
      float total;

      total = v1 + v2;

      return (total);
}

float subtracao (float v1, float v2){
      float total;

      total = v1 - v2;

      return (total);
}

float multiplicacao (float v1, float v2){
      float total;

      total = v1 * v2;

      return (total);
}

float divisao (float v1, float v2){
      float total;
```

```
      total = 0;

      if (v2 != 0) {
         total = v1 / v2;
      }
      else {
           cout << "\n\nErro de divisão por zero!";
      }

      return (total);
}

int main(int argc, char *argv[])
{
    float valor1, valor2, res;
    int operacao;

     cout << "\n=== Calculadora de 4 Operações Básicas ===";

    cout << "\n\nDigite 1o. valor: ";
    cin >> valor1;
    cout << "Digite 2o. valor: ";
    cin >> valor2;

    cout << "\n\nOperação a ser realizada sobre os valores:";
    cout << "\n\n1 - Somar";
    cout << "\n2 - Subtrair";
    cout << "\n3 - Multiplicar";
    cout << "\n4 - Dividir";
    cout << "\n\nEscolha uma opção: ";
    cin >> operacao;

    switch (operacao) {
        case 1:
             res = adicao (valor1,valor2);
             break;
        case 2:
             res = subtracao (valor1,valor2);
             break;
        case 3:
             res = multiplicacao (valor1,valor2);
             break;
        case 4:
             res = divisao (valor1,valor2);
             break;
        default:
             cout << " \n\nOpção escolhida é inválida!\n\n";
    }

    if ((operacao >= 1) && (operacao <= 4)) {
        cout << "\n\nO resultado final é: " << res << "\n\n";
    }

    system("PAUSE");
    return EXIT_SUCCESS;
```

```
}
```