Programação I
Notas de Aula

Ricardo Dorneles

CCET- UCS

06 de março de 2018

Índice

Ementa

- Resolução de problemas computacionais utilizando uma linguagem de programação. Caracterização de tipos de dados e variáveis. Estudo e emprego de estruturas de controle condicionais e de repetição. Modularização e reutilização de código. Representação de dados utilizando estruturas unidimensionais de dados homogêneos.

Algoritmo

- Um **algoritmo** é uma lista ordenada de comandos que, quando executados, resultam em uma sequencia de ações, que visam resolver um determinado problema.
- Ex:????
- Algoritmos podem ser descritos de diversas formas, gráficas ou textuais.
- Quando um algoritmo é escrito em uma **linguagem de programação**, é chamado de **programa**.
  - De quantas linguagens de programação você já ouviu o nome?

- Para a construção de qualquer tipo de algoritmo, é necessário seguir estes passos:
  1. Compreender completamente o problema a ser resolvido, destacando os pontos mais importantes e os objetos que o compõem.
  2. Definir os dados de entrada, ou seja, quais dados serão fornecidos e quais objetos fazem parte desse cenário-problema.
  3. Definir o processamento, ou seja, quais cálculos serão efetuados e quais as restrições para esses cálculos. O processamento é responsável pela transformação dos dados de entrada em dados de saída. Além disso, deve-se verificar quais objetos são responsáveis pelas atividades.
  4. Definir os dados de saída, ou seja, quais dados serão gerados depois do processamento.
  5. Construir o algoritmo utilizando um formalismo conhecido.
  6. Testar o algoritmo realizando simulações.

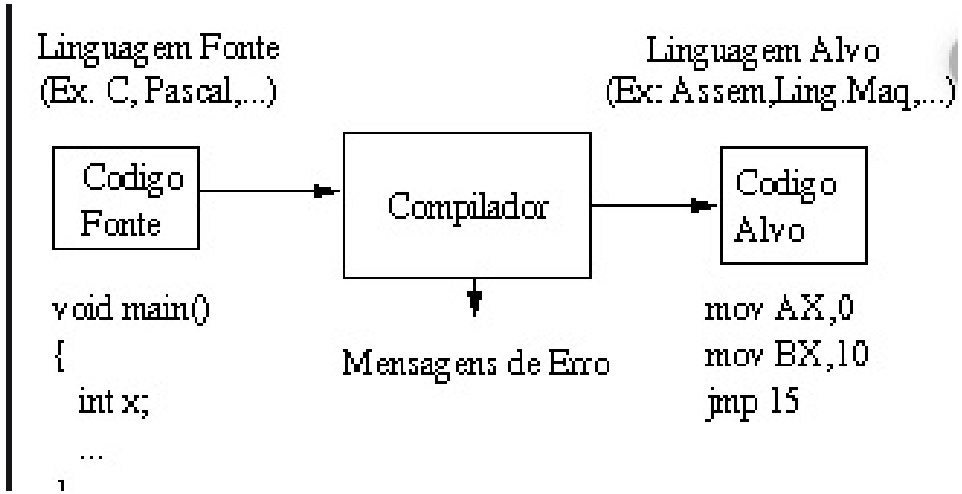
Linguagem C

- Nessa disciplina será utilizada a linguagem de programação C. Ela foi desenvolvida em 1972 por Denis Ritchie, e ainda é a linguagem mais usada na programação de microcontroladores e na indústria.
- Como linguagem de uso geral, é a segunda linguagem mais utilizada no mundo, perdendo apenas para Python.

Voltar para o índice

Compilação

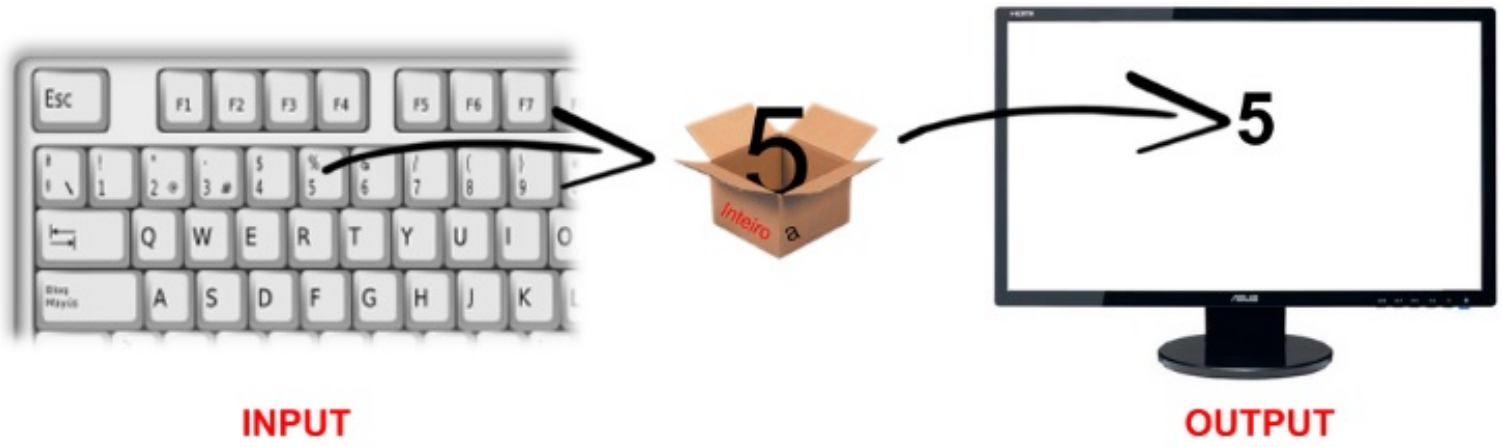
- Para que um programa escrito em uma linguagem de programação (dito "alto nível") possa ser executado, ele deve ser traduzido para uma "linguagem de máquina"
- Essa função é executada por um programa chamado **compilador**, normalmente associado a um ambiente de desenvolvimento (IDE - Integrated Development Environment) que contem editor e recursos para execução do programa



Compilação

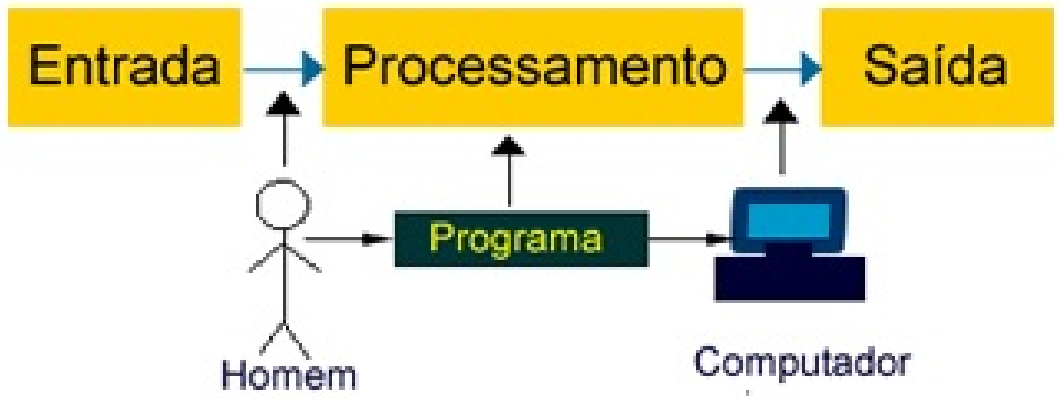
- Nessa disciplina utilizaremos as seguintes IDEs:
  - Dev C++ - Não é o melhor compilador do mundo, mas é de graça.
  - Visual Studio - É ótimo, mas meio complicado para iniciantes
  - WebAlgo (Portal ProCê) - O ambiente utilizado em algoritmos, agora em versão beta para linguagem C

- Programas de computador basicamente recebem dados, processam-nos e emitem resultados.
- Esses dados podem vir de várias fontes:
  - Teclado
  - Sensores
  - Arquivos
- E o resultado pode ser enviada para vários lugares:
  - Monitor
  - Atuadores (ligar motores, acender luzes, emitir sons)



Operação de entrada de dados (input)

- Nessa disciplina, nossos programas receberão dados pelo teclado, digitados pelo usuário, e os resultados serão escritos na tela



- Para a construção de qualquer tipo de algoritmo, é necessário seguir estes passos:
  1. Compreender completamente o problema a ser resolvido, destacando os pontos mais importantes e os objetos que o compõem.
  2. Definir os dados de entrada, ou seja, quais dados serão fornecidos e quais objetos fazem parte desse cenário-problema.
  3. Definir o processamento, ou seja, quais cálculos serão efetuados e quais as restrições para esses cálculos. O processamento é responsável pela transformação dos dados de entrada em dados de saída. Além disso, deve-se verificar quais objetos são responsáveis pelas atividades.
  4. Definir os dados de saída, ou seja, quais dados serão gerados depois do processamento.
  5. Construir o algoritmo utilizando um formalismo conhecido.
  6. Testar o algoritmo realizando simulações.

- Por exemplo: Suponha um programa para calcular o volume de um cilindro de raio r e altura h.
- Quais os dados de entrada?
- Que cálculos devem ser realizados
- Qual a saída esperada? O que o programa deve escrever?
- O que seria um exemplo de entrada e saída esperada para a entrada?

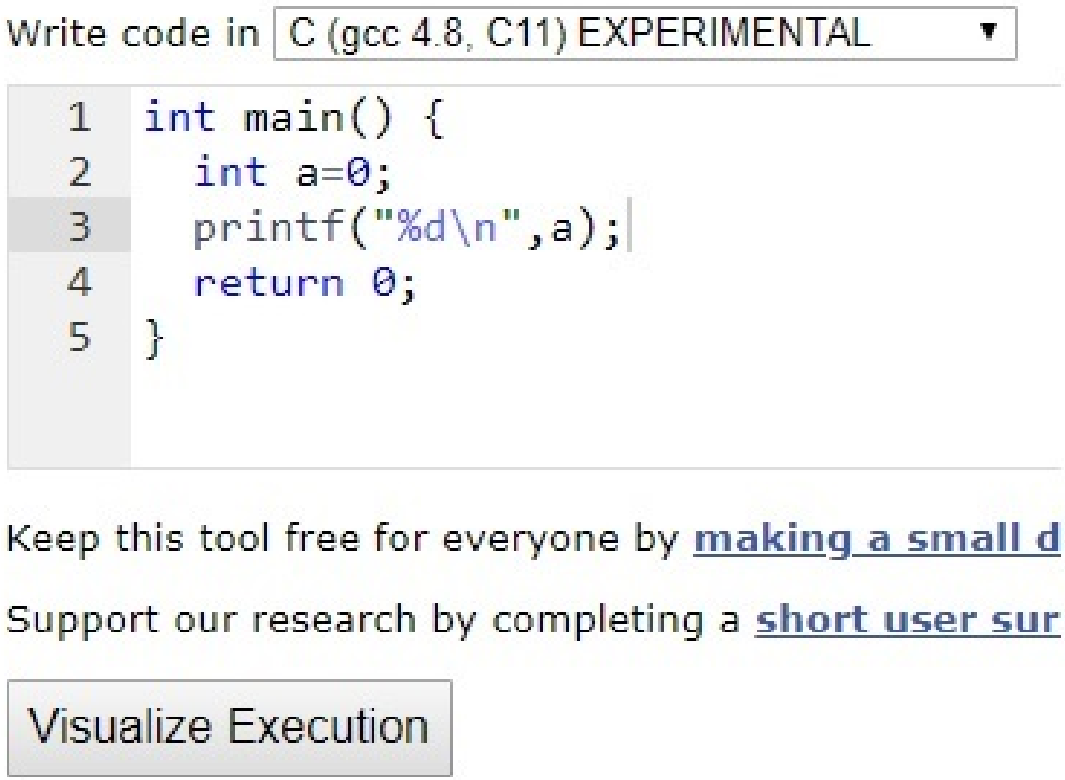
- Iniciaremos o uso do WebAlgo executando um programa simples.
- Considere o programa em C abaixo:

```
int main( )
{
int a,b,soma; /* declaração de variáveis */
a=3;
b=5;
soma=a+b;
}
```

- Baixe do acervo da turma o arquivo WebAlgo 2.3 e execute-o
- Selecione no menu Preferências ->"Configurar preferências do usuário", a opção "Linguagem de Programação C"
- No menu Arquivo, selecione a opção Novo
- Digite o programa acima
- Execute o programa passo-a-passo (a cada clicada no botão Passo, uma instrução será executada

Ferramenta fortemente recomendada

- http://pythontutor.com/ → Start visualizing your code now
- Selecionar linguagem
- Editar código (p.ex. código abaixo)
- Visualize execution



- Ao clicar em "visualize execution" é feita uma simulação da execução do código (aparece uma mensagem pedindo para esperar 10 segundos) e é gerada uma sequência de frames sobre a qual é exibida a simulação.
  - Eventualmente o servidor pode estar sobrecarregado, ou a duração da simulação excede o limite de frames permitido, e não será possível fazer a simulação.
- Como a exibição é feita após a simulação, não é possível interação com o usuário (ou seja, não há comandos de entrada (scanf))
- Dados de teste podem ser inseridos na declaração de variáveis. Ex na lâmina seguinte:

.

C (gcc 4.8, C11) EXPERIMENTAL!
see known bugs and report to philip@pgbovine.net

```
1  #include <stdio.h>
2  typedef struct nodo{int val;struct nodo *prox;} Tnodo;
3  int main() {
4  int m[10]={9,8,7,6,5,4,3,2,1,0},i,j,aux;
5  for (j=0;j<9;j++)
6  for (i=0;i<9;i++)
7     if (m[i]>m[i+1])
8     {
9       aux=m[i];
10     m[i]=m[i+1];
11     m[i+1]=aux;
12    }
13   return 0;
14 }
```

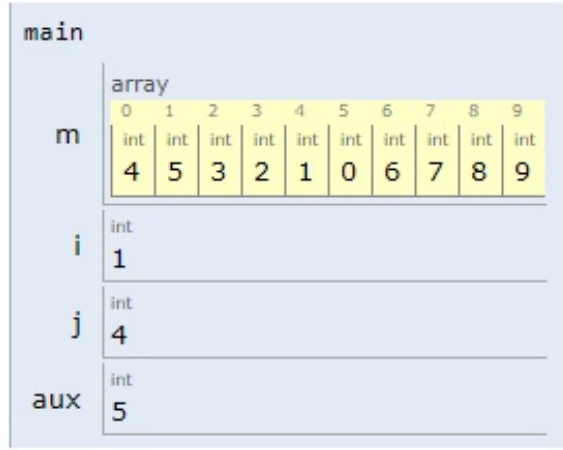
Edit code

line that has just executed
next line to execute

Click a line of code to set a breakpoint; use the Back and Forward buttons to jump there.

<< First | < Back | Step 179 of 318 | Forward > | Last >>

Stack



- Os botões First e Last permitem ir até o primeiro e o último frame da simulação.
- Os botões Back e Forward permitem avançar ou voltar um frame.
- É possível setar pontos de parada no código para que, ao chegar no ponto selecionado, a execução seja pausada.
  - Isso é feito clicando uma vez sobre a linha a ser inserido o break point. A linha ficará em vermelho.
  - Ao clicar em Forward, ao passar sobre o ponto de parada a simulação será interrompida.
  - Ao clicar em Last os breakpoints serão ignorados
  - Para remover o breakpoint, basta clicar novamente sobre a linha.

E agora, uma rápida explicação de cada um dos elementos do programa:
1) Inclusão de bibliotecas

```
#include <stdio.h>
```

- A linguagem C possui várias bibliotecas (conjuntos de funções prontas).
- Bibliotecas para leitura e escrita, matemáticas, para manipulação de textos, etc.
- Para usar funções de uma biblioteca, é necessário informar no início do programa quais bibliotecas serão utilizadas.
- Isso é feito com a cláusula #include <nome da biblioteca>.
- No nosso programa, a biblioteca stdio.h possue as funções de leitura e escrita (printf e scanf).

2) Bloco do programa principal (Função main)

```
int main( )
{

}
```

Um programa em C é formado por um conjunto de blocos (chamados funções). Um dos blocos é chamado de main e é por onde inicia a execução do programa. O formato de cada bloco é

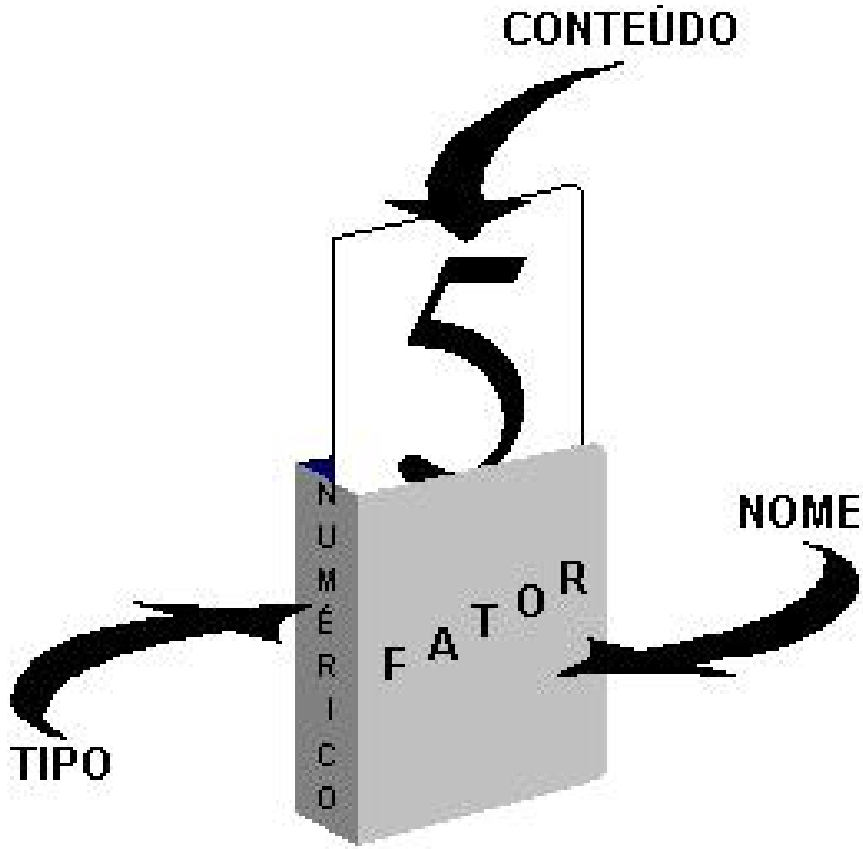
```
tipo_de_valor_retornado_pelo_bloco  nome_da_função  (parâmetros)
{
}
```

Variáveis

- São os elementos de armazenamento de dados em um programa.
- Uma variável corresponde fisicamente a uma posição de memória do computador.
- Ela contem sempre um (1) valor.
- Ao receber um novo valor, o valor anterior é substituído.
- Uma variável é identificada por um nome, chamado **identificador** da variável.
- Um identificador de variável deve iniciar por letra e só pode conter letras, dígitos e o caracter "sublinha". E não pode conter caracteres acentuados ou cedilha.
- E não pode ser nenhuma palavra reservada da linguagem (p.ex. **printf**, **scanf**, **void**, **int**...)

Voltar para o índice

- Uma variável pode ser vista como uma caixinha identificada por um nome e que pode armazenar valores de um tipo definido (por exemplo: inteiro ou real)



Voltar para o índice

Comando de Atribuição

- O comando utilizado para alterar o valor de uma variável chama-se comando de atribuição e tem o formato:
- variável = expressão; (leia-se "variável RECEBE expressão")
- Ex:

```
a = 1;
a = a+1;
a = a+b;
```

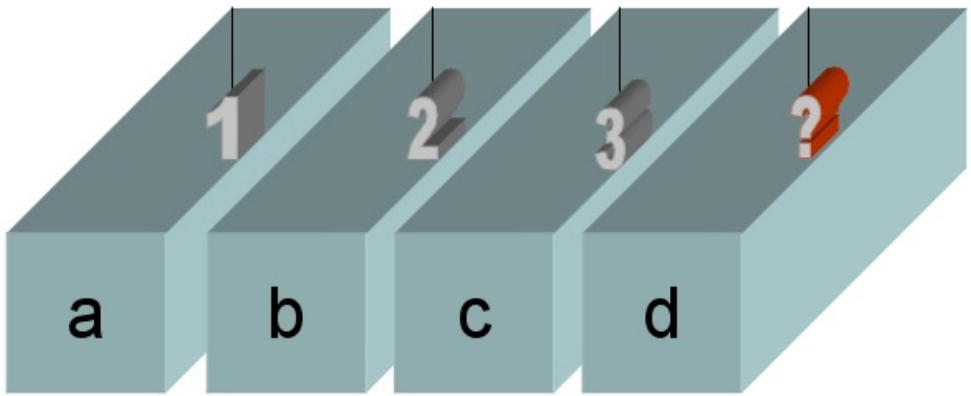
- Na atribuição, quando uma variável recebe uma expressão, o valor da expressão é calculado e o resultado é atribuído à variável.
- Ao utilizar constantes reais em expressões, o ponto (.) é utilizado como separador das casas decimais (ex: 3.14)

Voltar para o índice

- Em um programa que contenha as variáveis **a**, **b**, **c** e **d**, após a sequência de atribuições a seguir:

  a = 1;
  b = 2;
  c = 3;

- as variáveis terão os seguintes valores (o "?" em **d** significa que **d** ainda não tem um valor definido ou, em outras palavras, contem "lixo"):



Voltar para o índice

Ex: Qual o valor das variáveis a e b ao final da execução do código abaixo?

```
a = 1;
b = 2;
a = a+b;
b = a+b;
a = a+b;
b = a+b;
a = a+b;
b = a+b;
escreva (a,b)
```

Voltar para o índice

3) Declarações de variáveis

- As declarações de variáveis em C tem o formato:
  - tipo lista_de_variáveis;
- E os principais tipos são **int** (variáveis inteiras) e **float** (variáveis reais). Ex:
  - int c1,b1,cont;
  - float peso, media;
- Da mesma forma que em algoritmos, nomes de variáveis devem iniciar por letra e podem conter letras, dígitos e o caracter de sublinha (_).

4) Comentários

- Comentários podem ser escritos no código para melhorar sua clareza. Comentários são formados por qualquer sequência de caracteres delimitados por /* ..... */
- Os compiladores da linguagem C normalmente aceitam (apesar de não ser parte do C padrão) também comentários de linha, que iniciam por // e se estendem até o fim da linha
- Ex:

```
int a,b,soma; /* Sou um comentário!!! */
printf("Digite um valor:"); // Eu também!!!
```

- Considere o programa em C abaixo:

```
#include <stdio.h>
// 04/08/2017
int main( )
{
int lado,area; /* declaração de variáveis */
printf("Digite␣o␣comprimento␣do␣lado:");
scanf("%d",&lado);
area=lado*lado;
printf("A␣área␣é␣%d\n",area);
}
```

5) Comando de escrita: printf( )

- O comando para escrever coisas na tela é a função printf(), presente na biblioteca stdio.h (não esqueça o #include <stdio.h>
- A função printf( ) tem como primeiro parâmetro a mensagem a ser mostrada na tela.
  - Ex: printf("Oi");
- Se essa mensagem deve conter o valor de variáveis ou o resultado de expressões, as variáveis e expressões são listadas imediatamente após a mensagem. Ex:

```
a=3;
b=5;
printf("A soma de %d e %d vale %d",a,b,a+b);
```

- O exemplo anterior mostrará a mensagem "A soma de 3 e 5 vale 8".
- Para exibir valores de variáveis e expressões, a mensagem deve conter uma indicação do local na mensagem onde serão inseridos os valores das variáveis e expressões.
- Esses indicadores de posição tem o formato %d, se o valor a ser inserido for inteiro, e %f se o valor a ser inserido for float.
- Os indicadores de posição devem ser em mesmo número dos elementos a serem inseridos, e na mesma ordem.
- Um erro comum é utilizar o especificador %f para escrever valores inteiros ou %d para valores reais. O Dev C++ não identifica esse erro.

- No especificador de formato pode-se especificar também o número de casas (total e após a vírgula), se o valor será alinhado à esquerda ou à direita e se o espaço previsto será preenchido ou não com 0s à esquerda.
- Alguns exemplos são:
  - printf("%5d",x); - imprime o valor de x utilizando 5 espaços, alinhado à direita, preenchendo com espaços em branco à esquerda.
  - printf("%05d",x); - imprime o valor de x utilizando 5 espaços, alinhado à direita, preenchendo com zeros à esquerda.
  - printf("%5.3f',fx); - imprime o valor de fx utilizando 5 espaços no total, dos quais 3 após o ponto decimal, alinhado à direita, preenchendo com espaços em branco à esquerda.
- se o número de dígitos a ser mostrado exceder o tamanho definido para o campo, a informação de tamanho de campo é ignorada.

Caracteres especiais

- Também podem ser utilizados caracteres de formatação na mensagem. O principal caracter é o $\backslash n$ que causa uma quebra de linha no ponto onde foi inserido.
- O caracter $\backslash t$ imprime um caracter de tabulação
- O caracter $\backslash r$ posiciona o cursor no início da linha

- Códigos para formatação de impressão das expressões que devem ser passadas para o especificador de formato:

  %c - caractere simples
  %d ou %i - inteiro decimal com sinal
  %e ou %E - notação científica
  %f - ponto flutuante
  %g ou %G - %e ou %f (o mais curto!)
  %o - octal
  %s - cadeia de caracteres
  %u - inteiro decimal sem sinal
  %x ou %X - hexadecimal
  %p - ponteiro (endereço)
  %n - ponteiro (inteiro)
  %% - escreve o símbolo %

5) Comando de leitura: scanf( )

- O comando utilizado para receber dados do usuário através do teclado é o comando scanf( ).
- Os parâmetros do scanf são um especificador dos tipos dos valores que serão lidos, seguido da lista de variáveis que receberão os valores, sendo que o identificador de cada variável deve ser acompanhado de &.
- Ex:

```
scanf("%d%f",&a,&p);
```

- O exemplo acima recebe 2 valores do teclado, sendo o primeiro um valor inteiro e o segundo um valor float.
- Os dois valores são armazenados respectivamente nas variáveis a e p.
- Apenas os especificadores dos tipos (%f, %d, etc.)devem ir no primeiro parâmetro. Nada de espaços em branco ou outros caracteres. Coisas horríveis podem ocorrer se forem colocados outros caracteres.

Especificadores de formato

Especificador	Descrição	Caracteres lidos
i	inteiro	Qualquer quantidade de dígitos, opcionalmente precedidos por + ou -. Por padrão os dígitos são decimais (0 a 9), mas um prefixo 0 indica que são caracteres octais (0 a 7) e um prefixo 0x indica que são caracteres hexadecimais (0 a F).
d ou u	inteiro decimal	Qualquer quantidade de dígitos decimais (0-9), opcionalmente precedidos por sinal (+ ou -). d é para argumento com sinal, u para sem sinal
o	inteiro octal	Qualquer quantidade de dígitos octais (0-7), opcionalmente precedidos por sinal (+ ou -).
x	int hexadecimal	Qualquer quantidade de dígitos hexadecimais (0-9,a-f,A-F), opcionalmente precedidos por 0x ou 0X, e opcionalmente precedidos por sinal (+ ou -).
f, e, g,a	float	Uma série de dígitos decimais, opcionalmente contendo um ponto decimal, opcionalmente precedido de sinal (+ ou -) e opcionalmente seguido por e ou E e um número decimal. Algumas implementações suportam floats em hexadecimal, quando precedidos por 0x ou 0X.

Especificadores de formato (continuação)

Especificador	Descrição	Caracteres lidos
c	caracter	Se um comprimento maior que 1 é especificado, a função le a quantidade de caracteres especificada e armazena nas primeiras posições do vetor passado como parâmetro, sem acrescenter o \0 ao final.
s	string de caracteres	Qualquer quantidade de caracteres exceto o espaço em branco. Um \0 é automaticamente adicionado ao final da sequencia.
p	Pointer	Uma sequencia de caracteres representando um endereço de memória. O formato específico depende do sistem usado, mas é o mesmo usado no %p no printf.
[caracteres]	sequencia de caracteres	Qualquer número de caracteres dentre os especificados entre os colchetes
[^ caracteres]	sequencia de caracteres	Qualquer número de caracteres quaisquer, exceto os especificados entre os colchetes
% %	%	Dois % seguidos representam um %

6) Comando de atribuição:

- Tem o formato:

  identificador_da_variável = expressão;

- Os principais operadores aritméticos da linguagem C são o de adição (+), subtração (-), multiplicação (*), divisão (/).
- É importante, ao escrever uma expressão, considerar os tipo de operandos e o tipo do resultado da operação.
- Os 4 operadores resultam int se ambos os operandos são int, caso contrário, resultam float.
  - **Atenção** para o caso do operador de divisão. Quanto resulta a expressão 3/2? E 3.0/2?

- É importante também considerar a precedência (prioridade) dos operadores.
  - Os operadores *, / e % tem a mesma prioridade entre si, que é maior que a prioridade dos operadores + e - (subtração).
- O operador "-" também é utilizado como operador de troca de sinal, e nesse caso tem uma prioridade maior que todos os outros.
- A tabela a seguir resume os principais operadores aritméticos em C:

Operação	Operador	Prioridade
Troca de Sinal	-	1
Multiplicação	*	2
Divisão	/	2
Resto da divisão inteira	%	2
Soma	+	3
Subtração	-	3

- Observações:
  - O operador de troca de sinal é o único operador de apenas um operando
  - O operador de resto de divisão inteira só é aplicável a operandos inteiros

- Parênteses são utilizados para mudar a ordem de avaliação dos operadores em uma expressão.
- A linguagem C faz conversão automática de tipos.
  - Pode-se atribuir um int para uma variável float e vice-versa. Ao atribuir um float para um int, as casas decimais são truncadas.

Faça os programas a seguir:

1. Faça um programa que leia 3 valores reais, notas de um aluno, e escreva sua média aritmética. A média aritmética de um conjunto de valores é dada pela soma dos valores dividido pela quantidade de valores considerados.
2. Faça um programa que leia 2 valores reais v1 e v2 e calcule e escreva a área do triângulo que tem base igual a v1 e altura igual a v2. Dica:A área de um triângulo é dada pela expressão: (base x altura)/2

- Baixe do UCS Virtual / Acervo da Turma o programa webalgo.jar
- Execute-o
- Configure-o no menu Preferências - Configurar preferência do usuário - Linguagem de Programação, para trabalhar com a linguagem C.
- Se você ainda não tem conta no WebAlgo, crie uma conta (menu Usuário - Criar usuário)
  - Mesmo que você tenha conta pode ser interessante criar uma segunda conta, para manter separadas as soluções em C e em Português estruturado
- Resolva os 10 primeiros exercícios de Algoritmos Sequenciais.

Voltar para o índice

Principais funções da Biblioteca Math (#include <math.h>)

- Todas elas retornam valores tipo float.
  - pow (x,y) – Calcula x elevado a y.
  - sqrt(x) – Calcula a raiz quadrada de x.
  - fabs (x) – Calcula o valor absoluto de x.
- Ex:

  a = pow(3,2);
  a = sqrt(b);

S00000300) Faça um programa que leia 3 valores a, b e c, coeficientes de uma equação de segundo grau, e calcule e escreva as raizes da equação. As raízes de uma equação podem ser calculadas pela fórmula de Baskhara:

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- Teste seu programa com os coeficientes

  a=2

  b=5

  c=2

- que deve resultar em raízes x_1=-2.0 e x_2=-0.5

S00000400) Faça um programa que leia 2 valores e escreva o maior deles. O maior entre dois valores quaisquer v1 e v2 pode ser calculado pela expressão

$$\frac{v_1 + v_2 + |v_1 - v_2|}{2}$$

.

- O operador matemático de módulo ($|x|$) é representado em C pelas funções fabs(x) para números float e abs (x) para números inteiros, da biblioteca math.h
- Ex: |a+b| é representado em C por fabs(a+b)

S00000500) Faça um programa que leia 2 valores e escreva o menor deles.

Dica: O menor entre dois valores pode ser obtido a partir da soma dos dois e do valor do maior.

S00000600) Faça um programa que leia 3 valores escreva o maior deles.

Dica: O maior entre três valores pode ser encontrado pela aplicação repetida da expressão para encontrar o maior entre dois valores.

- S00000900) Faça um programa que lê 3 valores **a**, **b** e **c**, lados de um triângulo, e calcule e escreva a área do triângulo formado. A área de um triângulo de lados **a**, **b** e **c** pode ser calculada pela expressão

$$Area = \sqrt{S * (S - a) * (S - b) * (S - c)}$$

- onde S é o semi-perímetro, ou seja, a metade da soma dos lados ($\frac{a+b+c}{2}$).

Operador de resto da divisão inteira

- Além dos operadores vistos (+,-,*,/), a linguagem C possui também o operador %, que resulta no RESTO da divisão inteira do primeiro operando pelo segundo. Ex: x = 5 % 2; x receberá 1, que é o resto da divisão inteira de 5 por 2.
- O operador de resto de divisão inteira é frequentemente usado em cálculos que envolvem conversões, como conversão de segundos para minutos e segundos. Quantos minutos e segundos há em 200 segundos?
- minutos = segundos / 60;
- segundos = segundos % 60; // segundos que sobram ao retirar os minutos

- S00000700) Faça um programa que lê um valor inteiro em reais e calcula e escreve qual o menor número possível de notas de 100,50,20,10,5,2 e 1 real em que o valor pode ser decomposto. Dica:Isso pode ser calculado a partir de operações de divisão inteira.

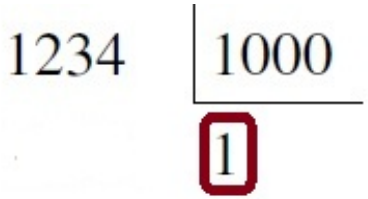
	375 reais	100
		3 notas de 100
e sobra	75 reais	50
		1 nota de 50
e sobra	25 reais	20
		1 nota de 20
e sobra	5 reais	10
		0 notas de 10
e sobra	5 reais	5
		1 nota de 5
e sobra	0 reais	2
		0 notas de 2
e sobra	0 notas de 1	

```
scanf("%d",&valor);
Notas100 = valor / 100;
valor = valor % 100;
Notas50 = valor / 50;
valor = valor % 50;
...
```

- S00000800) Faça um programa que lê uma quantia inteira em segundos e escreva o número de horas, minutos e segundos correspondente.

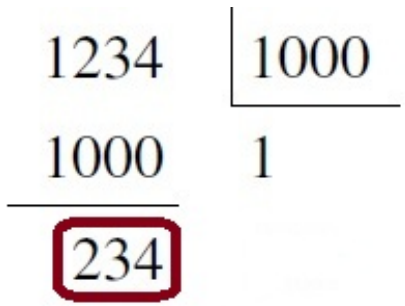
- S00001000) Faça um programa que le um valor entre 0 e 9999 e calcula a soma dos seus dígitos. O dígito menos significativo de um número inteiro pode ser obtido pelo resto da divisão do número por 10. Os dígitos restantes podem ser obtidos pela divisão inteira por 10.

Ex: Separação de dígitos



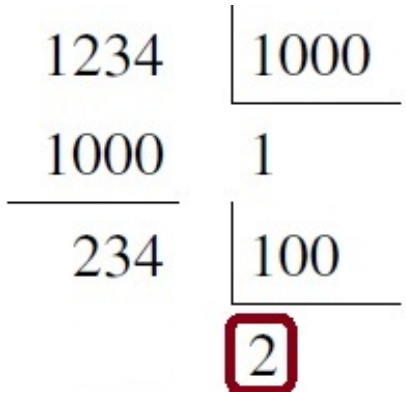

```
scanf("%d",&valor);
milhar = valor / 1000;
```

Ex: Separação de dígitos



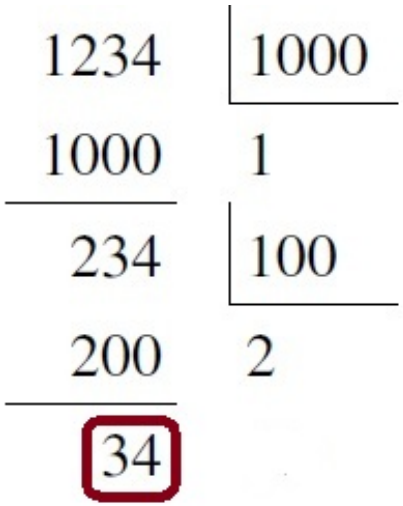

```
scanf("%d",&valor);
milhar = valor / 1000;
valor = valor % 1000;
```

Ex: Separação de dígitos



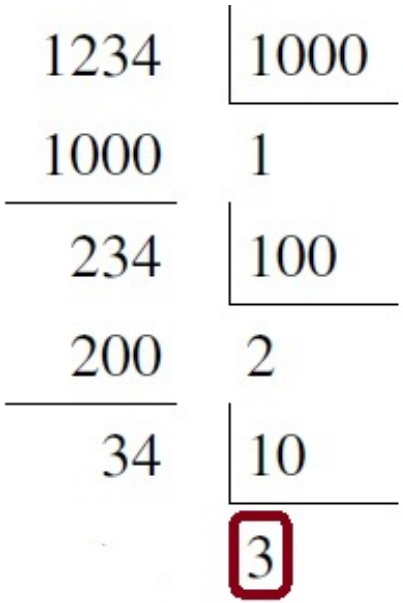

```
scanf("%d",&valor);
milhar = valor / 1000;
valor = valor % 1000;
centena = valor / 100;
```

Ex: Separação de dígitos



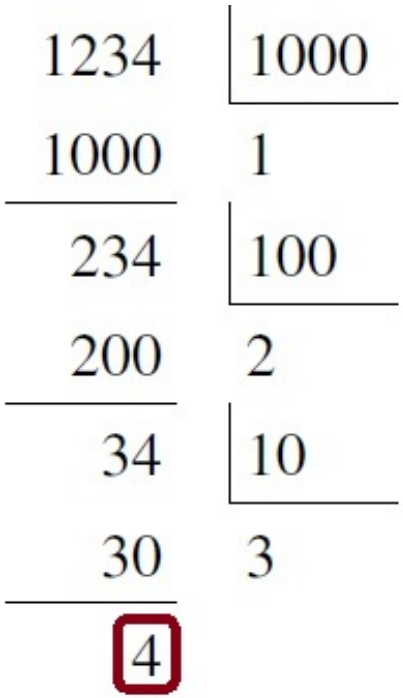

```
scanf("%d",&valor);
milhar = valor / 1000;
valor = valor % 1000;
centena = valor / 100;
valor = valor % 100;
```

Ex: Separação de dígitos



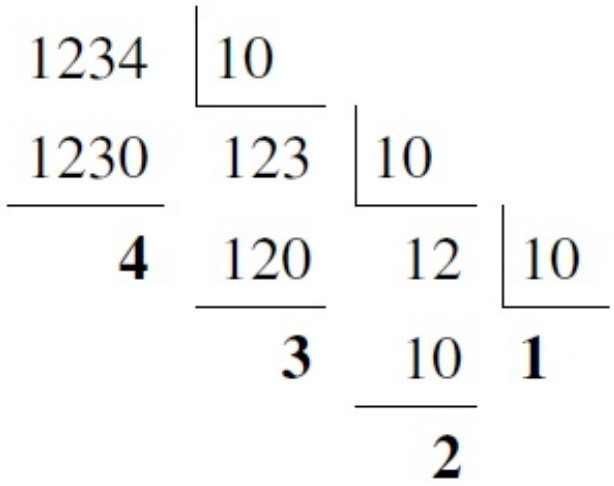

```
scanf("%d",&valor);
milhar = valor / 1000;
valor = valor % 1000;
centena = valor / 100;
valor = valor % 100;
dezena = valor / 10;
```

Ex: Separação de dígitos




```
scanf("%d",&valor);
milhar = valor / 1000;
valor = valor % 1000;
centena = valor / 100;
valor = valor % 100;
dezena = valor / 10;
unidade = valor % 10;
```

Ex: Separação de dígitos (b)



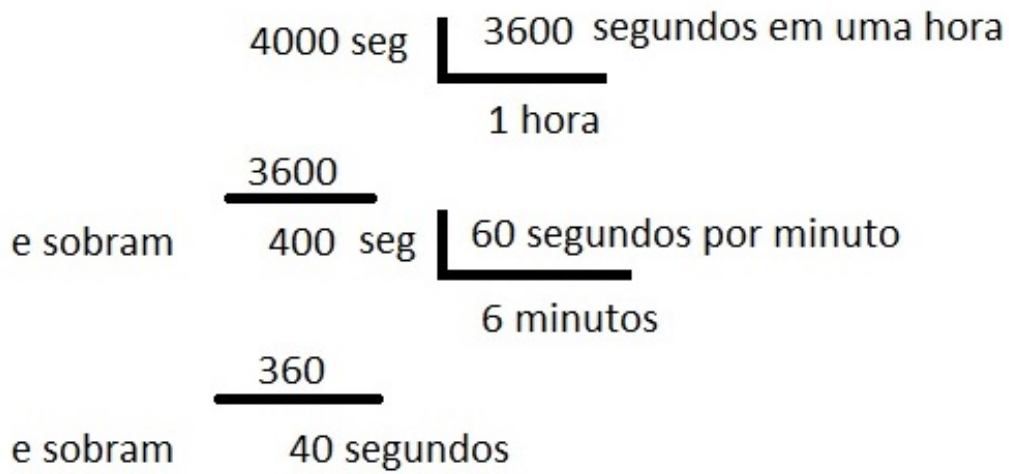

```
scanf("%d",&valor);
unidade = valor % 10;
valor = valor / 10;
dezena = valor % 10;
valor = valor / 10;
centena = valor % 10;
milhar = valor / 10;
```

- 11) Faça um programa que leia 4 valores, H_i, M_i, H_f, M_f, representando respectivamente a hora e minuto inicial e final de um evento, e calcule a duração do mesmo em horas e minutos. Considere que o evento inicia e termina no mesmo dia. Para simplificar o problema, converta cada par de valores em um único valor em minutos.

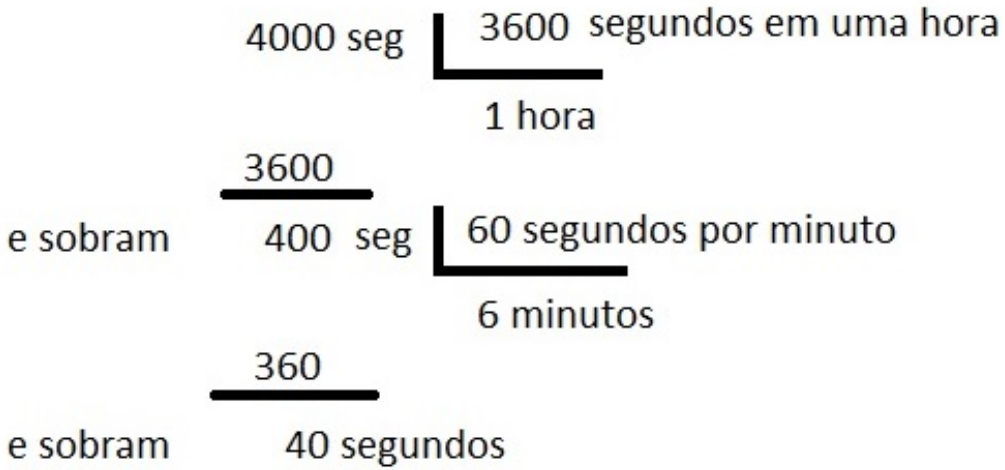
Conversão de segundos para horas, minutos e segundos

- A conversão entre segundos, minutos e horas é feita através de operações de divisão inteira, resto de divisão inteira e multiplicação.
- As principais conversões são:
  - Segundos para minutos: minutos = segundos / 60
  - Segundos para horas: horas = segundos / 3600
  - Minutos para segundos: segundos = minutos * 60
  - Minutos para horas: horas = minutos / 60
  - Horas para minutos: minutos = horas * 60
  - Horas para segundos: segundos = horas * 3600

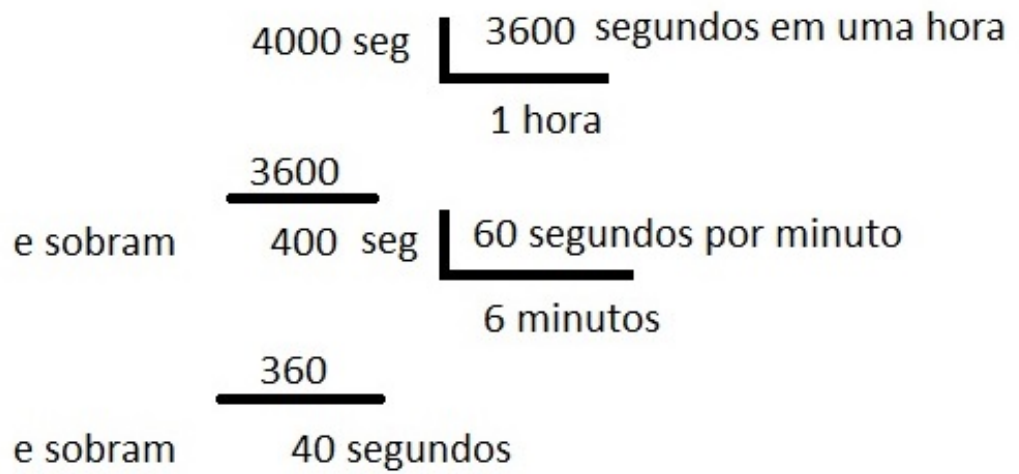
- A figura abaixo mostra a conversão de 4000 segundos para horas, minutos e segundos:



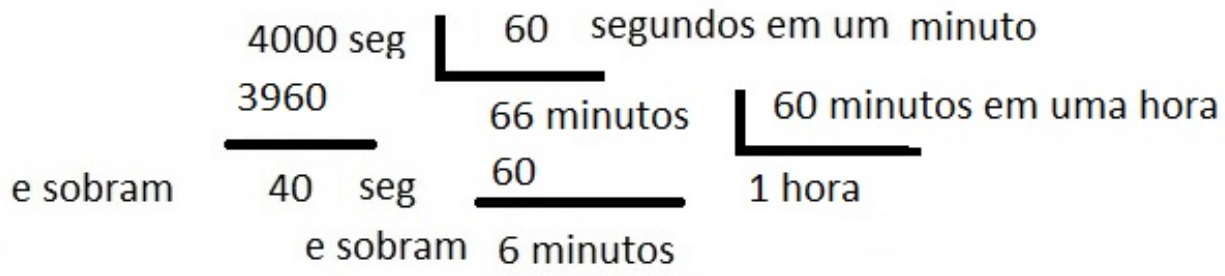
- Observe que o cálculo dos restos das divisões é uma parte importante da conversão, porque após calculado o número de horas, a continuação do cálculo é feita a partir dos segundos que sobraram da divisão (no exemplo, os 400 segundos) e esse valor é obtido a partir do operador de resto.



- O código abaixo mostra como a sequencia de divisões pode ser implementada:
- horas = segundos / 3600;
- segundos = segundos % 3600; // segundos que sobram após // calculadas as horas
- minutos = segundos / 60;
- segundos = segundos % 60;



- A sequência de operações mostrada não é a única solução. Pode-se começar convertendo os segundos para minutos, e em seguida converter esses minutos para horas, como mostrado abaixo:



Exercícios de divisão inteira

- URI Online Judge:
  - 1018 - Cédulas
  - 1019 - Conversão de Tempo
  - 1020 - Idade em Dias
- WebAlgo:
  - S700, S800, S1000, S1200, S1300, S1350, S1400, S1500, S1600, S1650, S1700 e S1800

- S00001300) Faça um programa que leia dois horários (hora, minuto e segundo) e escreva quantos segundos transcorreram entre esses dois horários.
- S00001350) Faça um programa que a partir de um horário (hora, minuto, segundo) e uma quantidade de segundos transcorridos, calcule o segundo horário. Obs: É possível que a hora resultante seja maior que 23, no caso de o horário resultante ocorrer no dia seguinte. Nesse caso pode-se aplicar sobre a hora resultante a operação %24 para remover 24 horas.
- S00001500) Faça um programa que leia a quantidade de alunos em uma sala de aula e a quantidade de alunos por grupo, e calcule e escreva quantos grupos serão formados e o resto de alunos que não foram suficientes para formar mais um grupo.

Biblioteca Locale.h

- Mensagens emitidas pelo printf não reconhecem caracteres acentuados.
- Para escrever mensagens com caracteres acentuados deve-se configurar o conjunto de caracteres a ser utilizado como os caracteres da língua portuguesa.
- Isso é feito no início do main pela chamada de função:

  setlocale (LC_ALL, "Portuguese");
- A função setlocale se encontra na biblioteca locale.h, é para usá-la é necessário o include

  #include <locale.h>

Tipos de variáveis inteiras em C

Tipo	Tamanho	valor mínimo	valor máximo	formatador
char	8 bits	-128	127	%d ou %i
short int	16 bits	-32536	32535	%d ou %i
int	32 bits	-2.000.000.000	2.000.000.000	%d ou %i
long int	32 bits	-2.000.000.000	2.000.000.000	%ld ou %li
long long int	64 bits	- 2e20	2e20	%lld ou %lli

- Observações:
  - Apesar do nome ("char" de "caracter"), variáveis do tipo char são utilizadas como inteiros de 8 bits.
  - "short" e "long" são considerados "modificadores" de tipos, mas em versões mais recentes de compiladores elas são suficientes para especificar o tipo, podendo ser omitida a palavra "int".
    - Ex: a declaração "long long int way" poderia ser escrita como "long long way"

Tipos de variáveis reais em C

- Variáveis do tipo float tem uma precisão aproximada de 7 dígitos decimais.
- Para situações em que se necessita maior precisão, usa-se variáveis do tipo double, cujo formatador é %lf.

O URI Online Judge

- O URI Online Judge é uma plataforma de submissão de soluções a aproximadamente 1600 problemas de programação, de todos os níveis de dificuldade.
- É mantido pela Universidade Regional Integrada (URI), Campus Erechim.
  - Entre no site do URI Online Judge e cadastre um usuário.
  - Resolva os problemas 1001 a 1010 (primeiros 10 sequenciais) do URI Online Judge (menu "Problemas" - "buscar").
- O main deve ser declarado como int (int main()) e ao final do main deve ser executado um "return 0;" conforme modelo criado.
- É interessante testar os programas em um outro compilador (por ex. Dev C++) e só enviar ao sistema de submissão quando ele estiver aparentemente correto.

Modificador unsigned

- O modificador "unsigned..." aplicado a uma declaração de variável inteira faz com que ela trabalhe somente com valores positivos.

Comando If ou Comando Condicional

- Até agora, nossos programas tem sempre o mesmo comportamento, ou seja, executam a mesma sequência de comandos.
- Normalmente o que ocorre é que, dependendo dos dados recebidos, o programa deve executar comandos diferentes, resultando em "comportamentos" diferentes.
- Por exemplo: No exercício de cálculo das raízes de uma equação de segundo grau, se os valores de a,b e c são tais que a equação não tem raízes reais, $b^2 - 4ac$ resultará em um valor negativo, e a extração de sua raiz quadrada ocasionará um erro de execução.
- Um programa mais robusto deveria testar o sinal de $b^2 - 4ac$ e calcular as raízes somente se fosse maior ou igual a zero.

- Para que um programa possa executar comandos diferentes em diferentes situações usa-se o comando **if**. Seu formato é:

```
if (condição)
    {
    comandos;
    }
```

- que pode ser lido como "**Se** a condição é **verdadeira**, **então** execute os comandos"
- **condição** é uma expressão de comparação que resulta verdadeiro ou falso e utiliza, para comparação, os seguintes operadores (chamados operadores **relacionais**, porque identificam a **relação** entre dois valores):
  - Igual : representado por $==$
  - diferente : representado por $!=$
  - maior, menor, maior ou igual, menor ou igual: $>$, $<$, $>=$, $<=$

- O objetivo das chaves na estrutura

```
if (condição)
    {
    comandos;
    }
```

- é agrupar os comandos dentro das chaves em um único bloco, de modo que todos estejam "dentro" do comando **if**.
- Se houver apenas um comando a ser executado, as chaves podem ser dispensadas usando o formato a seguir:

```
if (condição)
    comando;
```

- Exemplos:

```
if (a >= 0)
    printf("Positivo\n");
if (a > b)
    printf("O maior é %d\n",a);
if (a > b)
    {
    printf("O maior é %d\n",a);
    }
```

- Os dois últimos são equivalentes, para mostrar que as chaves são opcionais quando há apenas um comando.

- ERRO!!! Se houver mais de um comando "dentro" do if e não houver chaves, apenas o comando imediatamente após o if estará dentro do if. Todos os comandos a partir do segundo comando estarão fora do if e serão executados sempre, independentemente da condição ser verdadeira ou falsa.
- Exemplo:

```
if (a >= 0)
    printf("Positivo\n");
    printf("Esse printf está fora do if");
```

- ERRO!!! Um outro erro muito comum é colocar um ponto-e-vírgula após a condição como no exemplo abaixo:

```
if (a >= 0);
    printf("Positivo\n");
```

- Nesse exemplo, o ponto-e-vírgula termina o comando if, de modo que o printf está fora do if e será executado sempre, independentemente da condição ser verdadeira ou falsa.

- No caso em que um conjunto de comandos deve ser executado se a condição for verdadeira e um conjunto diferente deve ser executado se a condição for falsa, usa-se a forma abaixo:

```
if (condição)
        {
        lista 1 de comandos;
        }
else
        {
        lista 2 de comandos;
        }
```

- que pode ser lido como "**Se** a condição é **verdadeira**, **então** execute a lista 1 de comandos, **senão** (se a condição é **falsa**) execute a lista 2 de comandos"

- Exemplo:

```
if (a >= 0)
        printf("Positivo\n");
else
        printf("Negativo");
if (a > b)
        printf("O maior é %d\n",a);
else
        printf("O maior é %d\n",b);
```

- Se a lista de comandos do if ou do else tiver mais de um comando, eles devem ser agrupados utilizando chaves;
- Se a lista do if ou do else tiver apenas um comando, as chaves são desnecessárias (mas podem ser utilizadas mesmo com um único comando):

```
if (a > 0)
{
        printf("Sou positivo");
        b=0;
}
```

Troca de valores entre variáveis

- Considere o problema de ler dois números, armazenando-os nas variáveis **a** e **b** e escrevê-los em ordem crescente.
- Uma solução possível é identificar o maior e escrevê-los na ordem identificada.

```
if (a<b)
        printf("%d%d",a,b);
else
        printf("%d%d",b,a);
```

- Outra solução é ordenar os dois números para uma ordem conhecida (por exemplo, garantir que o menor valor esteja na variável **a**) e escrever a de menor valor em primeiro lugar.

- Primeiro garante-se que a variável **a** tem o menor dos dois valores. Isso pode ser feito comparando **a** com **b**, e se **a** é maior que **b**, troca-se os valores entre as duas variáveis. Após escreve-se as duas variáveis.

  if (a>b)

  ”troca-se os valores entre **a** e **b**”

  printf(”%d %d”,a,b);

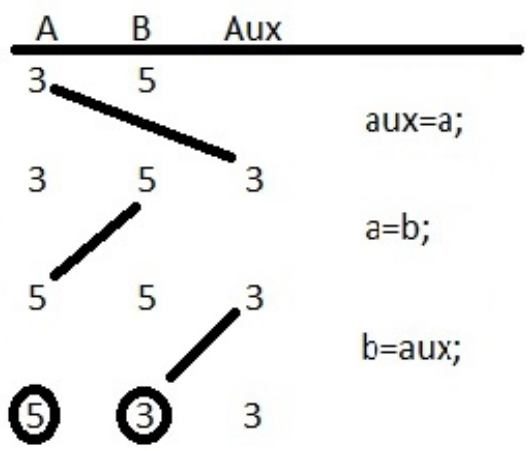
- Como se pode trocar os valores entre **a** e **b**?

  a=b;

  b=a;

- funciona?

- Para trocar os valores entre duas variáveis é necessário primeiro fazer uma cópia de uma das variáveis para uma outra variável. Normalmente chama-se essa variável de **aux**, por ter como função auxiliar na troca.
- A figura abaixo ilustra essa troca, iniciando por salvar o valor de a em aux:



- E a solução do problema anterior seria:

```
scanf("%d%d",&a,&b);
if (a>b)
      {
      aux=a;
      a=b;
      b=aux;
      }
printf("%d %d",a,b);
```

- O comando condicional deve ser utilizado quando há diversas situações que devem ser tratadas de formas diferentes pelo programa. Os passos para utilizá-lo são:
  1. Identificar com clareza quais as diferentes situações que devem ser tratadas pelo algoritmo.
  2. Definir qual o comportamento (comandos que devem ser executados) que o programa deve ter em cada uma das situações.
  3. Encontrar expressões que permitam ao programa identificar cada uma das situações.

Operadores Lógicos

- Há situações em que uma única comparação não é suficiente para o algoritmo identificar como tratar a situação.
- Uma situação simples seria a de encontrar, através de comparações, o maior valor entre 3 valores **a**, **b** e **c**.
- Para saber se o valor na variável **a** é o maior de todos, é necessário compará-lo com os valores em **b** e **c**.
  - Se **a** é maior que **b** **E** **a** é maior que **c**..."
- Ou pode-se querer saber, se entre 3 valores, algum é igual a zero:
  - Se **a**==0 **OU** **b**==0 **OU** **c**==0..."

- Nessas situações, para poder colocar mais de uma comparação, utiliza-se os chamados operadores lógicos, que combinam duas ou mais comparações para gerar um único resultado verdadeiro ou falso.
- Os operadores lógicos em C são 3:
  - O operador E (representado por &&). Quando aplicado a duas comparações, resulta verdadeiro se e somente se ambas as comparações são verdadeiras.
  - O operador OU (representado por ||). Quando aplicado a duas comparações, resulta verdadeiro se pelo menos uma delas é verdadeira (possivelmente as duas).
  - O operador de negação (representado pelo símbolo !) inverte o valor-verdade da expressão.

- uma condição pode conter diversas expressões de comparação combinadas com o uso desses 3 operadores.
- Ex: if (a>0 && b>0)... if (a>0 || b>0)...
- O operador **&&** tem prioridade maior que o operador ||.
- Se for necessário comparar um valor com outros dois (ex: a> *b* >c) deve-se efetuar as duas comparações separadamente e combiná-las com && (a>b && b>c).

Voltar para o índice

Exercícios de seleção simples do URI Online Judge

- 1035 - Teste de Seleção 1
- 1036 - Fórmula de Bhaskara
- 1037 - Intervalo
- 1038 - Lanche
- 1041 - Coordenadas de um Ponto

Caso mais simples: Há somente 2 situações

- Nesse caso um único "if" é suficiente para diferenciar as duas situações

```
if (a>0)
    printf("Positivo");
else
    printf("Negativo");
```

- Pode ocorrer também que no caso de a condição falhar, nada deve ser feito.
  - Nesse caso o "else" pode ser omitido

```
if (a>b)
    printf("%d é maior",A);
```

- Resolva os exercícios 20, 40, 60, 80, 100, 200, 1200, 300, 350 e 360 da lista de condicionais.

Caso menos simples: Há mais de 2 situações

- Nesse caso um único "if" não é suficiente para diferenciar as diferentes situações
- Pode-se utilizar um "if" para identificar cada situação
  - Nesse caso, a expressão deve ser suficiente para identificar cada caso dentre todos os casos

- C00000020 - Faça um algoritmo que leia um valor e escreva:
  - 0, se o valor é zero;
  - 1, se o valor é maior que zero;
  - -1 - se o valor é negativo.

Voltar para o índice

- Nesse exercício podem ocorrer 3 situações distintas, que exigem tratamentos distintos:
  - O valor lido é maior que 0
  - O valor lido é igual a 0
  - O valor lido é menor que 0
- E pode ser resolvido com 3 comandos if, um para cada situação:

```
printf("Digite um valor:");
scanf("%d",&a);
if (a>0)
    printf("1");
if (a==0)
    printf("0");
if (a<0)
    printf("-1");
```

Voltar para o índice

Alternativa B - Comandos if aninhados

- Uma alternativa interessante mais interessante é com o uso da cláusula **else**.
- Após identificado e tratado um dos casos, havendo mais de um caso restante, deve-se identificar qual o próximo caso a ser tratado.

```
if (a>0)
    printf("1");
else
    "Aqui devem ser tratados os casos restantes"
```

- O formato do comando if é:

  if (condição)

  lista de comandos 1

  else

  lista de comandos 2

- As listas de comandos (do entao e do senao) podem conter qualquer comando dentre os comandos já vistos, **scanf**, **printf**, atribuição... inclusive o comando **if**.
- Para utilizar um comando **if** dentro de outro (no então ou no senão), ele deve estar completo (**if** (condição) { lista de comandos }
- Um exemplo de uso é:

Voltar para o índice

```
if (a==0)
    printf("zero");
else
    if (a>0)
        printf("positivo");
    else printf("negativo");
```

Voltar para o índice

O que NÃO fazer em hipótese nenhuma

- Combinar as duas soluções:

```
if (a==0)
        printf("0");
if (a>0)
        printf("positivo");
        else printf("negativo")
```

- Qual é o problema com essa abordagem?

Voltar para o índice

- C00000100 - Faça um algoritmo que leia 3 valores v_1, v_2 e v_3, e escreva-os em ordem crescente.
- Há 6 ordens possíveis (sem contar a possibilidade de ocorrerem números repetidos)

  $v1 < v2 < v3$
  $v1 < v3 < v2$
  $v2 < v1 < v3$
  $v2 < v3 < v1$
  $v3 < v1 < v2$
  $v3 < v2 < v1$

Voltar para o índice

- Uma alternativa é utilizar um "if" para cada situação:

```
if (v1<v2 && v2<v3)
    printf ("%d %d %d",v1,v2,v3);
if (v1<v3 && v3<v2)
    printf ("%d %d %d",v1,v3,v2);
if (v2<v1 && v1<v3)
    printf ("%d %d %d",v2,v1,v3);
if (v2<v3 && v3<v1)
    printf ("%d %d %d",v2,v3,v1);
if (v3<v1 && v1<v2)
    printf ("%d %d %d",v3,v1,v2);
if (v3<v2 && v2<v1)
    printf ("%d %d %d",v3,v2,v1);
```

- O que acontecerá se houverem valores repetidos?
- Como se pode corrigir?

Voltar para o índice

- Alternativa B - Cada teste só é feito se o teste anterior falhou:

```
if (v1<v2 && v2<v3)
    printf("%d␣%d␣%d",v1,v2,v3);
else if (v1<v3 && v3<v2)
        printf("%d␣%d␣%d",v1,v3,v2);
     else if (v2<v1 && v1<v3)
             printf("%d␣%d␣%d",v2,v1,v3);
          else if (v2<v3 && v3<v1)
                  printf("%d␣%d␣%d",v2,v3,v1);
               else if (v3<v1 && v1<v2)
                       printf("%d␣%d␣%d",v3,v1,v2);
                    else if (v3<v2 && v2<v1)
                            printf("%d␣%d␣%d",v3,v2,v
```

- O que acontecerá se houverem valores repetidos? Como se pode corrigir?
- O último teste é necessário?

Voltar para o índice

- C00000200 - Faça um algoritmo que leia 3 valores v1, v2 e v3 e coloque-os em ordem crescente, de forma que v1 contenha o menor, v2 contenha o elemento do meio (nem o maior, nem o menor), e v3 contenha o maior. Escreva os valores ordenados.
  - A solução anterior não pode ser utilizada nesse exercício porque o enunciado pede explicitamente que os valores sejam trocados e v1 fique com o menor valor.
  - Para se conseguir isso, compara-se as variáveis entre si, trocando-as entre si quando necessário:

Voltar para o índice

```
scanf("%d%d%d",&a,&b,&c);
if (a>b){
      aux=a;
      a=b;
      b=aux;
      }
if (a>c){
      aux=a;
      a=c;
      c=aux;
      }
if (b>c){
      aux=b;
      b=c;
      c=aux;
      }
printf(" %d %d %d",a,b,c);
```

Voltar para o índice

- C00000250 - Escreva um algoritmo que leia os valores das quatro provas de um aluno e escreva a média aritmética considerando apenas as três melhores notas. Por exemplo, se o valores lidos foram 9, 9.5, 7, e 8, a média será $(9 + 9.5 + 8)/3$ (a prova de nota 7 é descartada). Dica:Não esqueça de considerar a possibilidade de ocorrerem notas iguais.
- Sugestão: Pode-se ordenar parcialmente os valores, de modo que o menor dos 4 valores esteja na primeira variável e calcular a média somente das outras 3 notas.

Voltar para o índice

C00000300) Faça um programa que leia 3 valores a, b e c, coeficientes de uma equação de segundo grau, e verifique se a equação tem raízes reais.

Se a equação tiver raízes reais, calcule e escreva as raizes da equação em ordem crescente.
Se não tiver, escreva "A equação não possui raízes reais".
Dica:As raízes de uma equação podem ser calculadas pela fórmula de Baskhara.

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

$$x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

- Uma equação não possui raízes se reais se $b^2 - 4ac < 0$

- C00000350 - Faça um algoritmo que leia 3 valores a, b e c, lados de um triângulo, e verifique o tipo de triângulo formado escrevendo:
  - 0 - se o triângulo é equilátero (os três lados são iguais);
  - 1 - se o triângulo é isósceles (dois lados iguais e um diferente);
  - 2 - escaleno (3 lados diferentes).

Voltar para o índice

Alternativa 1

```
scanf("%f%f%f",&a,&b,&c);
if (a==b && a==c)
        printf("Equilatero %d",0);
if (a!=b && a!=c && b!=c)
        printf("Escaleno %d",2);
if (a==b && a!=c || a==c && a!=b || b==c && a!=b)
        printf("Isósceles %d",1);
```

Voltar para o índice

Alternativa 2

```
scanf("%f%f%f",&a,&b,&c);
if (a==b && a==c)
        printf("Equilatero %d",0);
        else if (a!=b && a!=c && b!=c)
                printf("Escaleno %d",2);
                printf("Isósceles %d", 1);
```

Voltar para o índice

Alternativa 3

```
scanf("%f%f%f",&a,&b,&c);
if (a==b && a==c)
        printf("Equilatero %d",0);
        else if (a==b || a==c || b==c)
                printf("Isósceles %d", 1);
                else printf("Escaleno %d",2);
```

Voltar para o índice

- Qual das três alternativas é mais eficiente?
- Por quê é mais eficiente?
- Por que a expressão do Isósceles na alternativa 1 é diferente da alternativa 3?
- Em que situações se pode colocar um **se** dentro do outro (**Se**'s aninhados)
- Pode-se colocar sempre?
- Pode-se usar **Se**'s aninhados no código seguinte?

Voltar para o índice

C00000360) Faça um programa que leia 3 valores a, b e c, lados de um triangulo, e verifique o tipo de triangulo formado escrevendo:

- 0 - se o triângulo é retângulo ($A^2 = B^2 + C^2$);
- 1 - se o triângulo é acutângulo ($A^2 > B^2 + C^2$) ;
- 2 - obtusângulo ($A^2 < B^2 + C^2$).

Considere que, para aplicar as relações mostradas, o programa deve garantir que o maior dos 3 lados estará em A. Se o lado maior não está em A, o programa deve trocar os valores entre as variáveis para que o maior valor fique em A.

- C00000400 - Faça um algoritmo que leia 3 valores **a**,**b** e **c** e verifique se formam um triângulo. Se formarem, calcule e escreva a área do triângulo formado (veja exercício S00000900). Se não formarem, escreva -1.

  Obs: A área do triângulo de lados **a**, **b** e **c** pode ser calculada pela fórmula de Hierão, dada abaixo:

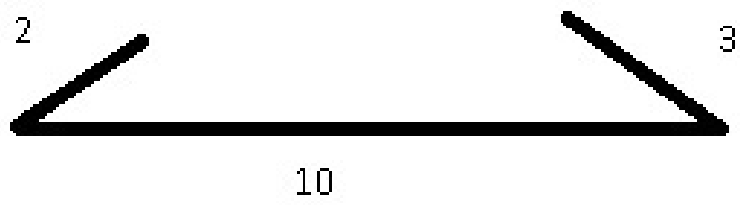
$$Area = \sqrt{S(S-a)(S-b)(S-c)}$$

onde

$$S = \frac{a+b+c}{2}$$

Voltar para o índice

Obs: Uma triângulo pode ser definido pelo valor de seus três lados. Entretanto, nem todo conjunto de 3 valores formam um triângulo. Assim, 3,4,5 formam um triângulo. 10,3,2 não formam. 10,5,5 formam? Qual a condição para que 3 valores formem um triângulo?



Voltar para o índice

Condição para 3 valores formarem um triângulo

- "Cada um dos três lados deve ser menor que a soma dos outros dois"
  - ou seja, "A deve ser menor que B+C, B deve ser menor que A+C e C deve ser menor que A+B"
  - ou, como uma condição: **A<B+C e B<A+C e C<A+B**

Voltar para o índice

Condição para 3 valores **NÃO** formarem um triângulo

- "Um dos lados deve ser maior ou igual à soma dos outros dois"
  - ou seja, "A deve ser maior ou igual a B+C, B deve ser maior ou igual a A+C **OU** C deve ser maior ou igual a A+B"
  - ou, como uma condição: **A>=B+C ou B>=A+C ou C>=A+B**

Voltar para o índice

- C00000410 - Faça um algoritmo que leia 3 valores **a**, **b** e **c** e verifique se formam um triângulo e, se formarem, o tipo de triângulo formado, escreva:
  - 0 - se não formarem triângulo;
  - 1 - se formarem um triângulo equilatero (os três lados são iguais);
  - 2 - se formarem um triângulo isósceles (dois lados iguais e um diferente);
  - 3 - se formarem um triângulo escaleno (3 lados diferentes)

Voltar para o índice

Tabela de precedência de operadores em C

Prec.	Símbolo	Função	Associatividade
1	++ - -	Incremento e decremento pósfixo	esquerda para direita
1	()	Parênteses (chamada de função)	
1	[ ]	Elemento de array	
1	.	Seleção de campo de structure	
1	->	Seleção de campo de structure a partir de ponteiro	
2	++ - -	Incremento e decremento prefixo	direita para a esquerda
2	+ -	Adição e subtração unária	
2	! ~	Não lógico e complemento	
2	(tipo)	Conversão de tipo de dado	
2	*	Desreferência	
2	&	Referência (endereço de elemento)	
3	* / %	Multiplicação, divisão, e módulo (resto)	esquerda para a direita
4	+ -	Adição e subtração	
5	<< >>	Deslocamento de bits à esquerda e à direita	
6	< <=	"menor que" e "menor ou igual que"	
6	> >=	"maior que" e "maior ou igual que"	
7	= = !=	"Igual a" e "diferente de"	
8	&	E bit a bit	
9	^	Ou exclusivo para bits	
10	|	Ou para bits	
11	&&	E lógico	
12	||	Ou lógico	
13	c ? t : f	Condição ternária	direita para esquerda
14	=	Atribuição	
14	+= -=	Atribuição por adição ou subtração	
14	*= /= %=	Atribuição por multiplicação, divisão ou módulo (resto)	
14	<<= >>=	Atribuição por deslocamento de bits	
14	&= ^= |=	Atribuição por operações lógicas	
15	,	vírgula	esquerda para direita

C00000410) Faça um programa que leia 3 valores l_1,l_2 e l_3 e verifique se formam um triângulo e, se formarem, o tipo de triângulo formado, escrevendo:

- 0 - se não formarem triângulo;
- 1 - se formarem um triângulo equilátero (os três lados são iguais);
- 2 - se formarem um triângulo isósceles (dois lados iguais e um diferente);
- 3 - se formarem um triângulo escaleno (3 lados diferentes)

- C00000500) Faça um programa que leia 3 notas de um aluno e escreva sua média harmônica.
- A média harmônica entre três valores N_1, N_2 e N_3 é calculada pela expressão

$$\frac{3}{\frac{1}{N_1} + \frac{1}{N_2} + \frac{1}{N_3}}$$

.
- O que acontece se alguma das notas for igual a 0? Que resultado o programa deve emitir?
- Se uma das notas for igual a 0, a média é igual a 0, mas se o programa tentar avaliar a expressão acima com uma nota igual a 0 ocorrerá uma divisão por 0, que é uma operação inválida em programação.

C00000600) Faça um programa que leia 3 notas de um aluno e escreva sua média harmônica. Se o aluno obteve média abaixo de 6.0, E SOMENTE NESSE CASO, leia uma quarta nota (da prova de recuperação) e substitua a menor das três notas pela nota da recuperação e recalcule a média harmônica. Escreva a média harmônica final e o conceito obtido:

- 0, se média harmônica (MH) $<$ 6.0;
- 1, se 6.0 $<=$ MH $<$7.0;
- 2, se 7.0 $<=$ MH $<$ 8.0;
- 3, se 8.0 $<=$ MH $<$ 9.0;
- 4, se MH$>=$9.0.

A média harmônica entre três valores N1, N2 e N3 é calculada pela expressão

$$\frac{3}{\frac{1}{N_1} + \frac{1}{N_2} + \frac{1}{N_3}}$$

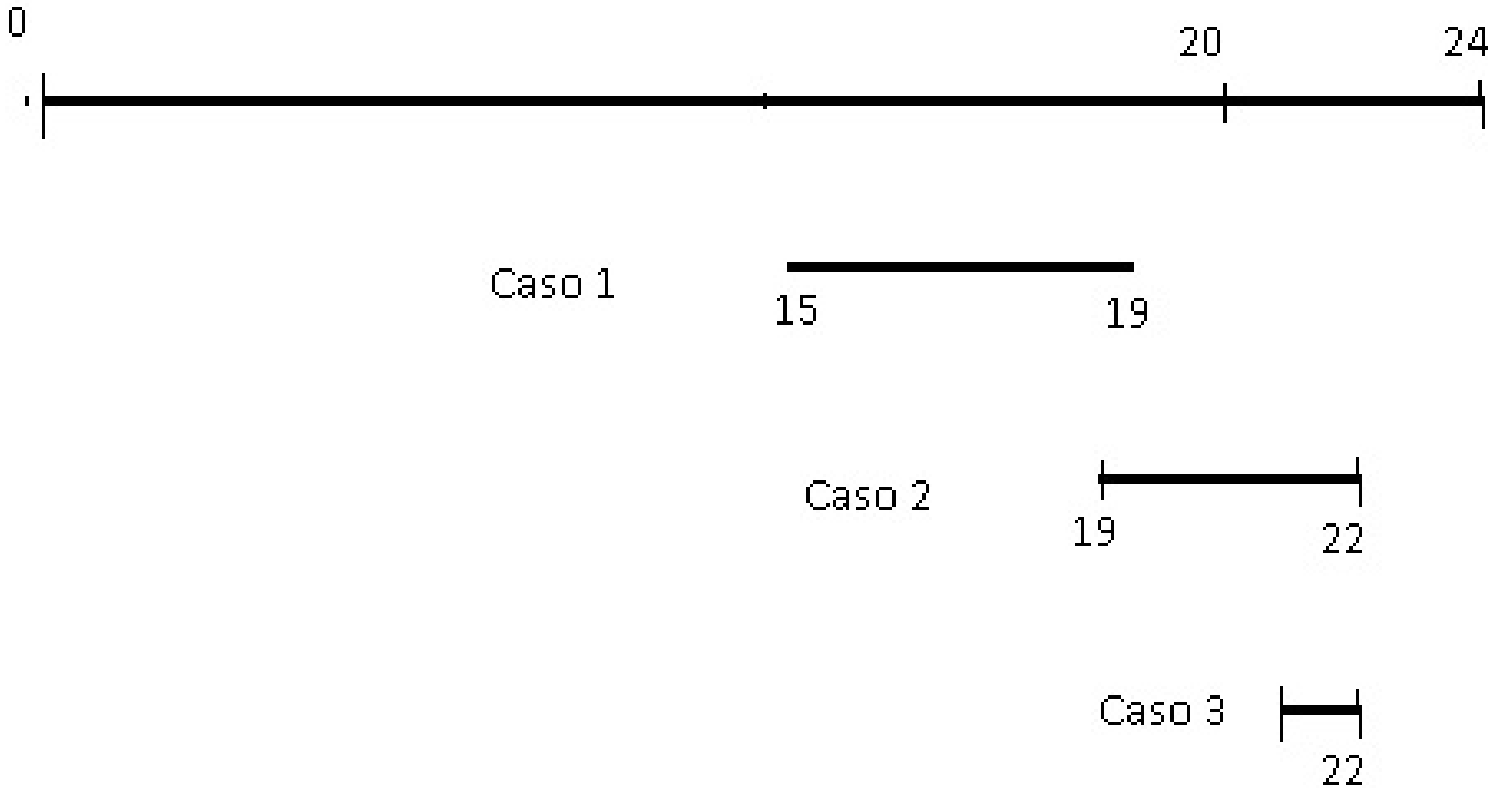
.

- C00000800) Faça um programa que leia 4(quatro) valores e escreva os 3 (tres) maiores em ordem decrescente. Considere que podem ocorrer valores iguais.

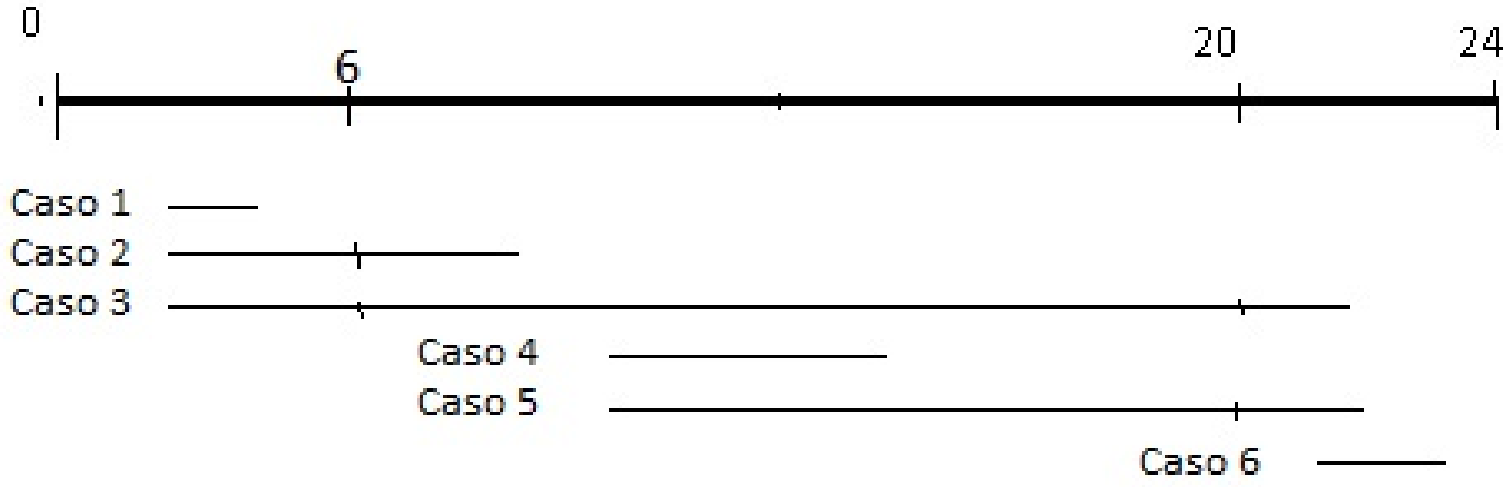
C00000660) Faça um programa que leia 4 valores, Hi, Mi, Hf, Mf, representando respectivamente a hora e minuto inicial e final de um evento, e calcule a duração do mesmo em horas e minutos. Considere que o evento pode iniciar em um dia e terminar no dia seguinte. Algumas (mas não todas) situações que podem ocorrer são:

- Entrada: 10:30 Saída: 10:45
- Entrada: 10:30 Saída: 11:15
- Entrada: 22:15 Saída: 1:45
- Entrada: 22:15 Saída: 10:45

- C00001000) Faça um programa que leia para um trabalhador o valor que ganha por hora, a hora de entrada e a hora de saída (valores inteiros, sem minutos) e calcule quanto ele ganhou pelo turno. Considere que ele entra e sai no mesmo dia, e que as horas a partir das 20:00 valem 20% a mais (adicional noturno).
  - Quantos casos diferentes existem?
  - Que cálculo deve ser feito em cada caso?
  - Como o algoritmo pode identificar cada caso?


0
20
24
Caso 1
15
19
Caso 2
19
22
Caso 3
22

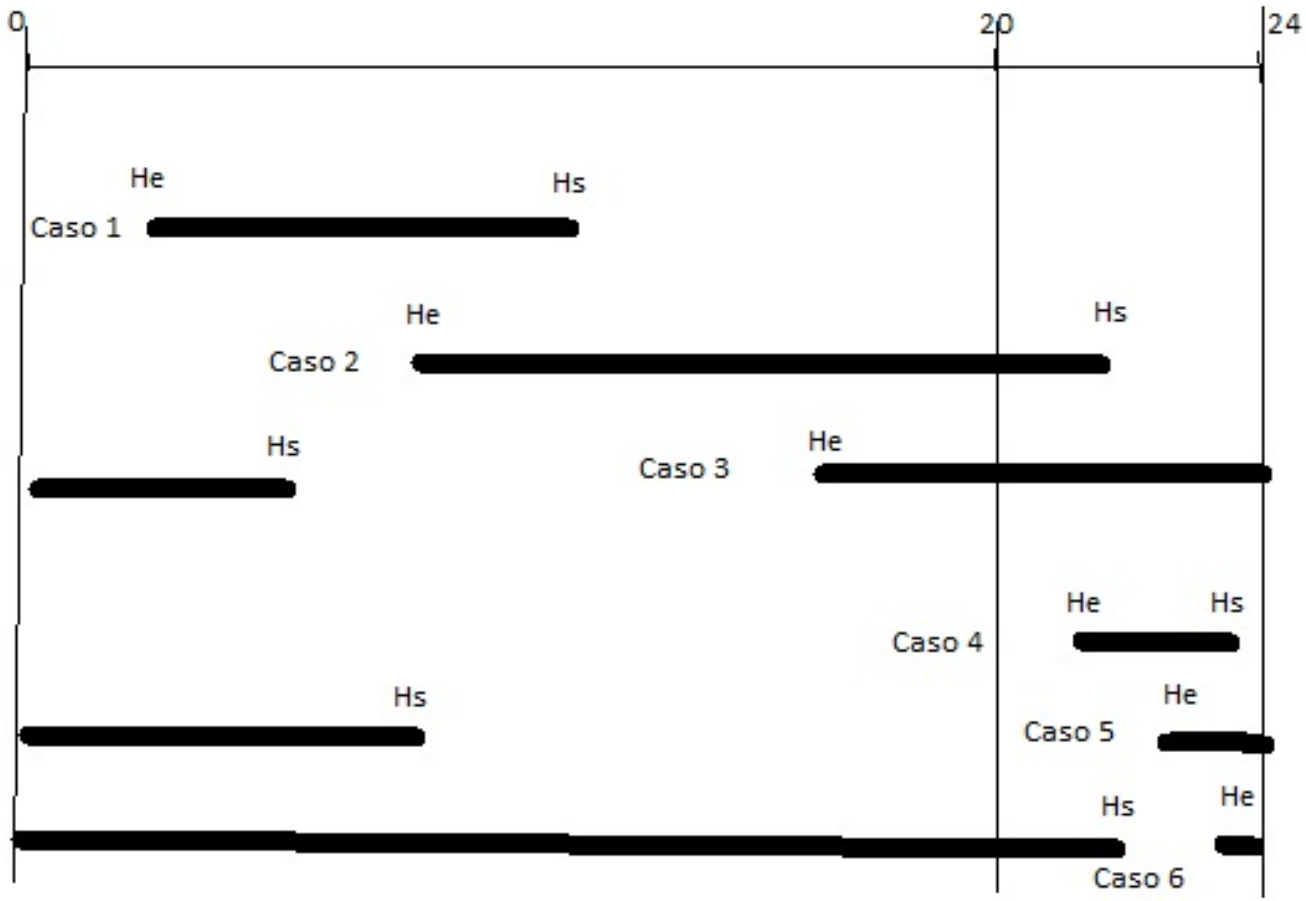
- C00001100) Faça um programa que leia para um trabalhador o valor que ganha por hora, a hora de entrada e a hora de saída (valores inteiros, sem minutos) e calcule quanto ele ganhou pelo turno. Considere que ele entra e sai no mesmo dia, e que as horas antes das 6:00 da manhã e a partir das 20:00 valem 20% a mais (adicional noturno).
  - Quantos casos diferentes existem? (figura adiante)
  - Que cálculo deve ser feito em cada caso?
  - Como o algoritmo pode identificar cada caso?


0
6
20
24
Caso 1
Caso 2
Caso 3
Caso 4
Caso 5
Caso 6

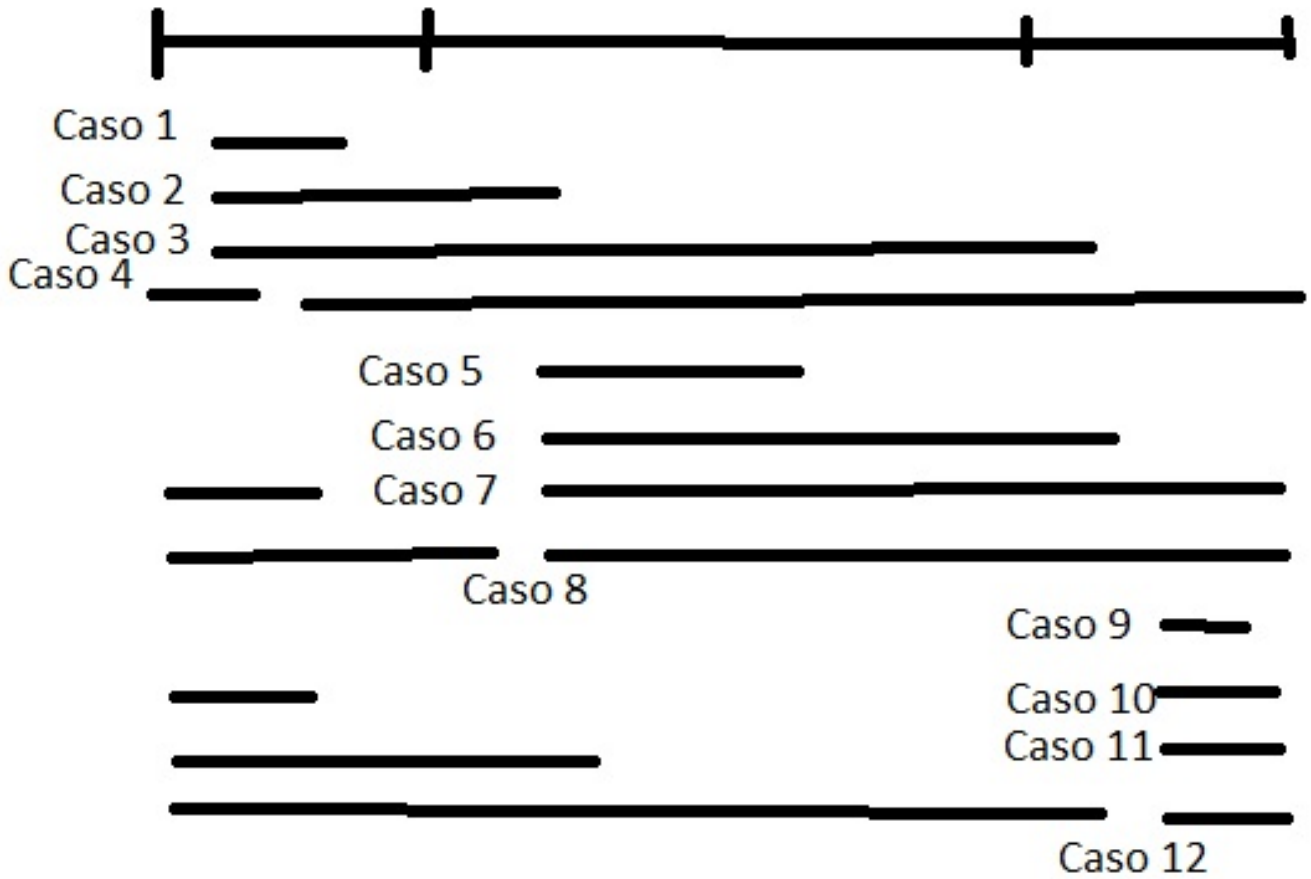
- C00001200) Faça um programa que leia para um trabalhador o valor que ganha por hora, a hora de entrada e a hora de saída (valores inteiros, sem minutos) e calcule quanto ele ganhou pelo turno. Considere que ele pode entrar em um dia e sair no outro, mas que o total de horas trabalhadas não excede 23 horas.
  - Há duas situações possíveis, que exigem cálculos diferentes.
    - a) O trabalhador entra e sai no mesmo dia. P.ex.: entrou às 9h e saiu às 15h.
    - b) O trabalhador entra em um dia e sai no dia seguinte. P.ex.: entrou às 20h e saiu às 2h.
  - Como o algoritmo pode identificar qual dos dois casos ocorreu?
  - Como pode ser calculado o número de horas em cada um dos casos?

- C00001250) Faça um programa que leia para um trabalhador o valor que ganha por hora, a hora de entrada e a hora de saída (valores inteiros, sem minutos) e calcule quanto ele ganhou pelo turno. Considere que ele pode entrar em um dia e sair no dia seguinte, e que se ele permanecer mais do que 8 horas, as duas horas a partir da nona hora valem 20% a mais, e as horas a partir da décima primeira hora valem 50% a mais (horas extras).
  - Obs: Nesse exercício o valor pago depende somente da quantidade de horas, não importando se as horas foram no mesmo dia ou entre dois dias. Nesse caso, a solução fica mais simples se, em um primeiro momento o algoritmo calcular o número de horas e, após, aplicar a tabela de preço.
  - E as únicas horas que custam 20% a mais são a nona e a décima. A décima-primeira hora já custa 50% a mais.

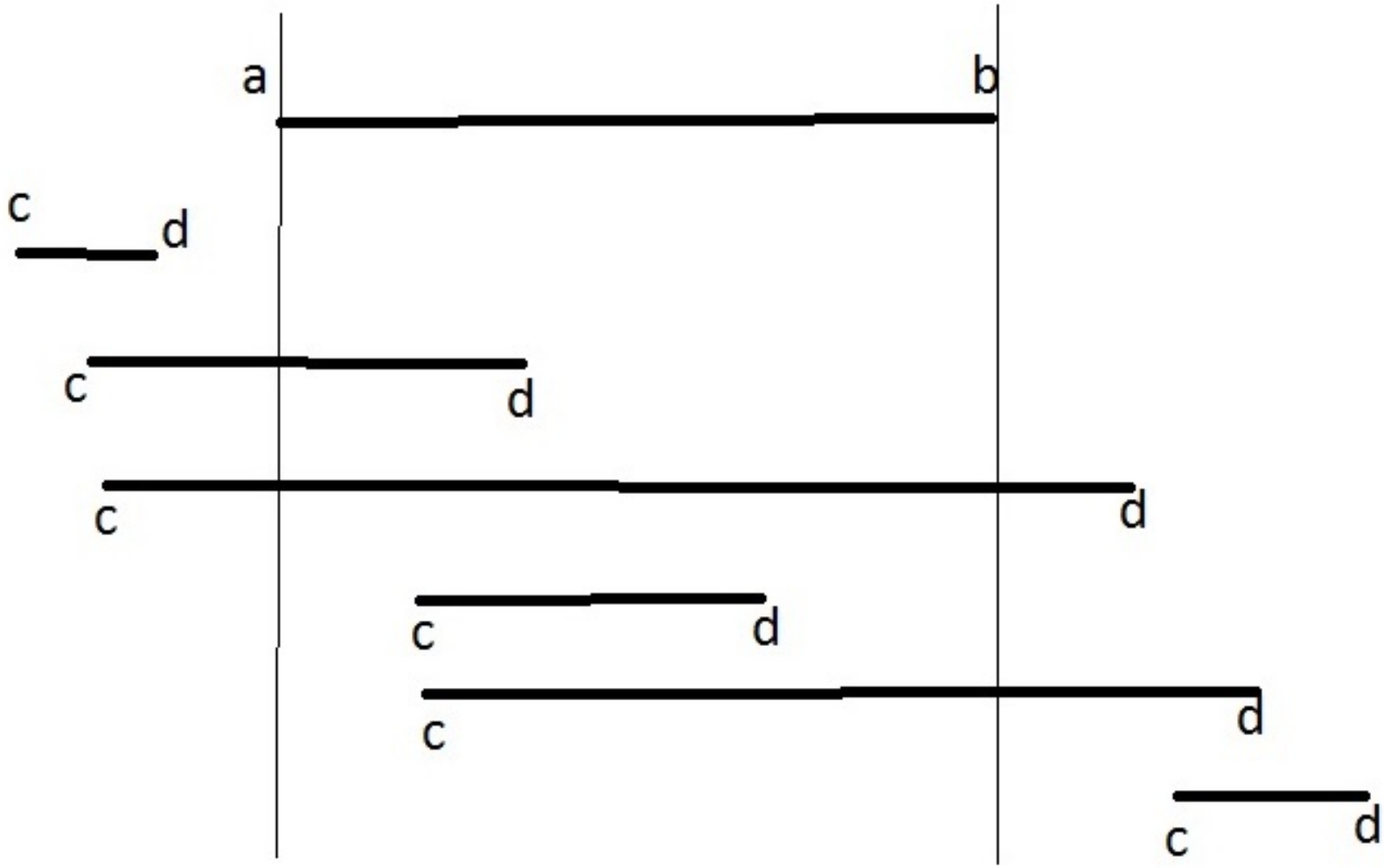
- C00001300) Faça um algoritmo que leia para um trabalhador o valor que ganha por hora, a hora de entrada e a hora de saída (valores inteiros, sem minutos) e calcule quanto ele ganhou pelo turno. Considere que ele pode entrar em um dia e sair no outro, mas que o total de horas trabalhadas não excede 23 horas. Considere que as horas das 20:00 às 24:00 valem 20% a mais (adicional noturno).


0
20
24
He
Hs
Caso 1
He
Hs
Caso 2
Hs
He
Caso 3
He
Hs
Caso 4
Hs
He
Caso 5
Hs
He
Caso 6

- C00001400) Faça um algoritmo que leia para um trabalhador o valor que ganha por hora, a hora de entrada e a hora de saída (valores inteiros, sem minutos) e calcule quanto ele ganhou pelo turno. Considere que ele pode entrar em um dia e sair no outro, mas que o total de horas trabalhadas não excede 23 horas. Considere também que as horas antes das 6:00 da manhã e a partir das 20:00 valem 20% a mais (adicional noturno).


Caso 1
Caso 2
Caso 3
Caso 4
Caso 5
Caso 6
Caso 7
Caso 8
Caso 9
Caso 10
Caso 11
Caso 12

- C00001450) Faça um algoritmo que leia 4 valores a,b,c,d representando os limites inferior e superior de dois intervalos fechados [a,b] e [c,d] e escreva os limites inferior e superior do intervalo resultante da intersecção dos dois intervalos. Ex: A intersecção entre os intervalos [1,5] e [2,4] é o intervalo [2,4]. Se os intervalos forem disjuntos, isso é, não tiverem intersecção, escreva "Intervalos são disjuntos".


a
b
c
d
c
d
c
d
c
d
c
d
c
d

- C00002100) A distância entre dois pontos definidos pelas coordenadas (X_1, Y_1) e (X_2, Y_2) é dada por

$$\sqrt{(X_1 - X_2)^2 + (Y_1 - Y_2)^2}$$

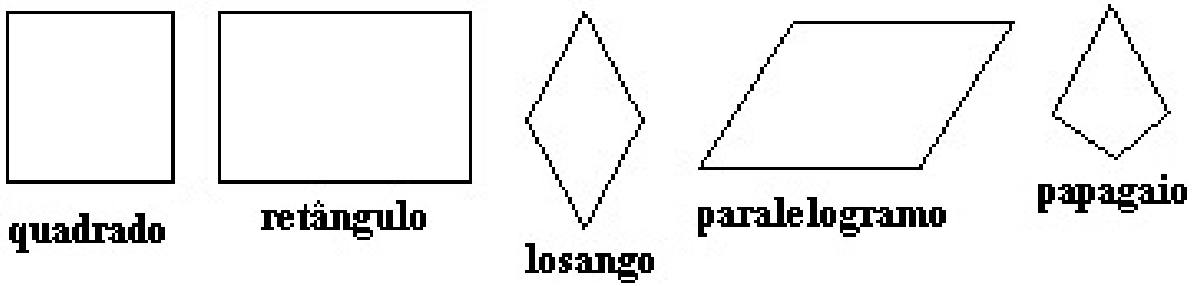
.

- Faça um programa que leia 8 valores representando as coordenadas X e Y de 4 pontos,vértices de um polígono, e verifique se os pontos formam um quadrado, escrevendo:

  1 - se formam um quadrado;
  0 - se não formam.

- Considere que os pontos são lidos no sentido horário, seguindo o perímetro do quadrado.

- C00002150) Faça um programa que leia oito valores correspondentes às coordenadas dos quatro vértices de um quadrilátero convexo no espaço cartesiano (X_0, Y_0, X_1, Y_1, X_2, Y_2, X_3, Y_3). O algoritmo deve identificar o tipo de polígono formado escrevendo:
  1. se os 4 pontos formam um quadrado;
  2. se formam um retângulo;
  3. se formam um losango;
  4. se formam um paralelograma;
  5. se formam um papagaio (2 pares de lados adjacentes iguais. Ex: lados de tamanhos 3,3,4 e 4);
  6. se não formam nenhum dos anteriores.

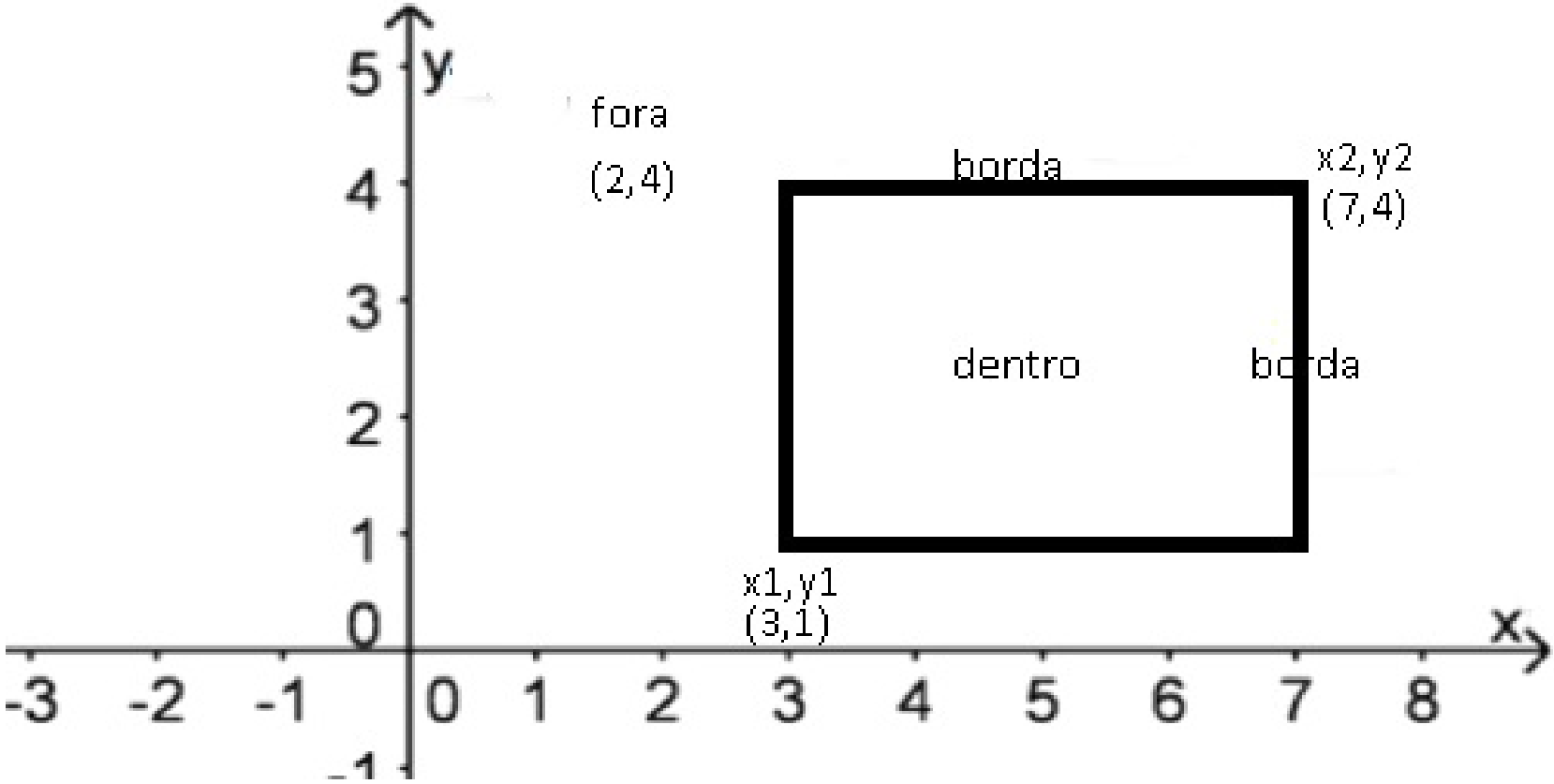

quadrado
retângulo
losango
paralelogramo
papagaio

- A distância (tamanho dos lados) entre dois pontos quaisquer (X_i, Y_i) e (X_j, Y_j) pode ser obtida através da fórmula distXiYiXjYj $= \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2)}$.
- Por exemplo, se os pontos lidos foram (3,2), (0,5), (3,8) e (6,5), a figura formada é um quadrado e o algoritmo escreve 1.
- Para que a figura seja um quadrado, os comprimentos das duas diagonais devem ser iguais, bem como os 4 lados.
- Se os pontos lidos foram (4,2), (1,4), (4,6) e (7,4), a figura formada é um losango.
- Se os pontos lidos foram (2,3), (0,5), (3,8) e (5,6), a figura formada é um retângulo.
- Se os pontos lidos foram (7,3), (0,3), (2,5) e (5,5), a figura formada não é nenhuma das anteriores e o programa escreve 6
- Os pontos são fornecidos em sentido horário ao longo do perímetro do quadrilátero.

- 26) Faça um programa que leia as dimensões (altura, largura e profundidade) de duas caixas e verifique se a primeira caixa pode ser colocada dentro da segunda. Considere que as caixas podem ser rotacionadas em qualquer direção. Se a primeira caixa couber dentro da segunda escreva 1, caso contrário escreva 0.

- C00003300) Faça um programa que leia 4 valores X1,Y1,X2,Y2, correspondendo às coordenadas do canto inferior esquerdo e canto superior direito de uma região retangular no plano. Leia a seguir dois valores X,Y correspondendo a um ponto no plano e escreva:

  0 - Se o ponto está fora da região retangular;
  1 - Se o ponto está dentro da região retangular;
  2 - Se o ponto está exatamente na borda da região retangular.


y
x
fora
(2,4)
borda
x2,y2
(7,4)
dentro
borda
x1,y1
(3,1)
5
4
3
2
1
0
-1
-3
-2
-1
0
1
2
3
4
5
6
7
8

- C00001750) Escreva um programa que leia duas datas, cada uma composta de Dia, Mês e Ano, e as escreva em ordem cronológica crescente.
- Ex:se as datas são 01/04/2000 e 17/05/1988, o algoritmo deve escrever 17/05/1988 01/04/2000.
- Sugestões:
  1. Converta cada data (dia,mês,ano) em um único valor, da forma como fizemos com hora-minuto-segundo, para poder efetuar uma única comparação; ou
  2. Efetue primeiro a comparação entre os anos, se esses forem iguais compare os meses e, se anos e meses forem iguais compare os dias.

- 29) Escreva um programa que leia três datas, cada uma composta de Dia, Mês e Ano, e as escreva em ordem cronológica crescente.
- Ex:se as datas são 01/04/2000, 17/05/1988 e 23/10/1969, o algoritmo deve escrever 23/10/1969 17/05/1988 01/04/2000.
- Sugestão: As mesmas do exercício anterior

- C00001790) Escreva um programa que leia uma data, composta por dia, mês e ano, e verifique se a data corresponde ao último dia do mês, escrevendo:
  - 1, se for o último dia do mês
  - 0 se não for o último dia do mês.
- Considere, para simplificar o problema, que ano bissexto é aquele divisível por 4, e que fevereiro tem 28 dias (29 em ano bissexto).
- Setembro, abril, junho e novembro têm 30 dias e todos os outros meses tem 31 dias.

- C00001800) Escreva um algoritmo que leia uma data, composta por dia, mês e ano, e escreva a data correspondente ao dia seguinte.
- Considere, para simplificar o problema, que ano bissexto é aquele divisível por 4, e que fevereiro tem 28 dias (29 em ano bissexto), setembro, abril, junho e novembro têm 30 dias e todos os outros meses tem 31 dias. Sugestão: lâmina seguinte

leia(dia,mes,ano)
Se {dia é o último dia do ano}
    estao escreva (1, 1, ano+1)
senao se {dia é o último dia do mes}
    estao escreva (1, mes+1, ano)
senao escreva(dia+1,mes,ano)

C00001850) Escreva um programa que leia uma data, composta por dia, mês e ano, e escreva quantos dias passaram-se desde o início do ano. Considere, para simplificar o problema, que ano bissexto é aquele divisível por 4, e que fevereiro tem 28 dias (29 em ano bissexto), setembro, abril, junho e novembro têm 30 dias e todos os outros meses tem 31 dias.

C00001900) Para enviar uma carta são necessários um selo e um envelope. O selo custa 12 centavos e o envelope custa 5 centavos. Faça um programa que leia uma quantia inicial de selos, envelopes e centavos, e escreva o número de cartas que podem ser enviadas com esses selos, envelopes e centavos (utilizando-os para comprar mais selos e envelopes). Considere que não é possível converter selos ou envelopes em dinheiro.

C00001925) Para enviar uma carta são necessários um selo e um envelope. O selo custa 12 centavos e o envelope custa 5 centavos. Faça um programa que leia uma quantia inicial de selos, envelopes e dinheiro (em reais), e escreva o número de cartas que podem ser enviadas com esses selos, envelopes e centavos (utilizando-os para comprar mais selos e envelopes). Escreva também, nessa ordem, a quantidade de selos, envelopes e dinheiro (em centavos), que restará após enviadas as cartas. Considere que não é possível converter selos ou envelopes em dinheiro.

C00001950) Uma fábrica produz um recipiente de plástico com sua tampa (também de plástico). Ambos os componentes utilizam o mesmo equipamento para fabricação (ou seja, não podem ser fabricados ao mesmo tempo). A fabricação do recipiente consome duas horas; a fabricação da tampa consome meia hora. Um cliente deseja o máximo de recipientes (com tampa) para 10 dias. A fábrica trabalha 24 horas/dia e já dispõe de uma quantidade r de recipientes e t de tampas em seu estoque (não necessariamente iguais). Faça um programa que leia os valores de r e t e informe o máximo de conjuntos recipiente-tampa que ela pode fornecer em 10 dias.

- C00003000) Escreva um programa que leia dois valores D e DS, correspondentes a um dia do mes, e ao dia da semana que corresponde a ele (1-domingo 2-segunda 3- terça 4-quarta 5-quinta 6-sexta 7-sábado). Calcule e escreva em que dia da semana caiu o dia primeiro do mês do dia digitado.
- Exemplo: dia 10 no mês é 3 (terça) na semana. Resposta 1 (domingo)
- Exemplo: dia 30 no mês é 4 (quarta) na semana. Resposta 3 (terça feira)

C00003100) Faça um programa que leia as dimensões (altura, largura e profundidade) de duas caixas e verifique se a primeira caixa pode ser colocada dentro da segunda. Considere que as caixas podem ser rotacionadas em qualquer direção. Se a primeira caixa couber dentro da segunda escreva 1, caso contrário escreva 0.

C00003200) Faça um programa que leia dois números de três dígitos cada um, e verifique se possuem os mesmos dígitos. Considere que pode haver dígitos repetidos em um número, e que a cada dígito de um número deve corresponder exatamente um dígito do outro número. Assim, os números 221 e 112 não tem os mesmos dígitos, porque apesar de ambos terem somente os dígitos 1 e 2, aos dois dígitos 2 do primeiro número corresponde o mesmo dígito no segundo número. O algoritmo deve escrever 1, caso os números tenham os mesmos dígitos, e 0 em caso contrário.

C00003250) Faça um programa que leia um número de três dígitos e escreva o maior número que possui os mesmos dígitos do número lido. Se um dígito aparecer repetido no número lido, deve ser repetido o mesmo número de vezes no número gerado.

Lista de problemas **sequenciais e condicionais** por categorias:

- Separação de dígitos
- Ordenar poucos valores
- Conversões hora-minuto-segundo
- Aplicação de tabelas

Voltar para o índice

Comandos de Repetição

- São utilizados para que um comando ou conjunto de comandos sejam executados mais de uma vez.
- A linguagem C possui 3 comandos de repetição:
  - O comando **for**
  - O comando **while**
  - O comando **do**...**while**
- O formato do comando **While** é:

  while (condição) comando;
  ou
  while (condição) { Lista de comandos }

  onde **condição** é uma expressão lógica, como as condições do comando **if**.

- Ao entrar no comando **while** a condição é avaliada.
- Se ela é verdadeira, a lista de comandos é executada.
- Isso (avalia condição + executa comandos) se repete até que a condição seja falsa.
- Por exemplo, o trecho ao lado escreve de quanto a quanto?

```
a=1;
while (a<4){
    printf("%d\n",a);
    a=a+1;
    }
```

Voltar para o índice

Teste de Mesa

- Uma coluna para cada variável e uma coluna para os comandos de escrita
- Executa-se os comandos sequencialmente, um a um
- A cada atribuição, a coluna da variável que recebeu o valor deve ser atualizada
- A cada comando de escrita, a coluna das escritas é atualizada
- Esse formalismo é usado há 60 anos e funciona muito bem. Não tente inventar algo diferente.

```
a=1;
while (a<4){
    printf("%d\n",a);
    a=a+1;
    }
```

a	esc
1	

Voltar para o índice

Teste de Mesa

```
a=1;
while (a<4){
        printf("%d\n",a);
        a=a+1;
        }
```

a	esc
1	1

Voltar para o índice

Teste de Mesa

```
a=1;
while (a<4){
        printf("%d\n",a);
        a=a+1;
        }
```

a	esc
1	1
2	

Voltar para o índice

Teste de Mesa

```
a=1;
while (a<4){
        printf("%d\n",a);
        a=a+1;
        }
```

a	esc
1	1
2	

Voltar para o índice

Teste de Mesa

```
a=1;
while (a<4){
      printf("%d\n",a);
      a=a+1;
      }
```

a	esc
1	1
2	2

E continua até concluir a simulação...

Voltar para o índice

- Uma estrutura de repetição, como o **while**, também é chamada de **laço de repetição** (ou **loop**)
- Cada repetição dos comandos dentro do **while** também é chamada de **iteração**.
  - Por essa razão algoritmos ou métodos que utilizam estruturas de repetição são ditos **iterativos**

Voltar para o índice

Ex: os trechos a seguir escrevem o que na tela? (teste de mesa)

a)

```
i=1;
while (i<=2){
    printf("%d\n",i);
    i=i+1;
}
```

b)

```
i=1;
while (i<=2){
    i=i+1;
    printf("%d\n",i);
}
```

Voltar para o índice

Ex: os trechos a seguir escrevem o que na tela? (teste de mesa)

c)

```
i=1;
while (i<3){
    i=i+1;
    printf("%d\n",i);
}
```

d)

```
i=1;
while (i<3){
    printf("%d\n",i);
    i=i+1;
}
```

Voltar para o índice

Ex: os trechos a seguir escrevem o que na tela? (teste de mesa)

e)

```
a=1;
b=2;
while (a<200){
a=a+b;
b=a+b;
printf("%d␣%d\n",a,b);
}
```

f)

```
a=100;
while (a>0){
    printf("%d\n",a%2);
    a=a/2;}
```

Voltar para o índice

g)

```
a=1234;
b=0;
while  (a>0){
    b=b*10+a%10;
    a=a/10;
}
printf("%d\n",b);
```

h)

```
a=100;
d=2;
while (a>1){
    if (a%d==0){
        printf("%d\n",d);
        a=a/d;}
    else d=d+1;
    }
```

Voltar para o índice

- I00000100 - Faça um algoritmo que escreva os números de 1 a 20.
- I00000200 - Faça um algoritmo que escreva todos os números pares entre 1 e 50.
  - Esse exercício pode ser resolvido por dois caminhos diferentes:

    a) Gerando todos os números entre 1 e 50 e escrevendo somente os que forem pares;
    b) Gerando e escrevendo somente os números pares.

Voltar para o índice

a)

```
a=1;
while (a<=50){
    if (a%2==0)
        printf("␣%d\n",a);
    a=a+1;
}
```

b)

```
a=2;
while (a<=50){
    printf("␣%d\n",a);
    a=a+2;}
// o incremento da variável
// que controla o laço pode
//ser diferente de 1
```

Voltar para o índice

- I00000300 - Faça um algoritmo que escreva todos os números entre 1 e 200 que são múltiplos de 11.
- (Não está no Portal) Faça um algoritmo que escreva 1000 vezes o nome do melhor time do Rio Grande do Sul
  - A solução para esse problema deve, de alguma forma, repetir 1000 vezes o comando *Escreval("Internacional!")*
  - Para isso, utiliza-se uma estrutura de repetição com uma variável que controle as 1000 repetições.

Voltar para o índice

```
i=1;
while (i<=1000){
    printf("Internacional!!!\n");
    i=i+1;
}
```

- Nesse algoritmo, a variável **i** tem a importante função de controlar o número de repetições, para que o comando de escrita seja executado exatamente 1000 vezes, nem uma a mais, nem uma a menos.
- À variável que controla o número de repetições dá-se o nome de **variável de controle**.

Voltar para o índice

- Um laço controlado por variável de controle deve conter 3 elementos:
  1. A definição do valor inicial (inicialização) da variável de controle, que deve obrigatoriamente ser feita fora do laço, e apenas uma vez, no início
  2. O incremento (ou decremento) da variável de controle
  3. Uma condição de saída que resulte falsa quando o número de iterações desejadas for completada.

Voltar para o índice

Alguns erros comuns ao implementar laços controlados são:

- Falta da inicialização da variável de controle
- Inicialização da variável de controle dentro do laço
- Falta do incremento (ou decremento) da variável de controle
- Condição de saída incompatível com o valor inicial e incremento da variável de controle, o que fará com que nunca termine a execução do laço (o famoso "**entrar em loop**")

Voltar para o índice

I00000400 - Faça um algoritmo que leia 10 números e escreva os que forem pares.

```
i=1;
while (i<=10){
        scanf("%d",&N);
        if (N%2==0)
                printf("%d ",N);
        i=i+1;
}
```

Voltar para o índice

Exercícios envolvendo contagem

- I00000600 - Faça um algoritmo que leia 30 valores e conte quantos estão no intervalo [10,20], escrevendo ao final essa informação.
  - A estrutura desse algoritmo é semelhante ao anterior, sendo necessária uma variável para controlar a contagem de 30 vezes e uma variável para armazenar o valor lido.
  - Além delas é necessário uma variável para controlar a contagem.
  - Essa variável (contadora) deve iniciar com 0 e, a cada vez em que for necessário contar, deve ser incrementada de 1.
  - Lembrando que o colchete representa um intervalo fechado (isto é, que inclui o limite do intervalo) e os parênteses representam um intervalo aberto (que não incluem o limite do intervalo)

Voltar para o índice

```
cont=0;
i=1;
while (i<=30){
        scanf("%d",&N);
        if (N>=10 && N<=20)
              cont=cont+1;
        i=i+1;
}
printf("cont=%d",cont);
```

Voltar para o índice

- I00000700 - Faça um algoritmo que leia 30 valores e conte quantos estão em cada um dos intervalos [0,25],(25,50], (50,75], (75..100], escrevendo ao final essa informação.
  - Semelhante ao anterior, mas é necessário um contador para cada um dos 4 intervalos
- I00000750 - Faça um algoritmo que leia um valor N. Escreva a seguir os números de 1 a N.
  - Semelhante ao I00000100, mas ao invés de escrever os valores até 20, escreve-se até N, sendo N um valor lido antes de entrar na repetição.

Voltar para o índice

I00000500 - Faça um algoritmo que leia números até que seja digitado um número negativo, e escreva todos que forem pares.

```
scanf("%d",&N); // lê o primeiro valor
while (N>0){
    if (N%2==0)
        printf("%d ",N);
    scanf("%d",&N); // lê o próximo valor de N
}
```

Voltar para o índice

Comando Break

- O comando break encerra a execução de um comando de repetição fazendo com que a execução continue a partir do comando imediatamente após o fim do comando de repetição.
- Normalmente é usado dentro de um comando condicional
- Ex: Ler números até que sejam digitados 10 números pares ou um número igual a zero:

```
while (quant<10){
        scanf("%d",&n);
        if (n==0) break;
        quant=quant+1;
}
```

- I00000800 - Faça um algoritmo que leia um valor N. Leia, a seguir, N valores e escreva todos que forem positivos.
- I00000900 - Faça um algoritmo que leia um valor N, e escreva todos os seus divisores.
  - Semelhante ao I750, mas ao invés de escrever todos os valores de 1 a N, testa-se para cada valor se é divisor de N e escreve-se somente os que são divisores.
- I00001000 - Faça um algoritmo que leia um valor N, e conte quantos divisores possui, escrevendo ao final essa informação.

Voltar para o índice

Comando **for**

- O comando for é um comando de repetição do tipo "enquanto", que contem a inicialização e o incremento da variável de controle.
- Sua sintaxe é:

  for (inicialização; condição; incremento)

  comando;

- onde **comando** é um comando simples ou um grupo de comandos, agrupados com chaves.
- **Inicialização** é um conjunto de atribuições iniciais (pode ser mais de uma, separadas por vírgulas).
- **Condição** é a expressão lógica que, enquanto for verdadeira, faz com que o comando continuará sendo executado.
- **Incremento** é um conjunto de comandos que será executado ao final de cada iteração.

Equivalência entre o "while" e o "for"

```
i=0;
while (i<10){
        comando
        i=i+1;
}
```

```
for (i=0;i<10;i=i+1)
        comando;
```

Observações

- Os campos de inicializações, condição e incrementos são separados entre si por ponto-e-vírgula.
- O campo **inicializações** do cabeçalho do for pode conter mais de uma inicialização.
  - Nesse caso, as inicializações são separadas por vírgulas.
  - O mesmo ocorre nos incrementos.
  - Ex: for (i=0,j=0; j<10; j=j+1,i=i+1)...
- O cabeçalho do for não precisa obrigatoriamente conter inicializações ou incrementos.
  - Nesse caso as listas de inicializações e incrementos ficam vazias, mas os ponto-e-vírgula entre os campos de inicializações, condição e incrementos são obrigatórios.
  - Ex: for (;i<10;) ...

Ex:

```
for (i=0 ; i<10 ; i=i+1)
/* para i de 0, ENQUANTO i for menor que 10, faça */
      printf ( "%d \n " , i ) ;
```

Ex. com dois comandos de inicialização e dois de incremento:

```
for (i=10 , j=1 ; i<10 ; i=i+1 , j=j-1 )
      printf ( "i = %d j = %d \n " , i , j ) ;
```

Ex. com mais de um comando na lista de comandos:

```
for (soma=0 , i=0 ; i<10 ; i++)
      {
      soma=soma+i;
      printf("soma = %d \n",soma);
      }
```

- A expressão de condição do comando for pode ser omitida (deixada em branco). Nesse caso a condição é avaliada como verdadeira.
- O for a seguir implementa um "laço infinito":

```
for (;;)
    {
    "faz alguma coisa infinitamente ou
    até ser executado um break"
    }
```

- Uma outra alternativa para um laço infinito é

  for (;1;) ou while(1)
- já que em C o valor 0 é interpretado como falso, e qualquer valor diferente de 0 é interpretado como verdadeiro.

- Cada repetição dos comandos dentro do comando for também é chamada **iteração** (daí o nome de "**programas iterativos**").
- Erros comuns:

1) Colocar um ponto-e-vírgula na linha do for:

```
for (i=0;i<10;i=i+1);
        printf("%d\n",i);
```

Assim como no **if**, esse ponto-e-vírgula faz com que o comando **for** termine, de modo que o comando seguinte está fora do **for** e será executado só uma vez.

2) Não colocar chaves no grupo de comandos:

```
for (i=0,soma=0;i<10;i=i+1)
    soma=soma+i;
    printf("soma=%d\n",soma);
```

Nesse exemplo, apenas o primeiro comando (soma=soma+i) está dentro do **for** e será repetido.

3) Tratar a condição do **for** como uma condição de repita (até que...) ao invés de uma condição de enquanto. Ex.

```
for (i=0;i==10;i=i+1)
// pensando que a condição é ATÉ QUE i=10
    printf("%d\n",i);
```

I00001100 - Faça um algoritmo que leia um valor N, e calcule e escreva a soma dos seus divisores.

```
int n,i,soma;
scanf("%d",&n);
soma=0;
for (i=1;i<=n;i++)
      if (n%i==0)
            soma+=i;
printf("%d",soma);
```

Pós-incremento e Pré-incremento

- i = i +1 pode ser escrito como i++ ou ++i
- i = i -1 pode ser escrito como i- - ou - -i
- A diferença entre o ++i (pré-incremento) e o i++ (pós incremento) é que, quando ocorre dentro de uma expressão, o valor utilizado na expressão é o valor antes do incremento ou depois do incremento.
- x=i++; // o incremento de i é feito após a atribuição
- x=++i; // o incremento de i é feito antes da atribuição

Formas reduzidas de atribuição:

- Quando um dos operandos de uma operação é a variável que receberá o resultado (p.ex: a = a + 3;) pode-se omitir o operando da expressão, resultando o comando a+=3;
- Da mesma forma pode-se utilizar essa versão reduzida para os operadores de subtração, divisão e multiplicação:
  - a*=3; equivale a a=a*3;
  - a-=3; a=a-3;
  - a/=3; a=a/3;
- de forma que uma forma de fazer um laço para escrever os números de 1 a 10 fica:

```
for ( i = 1 ; i <= 10 ; i+=1 )
    printf("%d\n",i);
```

Comandos break e continue:

- O comando **break** é utilizado quando se deseja interromper a execução das repetições, sem completá-las
- A execução do comando **break** faz com que a execução vá para o comando seguinte ao comando **for**, sem completar a iteração corrente.
- ex:

```
for (i=0;i<10;i++)
    {
    scanf("%d",&a);
    if (a==0) break;
    /* faz com que saia do for se for digitado 0 */
    }
```

Comando Continue

- O comando **continue** é utilizado quando se deseja interromper a execução da iteração atual, mas deseja-se continuar executando as próximas iterações do for.

Uso de valores numéricos como valores lógicos (verdadeiro, falso):

- A linguagem C permite utilizar valores numéricos como valores lógicos (verdadeiro ou falso).
- Nesse sentido, o valor 0 representa o valor lógico falso, e um valor diferente de 0 representa o valor lógico verdadeiro.
- Assim o for a seguir faz com que o programa fique indefinidamente dentro do for, até ser executado um comando break:

```
for (;1;)
{
/* comandos */
}
```

- I00001150 - Faça um algoritmo que leia um valor N inteiro e maior do que 1, e calcule e escreva o seu maior divisor (excetuando N).
  - Isso pode ser feito procurando o primeiro divisor a partir de N para baixo.

N=10	d=9	não é
N=10	d=8	não é
N=10	d=7	não é
N=10	d=6	não é
N=10	d=5	Achei!!!!

```
scanf("%d",&N);
d=N-1;
while (N%d!=0)
        d=d-1;
printf("%d",d);
```

- I00001155 - Faça um algoritmo que leia um valor N inteiro e maior do que 1, e calcule e escreva o seu menor divisor maior do que 1.
  - Mesma estrutura do anterior. Procura-se o primeiro divisor a partir de 2, incrementando d até encontrar um valor que seja divisor de N.

- I00001200 - Um número inteiro maior do que 1 é primo se ele possui como divisores somente o 1 e ele mesmo. Faça um algoritmo que leia um número e verifique se é primo escrevendo:
  - 1 - se o número é primo;
  - 0 - se o número não é primo.
- Dica:Pode-se verificar se um número é primo encontrando seu primeiro divisor maior que 1.
  - Se o primeiro divisor for o próprio número, ele é primo.
- Ou pode-se contar quantos divisores o número possui. Números primos possuem exatamente 2 divisores.
- Ou pode-se calcular a soma dos divisores. A soma dos divisores de um número primo N é igual a N+1.

- Buscando o cálculo do primeiro divisor do exercício 1155 ou contando a quantidade de divisores de N do exercício 1000 (em vermelho)

```
scanf("%d",&N);
d=2;
while (N%d!=0)
        d=d+1;
if (N==d)
        printf("%d",N);
```

```
scanf("%d",&N);
cont=0;
for (i=1;i<=N;i++)
        if (N%i==0)
                cont=cont+1;
if (cont==2)
        printf("%d",N);
```

Exercícios envolvendo busca de maior ou menor

- I00001400 - Faça um algoritmo que leia 10 números positivos e escreva ao final o maior deles.
  - Todos os exercícios envolvendo busca de maior ou menor tem a mesma estrutura.
  - Utiliza-se uma variável para armazenar o maior, e a cada novo valor compara-se o novo valor com o maior anterior, e se o novo valor for maior que o maior anterior, o novo valor substitui o maior.
  - algo como:

```
maior = 0;
for (i=1;i<= 10;i++){
        scanf("%d",&N);
        if (N > maior)
                maior = N;
}
printf("O maior é %d",maior);
```

- I00001500 - Faça um algoritmo que leia 10 números, positivos ou negativos, e escreva ao final o maior deles.
  - Cuidado com o valor inicial da variável que armazena o maior.
  - O que acontecerá se todos os valores lidos forem negativos?
  - Nesse caso qual pode ser o valor inicial da variável que guarda o maior?
  - Uma possibilidade poderia ser iniciar maior com um valor bem pequeno.
    - por exemplo: -999999...
    - mas qual o menor valor possível para uma variável? Depende da linguagem, do compilador...
    - e se todos os valores forem menores do que o valor inicial?

- Uma alternativa é ler o primeiro valor fora do laço e utilizar o primeiro valor como valor inicial para o maior.
  - Nesse caso, como um valor já foi lido fora do laço, dentro do laço deve ser lido um valor a menos.
  - Algo como:

```
scanf("%d",&maior);
for (i=1;i<= 9;i++){
        scanf("%d",&N);
        if (N > maior)
                maior = N;
}
printf("O maior é %d",maior);
```

- I00001600 - Faça um algoritmo que leia 10 números e escreva ao final o maior e o menor deles.
  - nesse caso deve-se ter uma variável para guardar o maior, e uma para guardar o menor.
  - ambas as variáveis podem começar com o primeiro valor lido, que é, ao mesmo tempo, o maior e o menor (já que é o único até o momento)

15) Dois números n_1 e n_2 são ditos amigos entre si se a soma dos divisores de n_1 (excluindo o próprio n_1) é igual a n_2, e a soma dos divisores de n_2 (excluindo o próprio n_2) é igual a n_1. Ex: 220 e 284. Faça um programa que leia 2 valores e verifique se são amigos entre si escrevendo uma mensagem apropriada.

19) Faça um programa que leia, para 10 pessoas, seu peso e altura e escreva o peso e a altura da pessoa mais alta.
20) O índice de massa corporal (IMC) de uma pessoa é dada pela relação peso/($altura^2$). Faça um programa que leia, para 10 pessoas, seu peso e altura e escreva o IMC, peso e altura da pessoa de maior IMC.

comando do...while

- tem o formato

```
do {
        comandos
} while (condição);
```

- e **condição** é uma condição de enquanto.
- A diferença em relação ao **for** e ao **while** é que a condição é avaliada após a execução dos comandos, de modo que os comandos são executados ao menos uma vez.
- Cabe salientar que, apesar de a condição ser avaliada ao final, ainda é um **while** e a condição é uma condição para continuar executando, e não uma condição de saída
- Ex:

```
i=1;
do {
      printf("%d\n",i);
      i=i+1;}
while (i<=10);
```

1) Faça um programa que leia números até ser digitado um número negativo e escreva ao final a sua soma (o número negativo não entra na soma). Faça uma versão com o while e uma com o do...while, com e sem o uso do comando break.

Exercícios envolvendo números primos

- I00001900 - Faça um algoritmo que leia 10 números e escreva os que forem primos.

```
for (i=1;i<=10;i++){
        scanf("%d",&N);
        if "N é primo"  // ??????
              printf("%d",N);
}
```

- Pode-se testar, para cada um dos 10 números, se tem exatamente 2 divisores:

```
for (i=1;i<=10;i++){
        scanf("%d",N);
        if "N tem só dois divisores" // tem que contar os
              printf("%d",N); // divisores antes
        }
```

- Ou pode-se testar se o primeiro divisor maior que 1 é o próprio número, nesse caso ele não tem nenhum outro divisor e é primo

```
for (i=1;i<=10;i++){
        scanf("%d",&N);
        if "o primeiro divisor de N é o próprio N" // tem
        // que encontrar antes o primeiro divisor de N
              printf("%d",N);
}
```

- Vamos tentar essa solução!!!

- Buscando o cálculo do primeiro divisor do exercício 1155 (em vermelho)

```
for (i=1;i<=10;i++){
        scanf("%d",&N);
        d=2;
        while (N%d!=0)
                d=d+1;
        if (N==d)
                printf("%d",N);
}
```

- Mas pode ter "while" dentro de "for"?

- Sim, pode ter **for** dentro de **for**, **for** dentro de **while**, **while** dentro de **for**...
- deve-se tomar cuidado com o lugar onde vai a inicialização das variáveis do laço interno
  - Ela deve estar dentro do laço externo mas fora do laço interno
  - Ex: o d=2; do exemplo anterior.

- I00001950 - Faça um algoritmo que escreva os 50 primeiros números primos.
  - Os 50 primeiros números primos são 2, 3, 5... (até completar 50)
  - Deve-se testar então todos os valores a partir de 2, escrevendo e contando cada um que for primo, até a contagem de primos chegar a 50

```
cont=0;
N=2;
while (cont<50){
    if "N é primo" {
        printf("%d",N);
        cont=cont + 1;
    }
    N=N+1;
}
```

- I00002000 - Faça um algoritmo que leia 2 números N1 e N2 e escreva a soma dos números primos entre N1 e N2 (incluindo N1 e N2 se algum deles for primo).

```
soma=0;
scanf("%d%d",&N1,&N2);
for (N=N1;N<=N2;N++){
      if "N é primo"
            soma=soma+N;
}
printf("%d",soma);
```

- I00002100 - Faça um algoritmo que leia 2 números N1 e N2 e escreva o produto dos números primos entre N1 e N2 (incluindo N1 e N2 se algum deles for primo).
- I00002200 - Faça um algoritmo que leia um número N e escreva os N primeiros números primos maiores que 100.

9) Faça um programa que leia um número inteiro N e escreva o maior número primo menor do que N.

10) Um número perfeito é o número que é igual à soma de seus divisores, exceto o próprio número (ex: $6 = 1 + 2 + 3$. Faça um programa que leia 10 números e verifique para cada um se é perfeito ou não, escrevendo uma mensagem apropriada.
11) Faça um programa que leia, para um funcionários: o valor que ganha por hora e 30 pares de valores (hora de entrada e hora de saída, inteiros, sem minutos) e calcule e escreva o quanto ganhou no mês. O funcionário não pode trabalhar mais de 23 horas seguidas e pode iniciar em um dia e terminar no dia seguinte.

- I00002600 - O MDC (máximo divisor comum) entre dois números n1 e n2 é o maior número que é divisor de n1 e de n2.
- Faça um algoritmo que leia dois números e escreva seu MDC.
  - Uma alternativa é procurar a partir de algum dos números, decrementando o MDC até encontrar um valor que seja divisor dos dois

a	b	MDC	
9	6	6	divide **b**, mas não divide **a**
9	6	5	não divide nenhum dos dois
9	6	4	não divide nenhum dos dois
9	6	3	Achei!!! divide **a** e **b**

- Outra alternativa é decompor os dois números em seus fatores primos (exercício 3000 e teste de mesa da primeira aula de repetição).
  - O MDC é o produto dos fatores primos comuns aos 2

a	b	divisores	
30	50	**2**	divide **a** e **b**, entra no MDC
15	25	3	divide somente o **a**. Não entra no MDC
5	25	**5**	divide **a** e **b**, entra no MDC
1	5	5	divide somente o **b**, não entra no MDC
1	1		fim da fatoração. MDC é 2 * 5 = **10**

- I00002700 - O fatorial de um número N (representado por N!) é o produto de todos os números de 1 a N. Assim, 4! = 1 x 2 x 3 x 4 = 24. Faça um algoritmo que leia um número N e escreva seu fatorial.
- I00002750 - O número de combinações de N diferentes objetos em grupos de P objetos é dado por $\frac{N!}{P!(N-P)!}$. Faça um algoritmo que leia uma quantidade N de objetos e o tamanho P dos grupos a serem formados, e calcule e escreva a quantidade de grupos que podem ser formados.

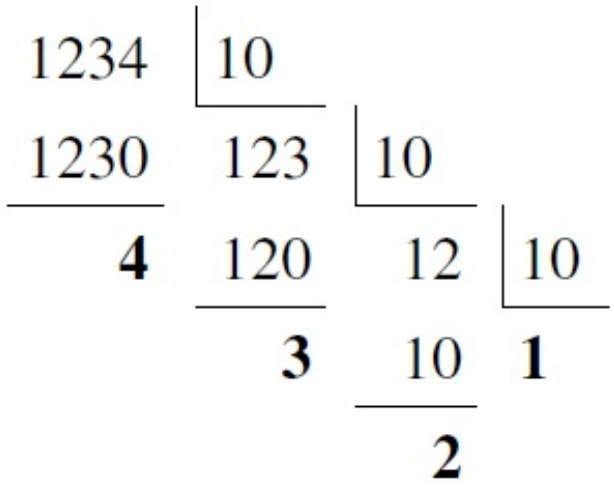
12) O MDC (máximo divisor comum) entre dois números n_1 e n_2 é o maior número que é divisor de n_1 e de n_2. Faça um programa que leia dois números e escreva seu MDC.
13) Faça um programa que leia 10 valores inteiros menores que 20 e, para cada um, calcule e escreva seu fatorial. O programa deve ignorar todos os valores maiores ou iguais a 20.
14) O MMC (mínimo múltiplo comum) entre dois números n_1 e n_2 é o menor número que é múltiplo de n_1 e de n_2. Faça um programa que leia dois números e escreva seu MMC.

Exercícios de separação de dígitos

- I00003100 - Faça um algoritmo que leia 10 números e para cada um escreva a soma dos seus digitos formantes.
  - Algoritmos de separação de dígitos de modo geral são baseados no cálculo do resto da divisão inteira por 10.
  - Essa operação se repete até que não sobrem mais dígitos. Algo como:

```
scanf("%d",N);
Soma=0;
while (N>0){
        Soma = Soma + N %10;
        N = N / 10;
}
printf("%d",Soma);
```

Ex: Separação de dígitos




```
scanf("%d",valor);
while (valor>0){
    digito= valor % 10;
    printf("%d",digito);
    valor = valor / 10;
    }
```

- I00005120 - Faça um algoritmo que leia um número inteiro e escreva quantas vezes ocorre o dígito 2.
- I00005150 - Faça um algoritmo que leia um número inteiro (máximo 5 casas - não maior que 99999) e mostre quantas ocorrências há de cada dígito de 0 a 9
- I00005100 - Faça um algoritmo que leia um número N até 100.000.000 e verifique se possui dígitos repetidos escrevendo:

  1 - se possuir dígitos repetidos;
  0 - se não possuir.

- I00003500 - Faça um algoritmo que escreva os primeiros 100 números cuja soma dos dígitos formantes é 10.
- I00003050 - Faça um algoritmo que leia um número e escreva seus dígitos na ordem contrária em que aparecem no número. Por exemplo, se o numero lido foi 32417, o algoritmo escreve 7,1,4,2,3.
- I00003200 - Faça um algoritmo que leia um número N e gere um número com os dígitos de N invertidos (teste de mesa da primeira aula de "Enquanto").

15) Faça um programa que leia um número N e efetue sua fatoração, escrevendo os fatores que o compõem. Ex: 28 = 2 x 2 x 7 60 = 2 x 2 x 3 x 5

19) Faça um programa que leia um número N até 100.000.000 e verifique se é palíndromo, ou seja, se se igual quando lido da esquerda para a direita e da direita para a esquerda. Ex: 13731. Escreva 1 se for palíndromo e 0 se não for.

20) Faça um programa que escreva os primeiros 100 dígitos cuja soma dos dígitos formantes é 10.
21) Uma pessoa aplicou um determinado valor em um banco. Sabe-se que o banco pagará 5% ao mês de juros. Faça um programa que leia o valor aplicado e calcule e escreva a quantidade mínima de meses necessários para que a pessoa obtenha R$1000,00 ou mais de rendimento. Por exemplo, se a pessoa aplicou 10.000 reais, ao final do primeiro mês terá 10.500, e ao final do segundo mês terá 11.025 e foram necessários 2 meses para alcançar um rendimento de mais de 1000 reais.

I00005600 - O CPF é formado por onze dígitos (999999999-99), dos quais os dois últimos são verificadores (controle), ou seja, a partir dos nove primeiros dígitos pode-se determinar os últimos dois. Considerando o CPF no formato abcdefghi-jk, onde cada letra representa um dígito, pode-se:

- calcular o primeiro dígito verificador (j), da seguinte forma:
  - somar: $10a + 9b + 8c + 7d + 6e + 5f + 4g + 3h + 2i$
  - encontrar o resto da divisão dessa soma por 11.
  - se o resto for igual a zero ou um, o dígito é zero, senão o dígito é onze menos esse resto.
- calcular o segundo dígito verificador (k):
  - somar: $11a + 10b + 9c + 8d + 7e + 6f + 5g + 4h + 3i + 2j$
  - encontrar o resto da divisão dessa soma por 11.
  - se o resto for igual a zero ou um, o dígito é zero, senão o dígito é onze menos esse resto.

Fazer um algoritmo que leia o CPF (somente primeiros nove dígitos) e escreva separadamente os verificadores (dois últimos).

Exercícios envolvendo datas

- I00003950 - Faça um algoritmo que leia duas datas, cada uma formada por dia, mês e ano, e escreva todas as datas entre as duas, incluindo a data inicial e a data final. Considere que ano bisexto é aquele divisível por 4. Considere também que a data inicial é menor ou igual à data final.
  - Exercícios iterativos envolvendo intervalos de datas de modo geral se baseiam em calcular a data do dia seguinte identificando se a data é último dia do mês, para identificar a virada de mês e de ano. Algo como:
  - if (d==31 && m==12) {d=1; m=1; a=a+1;}
    else if (d==31 ||
    d==30 && (m==4 || m==6 || m==9 || m==11) ||
    d==28 && m==2 && a%4!=0 || d==29 && m==2)
    {d=1; m=m+1;}
    else d=d+1;

- I00003900 - Faça um algoritmo que leia duas datas, cada uma formada por dia, mês e ano, e escreva o número de dias entre as duas, incluindo a data inicial e a data final. Considere que ano bisexto é aquele divisível por 4. E considere que abril, junho, setembro e novembro tem 30 dias, fevereiro tem 28 (29 em ano bissexto) e os outros meses tem 31.

- I00003960 - Faça um algoritmo que leia 4 valores d,m,a,ds, onde d,m,a representam uma data (dia,mês,ano) e ds representa um dia da semana (1-domingo, 2-segunda, 3-terça, 4-quarta, 5-quinta, 6-sexta, 7-sábado), e escreva as datas das próximas 3 sextas-feiras 13, a partir da data digitada, incluindo a própria data, se for o caso. Considere que ano bissexto é o ano divisível por 4.
- I00003970 - Faça um algoritmo que leia o mês e o ano de uma sexta-feira 13 e escreva o mês e ano das próximas 5 ocorrências de sexta-feira 13. Considere que ano bissexto é o ano divisível por 4.
- I00005200 - Faça um algoritmo que leia o dia e mês de uma data, e o dia da semana (1 - domingo, 2 - segunda... 7 - sábado) e escreva o dia da semana correspondente ao dia primeiro do mês digitado.

- I00003800 - Data juliana é o número de dias tanscorridos no ano (ex:236/1995). A faixa é de 1 a 365 (366 se o ano for bisexto). Escreva um algoritmo que leia uma data juliana (dia e ano) e a converta para o formato DD/MM/AAAA escrevendo a data. Considere que ano bisexto é aquele divisível por 4.

Exercícios de identificação de sequências

- I00004700 - Faça um algoritmo que leia 20 números e verifique qual a maior sequência estritamente crescente (isso é, cada número é maior que o anterior) dentre os números digitados, escrevendo ao final o primeiro e último valor da sequência.
- I00004850 - Faça um algoritmo que leia 15 números e identifique a maior sequência de números pares dentre os números lidos, escrevendo ao final o comprimento da sequência identificada, bem como o primeiro e o último valor da sequência. No caso de mais de uma sequência do mesmo comprimento, o algoritmo deve considerar a primeira delas.

- I00005400 - Faça um algoritmo que leia 20 números e escreva a maior sequência de valores iguais entre os números digitados.

I00006200 - Faça um algoritmo que para 10 cartas de baralho leia o seu valor (entre 1 e 13) e naipe (1 - ouros, 2 - copas, 3 - paus, 4 - espadas) em sequência, e escreva:

1 - se as cartas contem uma canastra;

0 - se não contem.

Considere como canastra apenas uma sequência crescente de 7 cartas do mesmo naipe de numeração contínua (cuja diferença entre duas cartas seja igual a 1). Dica:Considere que podem ocorrer cartas de mesmo número e naipe (uso de mais de um baralho). Nesse caso a carta de mesmo número e naipe não é contada, mas não quebra a sequência.

I00006201 - Faça um algoritmo que leia, para 10 cartas, seu naipe (1 a 4) e número (1 a 13), em ordem decrescente de naipe e número e escreva a maior sequência decrescente (mesmo naipe, numeração decrescente contínua) formada a partir da primeira carta. Caso a sequencia possua mais de 7 cartas, escreva apenas as primeiras sete cartas da sequencia. Se a sequencia tiver menos de 7 cartas, escreva a sequencia encontrada e um 0 ao final da sequencia.

I00006400 - Baseando-se na sequência de fibonacci (0,1,1,2,3...), verifique se uma sequência de cinco números, faz parte da sequência, devolvendo como resposta "0" se faz parte, ou "1" se não faz parte. Dica:Considere que números "repetidos" fazem parte da sequência proposta, ou seja, devem ser analisados individualmente como os demais números fornecidos.

- I00004800 - Faça um algoritmo que leia uma sequência de 20 números e escreva:

  0 - Se todos os números são iguais;
  1 - Se a sequencia é não-decrescente (cada número é maior OU IGUAL ao anterior);
  2 - se a sequencia é estritamente crescente (cada número é MAIOR que o anterior);
  3 - Se a sequencia é não-crescente (cada número é menor OU IGUAL ao anterior);
  4 - Se a sequencia é estritamente decrescente (cada número é MENOR que o anterior);
  5 - A sequencia é desordenada (há partes crescentes e partes decrescentes)

I00005450 - Faça um algoritmo que leia um número inteiro positivo N não maior que 1000000000000000000 (1E18), calcule e escreva o maior inteiro menor ou igual a sua raiz quadrada, ou seja, calcule e escreva a parte inteira da raiz quadrada no número informado.
I00005500 - Georg Cantor demonstrou que os números racionais são enumeráveis pela sequência:

$$\frac{1}{1}, \frac{1}{2}, \frac{2}{1}, \frac{1}{3}, \frac{2}{2}, \frac{3}{1}, \frac{1}{4}, \frac{2}{3}, \frac{3}{2}, \frac{4}{1}, \frac{1}{5}, \frac{2}{4} \ldots$$

Fazer um algoritmo que leia N (no intervalo de 1 até 1000000) e escreva o enésimo número dessa sequência (escreva o numerador e o denominador).

I00005501 - Georg Cantor demonstrou que os números racionais são enumeráveis pela sequência:

$$\frac{1}{1}, \frac{1}{2}, \frac{2}{1}, \frac{1}{3}, \frac{2}{2}, \frac{3}{1}, \frac{1}{4}, \frac{2}{3}, \frac{3}{2}, \frac{4}{1}, \frac{1}{5}, \frac{2}{4} \cdots$$

Fazer um algoritmo que leia N (no intervalo de 1 até $10000000000000000 = 1E18$) e escreva o enésimo número dessa sequência (escreva o numerador e o denominador).

I00005700 - Foram entrevistados 500 alunos de uma universidade. Para cada aluno entrevistado foi registrado o código do curso que ele frequenta (1: engenharia; 2: computação; 3: administração) e sua idade. Faça um algoritmo que processe tais dados fornecendo:

(a) número de alunos por curso;
(b) número de alunos com idade entre [20 25] anos por curso; e
(c) código do curso com a menor média de idade.

Dica:São necessários vários contadores.

I00005800 - O quociente da operação de divisão pode ser obtido subtraindo-se o divisor do dividendo. Da diferença, subtraímos novamente o divisor e assim sucessivamente até que a diferença seja menor do que o divisor. A quantidade se subtrações é o quociente. Assim, por exemplo:

-21 /4 =

21 - 4 = 17 (1)

17 - 4 = 13 (2)

13 - 4 = 9 (3)

9 - 4 = 5 (4)

5 - 4 = 1 (5)

Para o exemplo anterior, o número de subtrações é 5. Logo, o quociente é -5 (divisor e dividendo têm sinais diferentes). Faça um algoritmo que leia o dividendo (0, positivo ou negativo) e o divisor (positivo ou negativo) e escreva o quociente usando o algoritmo acima. NÃO deve ser usado o operador de divisão. Dica: Registre, de alguma forma, os sinais dos operandos e transforme-os para positivo.

I00005810 - A operação de multiplicação, quando o multiplicador é um número inteiro, nada mais é do que uma sucessão de somas. Assim, por exemplo,

- 4X5= 5+5+5+5= 20.

Escreva um algoritmo que leia o multiplicando e o multiplicador e, através de uma sucessão de somas, calcule e escreva o produto. O multiplicador é, necessariamente, um número inteiro. Porém, multiplicador e multiplicando podem ser 0 ou negativos.

- I00003600 - Uma pessoa aplicou um determinado valor em um banco. Sabe-se que o banco pagará 5% ao mês de juros. Fazer um algoritmo que leia o valor aplicado e calcule e escreva a quantidade mínima de meses necessários para que a pessoa obtenha R$1000,00 ou mais de rendimento. Por exemplo, se a pessoa aplicou 10.000 reais, ao final do primeiro mês terá 10.500, e ao final do segundo mês terá 11.025 e foram necessários 2 meses para alcançar um rendimento de mais de 1000 reais.

I00005900 - Um número N é dito um Meridiano Perfeito se existe um número M para o qual a soma dos números de 1 a N-1 é igual à soma dos números de N+1 a M. Assim, 6 é um Meridiano Perfeito porque 1+2+3+4+5=7+8. Faça um algoritmo que leia um número e verifique se é um meridiano perfeito, escrevendo 1, se o número é um meridiano perfeito, e 0, se não for.

I00006000 - Um número N é dito um Meridiano Perfeito se existe um número M para o qual a soma dos números de 1 a N-1 é igual à soma dos números de N+1 a M. Assim, 6 é um Meridiano Perfeito porque 1+2+3+4+5=7+8. Faça um algoritmo que leia um valor N, e escreva os N primeiros Meridianos Perfeitos maiores que 1.

I00006100 - Um número é dito regular, ou número de hamming (ou "5-smooth number" ou "ugly number") se tem como fatores primos apenas 2, 3 e 5, ou seja, é divisível somente por múltiplos de 2, 3 ou 5. Assim, os primeiros números regulares são 1 (é regular por definição, $2^0 * 3^0 * 5^0$), 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, 16, 18, 20, 24, 25, 27,30,32.... Faça um algoritmo que leia um número N, menor ou igual a 100, e escreva os N primeiros números regulares.

I00006500 - Leia um termo n que define o tamanho máximo dos lados de um triangulo retângulo podem assumir, sendo que os lados começam com tamanho 3, 4 e 5 (a,b,c). Em sequencia escreva todos os triângulos retângulos em que o valor máximo do lado é n e os três lados são inteiros. Cada triangulo deve ser escrito uma única vez em ordem crescente dos lados. Dica: O triângulo retângulo é dado como válido pela formula ($a^2 + b^2 = c^2$)

Exercícios envolvendo séries numéricas

- I00004000 - Faça um algoritmo que calcule e escreva a soma dos 100 primeiros termos da sequência a seguir: 1+3+5+7+9....
- I00004100 - Faça um algoritmo que calcule e escreva a soma dos 100 primeiros termos da sequência a seguir:

$$\frac{1}{1} + \frac{1}{3} + \frac{1}{5} + \frac{1}{7} + \frac{1}{9} + ...$$

- I00004200- Faça um algoritmo que leia um valor X e calcule e escreva a soma dos 100 primeiros termos da sequência a seguir:

$$\frac{X}{1} + \frac{X}{3} + \frac{X}{5} + \frac{X}{7} + \frac{X}{9} + ...$$

- I00004300 - Faça um algoritmo que leia um valor X e calcule e escreva a soma dos 100 primeiros termos da sequência a seguir:

$$\frac{X}{1} - \frac{X}{3} + \frac{X}{5} - \frac{X}{7} + \frac{X}{9} - \ldots$$

- I00004400 - Faça um algoritmo que leia um valor N e escreva a soma dos N primeiros termos da série a seguir:

$$\frac{1}{2} + \frac{1}{3} + \frac{1}{5} + \frac{1}{7} + \frac{1}{11} + \frac{1}{13} + \frac{1}{17} + \ldots$$

ou seja, a série onde os denominadores são os números primos

- I00004900 - Faça um algoritmo que leia a média e a quantidade de faltas para cada aluno de um grupo de 20 alunos, e escreva ao final o percentual de alunos aprovados. Os alunos serão considerados reprovados se não atingirem média 6.0. Também serão reprovados os alunos que tiverem mais de 20 faltas. Os alunos com uma quantidade de faltas entre 10 e 20 serão aprovados se obtiverem uma média mínima de 7.5.

Lista de problemas iterativos por categorias:

- somadores: 1100, 1300,2000,2100,2300,2700,2750,2800,4600
- contadores: 600, 700, 1000, 1610, 1950, 2200, 3500, 3700, 5700, 6800
- divisores: 900, 1000,1100, 1150, 1155, 1300, 2300
- primos: 1200, 1900, 1950, 2000, 2100, 2200, 2250, 2260, 2270, 2280, 2550, 4500, 6700
- datas: 3800, 3900, 3950, 3960, 3970, 5200
- separação de dígitos: 3050, 3100, 3200, 3300, 3400, 3500, 5000, 5100, 5120, 5150, 5160, 5170, 5300, 5600, 6800
- maior/menor: 1400, 1500, 1600, 1610, 1700, 1750, 1800, 6800
- sequências de valores: 4700, 4800, 4850, 5400, 6200, 6201, 6400
- somatório de séries: 4000, 5500
- manter últimos valores: 2500, 2525, 2550, 3350, 4500
- flags:

22) Faça um programa que leia 20 números inteiros e escreva quantos números são iguais ao menor número lido. Dica: Use um contador, incrementando-o ao encontrar um elemento igual ao menor corrente, e reinicializando-o ao encontrar um elemento menor do que o menor corrente.

- 26) Faça um algoritmo que calcule e escreva a soma dos 100 primeiros termos da sequência a seguir:
  1+3+5+7+9....
- 27) Faça um programa que calcule e escreva a soma dos 100 primeiros termos da sequência a seguir:

$$\frac{1}{1} + \frac{1}{3} + \frac{1}{5} + \frac{1}{7} + \frac{1}{9} + ...$$

- 28) Faça um programa que leia um valor X e calcule e escreva a soma dos 100 primeiros termos da sequência a seguir:

$$\frac{X}{1} + \frac{X}{3} + \frac{X}{5} + \frac{X}{7} + \frac{X}{9} + ...$$

- 29) Faça um programa que leia um valor X e calcule e escreva a soma dos 100 primeiros termos da sequência a seguir:

$$\frac{X}{1} - \frac{X}{3} + \frac{X}{5} - \frac{X}{7} + \frac{X}{9} - \ldots$$

- 30) Faça um programa que leia um valor N e escreva a soma dos N primeiros termos da série a seguir:

$$\frac{1}{2} + \frac{1}{3} + \frac{1}{5} + \frac{1}{7} + \frac{1}{11} + \frac{1}{13} + \frac{1}{17} + \ldots$$

ou seja, a série onde os denominadores são os números primos

- I00004500) Um número piramidal é um número que é igual à soma de 3 números primos consecutivos (ex: $15 = 3 + 5 + 7$). Faça um programa que leia um valor N e escreva os 10 primeiros números piramidais maiores ou iguais a N.
- 32) Faça um programa que leia 30 números e escreva ao final:
  - a média dos números pares digitados (soma dos pares dividido pela quantidade de pares);
  - quantos números são primos e
  - o produto (multiplicação) de todos os números lidos que são múltiplos de 5.

33) Faça um programa que leia 20 números e verifique qual a maior sequência crescente dentre os números digitados, escrevendo ao final o primeiro e último valor da sequência.

- 34) Faça um programa que leia uma sequência de 20 números e escreva:

  0 - Se todos os números são iguais;
  1 - Se a sequencia é não-decrescente (cada número é maior OU IGUAL ao anterior);
  2 - se a sequencia é estritamente crescente (cada número é MAIOR que o anterior);
  3 - Se a sequencia é não-crescente (cada número é menor OU IGUAL ao anterior);
  4 - Se a sequencia é estritamente decrescente (cada número é MENOR que o anterior);
  5 - A sequencia é desordenada (há partes crescentes e partes decrescentes)

35) Faça um programa que leia a média e a quantidade de faltas para cada aluno de um grupo de 20 alunos, e escreva ao final o percentual de alunos aprovados. Os alunos serão considerados reprovados se não atingirem média 6.0. Também serão reprovados os alunos que tiverem mais de 20 faltas. Os alunos com uma quantidade de faltas entre 10 e 20 serão aprovados se obtiverem uma média mínima de 7.5.

- 36) Faça um programa que leia um número inteiro qualquer e escreva os dígito desse número em ordem crescente do valor de cada dígito.
- 37) Faça um programa que leia um número inteiro qualquer e verifique se possui algum dígito repetido escrevendo ao final:
  - 0 - se ele não contem nenhum dígito repetido;
  - 1 - se ele contem algum dígito repetido.

39) Faça um programa que leia um número de 8 dígitos e calcule a soma ponderada dos seus dígitos, com peso 1 para o dígito menos significativo, peso 2 para o segundo dígito, peso 3 para o terceiro e assim por diante. Escreva a soma calculada.

42) Faça um programa que leia um número inteiro (máximo 5 casas - não maior que 99999). Mostre o dígito que mais se repete. Em caso de empate, mostre o menor deles.

- 43) Um número inteiro é dito ascendente se cada um dos seus algaristmos é maior do que o algarismo imediatamente à sua esquerda. Por exemplo, o número 3589. Faça um programa que leia um número inteiro e verifique se ele é ou não ascendente escrevendo:
  - 1 - se ele é ascendente;
  - 0 - se ele não é ascendente.

46) Faça um programa que leia 20 números e escreva a maior sequência de valores iguais entre os números digitados.
47) Faça um programa que leia um número inteiro positivo N não maior que 100000000000000000 (1E18), calcule e escreva o maior inteiro menor ou igual a sua raiz quadrada, ou seja, calcule e escreva a parte inteira da raiz quadrada no número informado.

- 51) Foram entrevistados 500 alunos de uma universidade. Para cada aluno entrevistado foi registrado o código do curso que ele frequenta (1: engenharia; 2: computação; 3: administração) e sua idade. Faça um programa que processe tais dados fornecendo:
  - (a) número de alunos por curso;
  - (b) número de alunos com idade entre [20 25] anos por curso; e
  - (c) código do curso com a menor média de idade.

  Dica:São necessários vários contadores.

52) O quociente da operação de divisão pode ser obtido subtraindo-se o divisor do dividendo. Da diferença, subtraímos novamente o divisor e assim sucessivamente até que a diferença seja menor do que o divisor. A quantidade se subtrações é o quociente. Assim, por exemplo;
-21 /4 =
21 - 4 = 17 (1)
17 - 4 = 13 (2)
13 - 4 = 9 (3)
9 - 4 = 5 (4)
5 - 4 = 1 (5)
Para o exemplo acima, o número de subtrações é 5. Logo, o quociente é -5 (divisor e dividendo têm sinais diferentes). Faça um programa que leia o dividendo (0, positivo ou negativo) e o divisor (positivo ou negativo) e escreva o quociente usando o algoritmo acima. NÃO deve ser usado o operador de divisão. Dica: Registre, de alguma forma, os sinais dos operandos e transforme-os para positivo.

53) A operação de multiplicação, quando o multiplicador é um número inteiro, nada mais é do que uma sucessão de somas. Assim, por exemplo,
4X5= 5+5+5+5= 20.
Escreva um programa que leia o multiplicando e o multiplicador e, através de uma sucessão de somas, calcule e escreva o produto. O multiplicador é, necessariamente, um número inteiro. Porém, multiplicador e multiplicando podem ser 0 ou negativos.
17 - 4 = 13 (2)
13 - 4 = 9 (3)
9 - 4 = 5 (4)
5 - 4 = 1 (5)
Para o exemplo acima, o número de subtrações é 5. Logo, o quociente é -5 (divisor e dividendo têm sinais diferentes). Faça um algoritmo que leia o dividendo (0, positivo ou negativo) e o divisor (positivo ou negativo) e escreva o quociente usando o algoritmo acima. NÃO deve ser usado o operador de divisão. Dica: Registre, de alguma forma, os sinais dos operandos e transforme-os para positivo.

57) Faça um programa que para 10 cartas de baralho leia o seu valor (entre 1 e 13) e naipe (1 - ouros, 2 - copas, 3 - paus, 4 - espadas) em sequência, e escreva: 1 - se as cartas contem uma canastra; 0 - se não contem. Considere como canastra apenas uma sequência crescente de 7 cartas do mesmo naipe de numeração contínua (cuja diferença entre duas cartas seja igual a 1).
Dica:Considere que podem ocorrer cartas de mesmo número e naipe (uso de mais de um baralho). Nesse caso a carta de mesmo número e naipe não é contada, mas não quebra a sequência.

58) Faça um programa que, baseando-se na sequência de fibonacci (0,1,1,2,3...), verifique se uma sequência de cinco números, faz parte da sequência, devolvendo como resposta "0" se faz parte, ou "1" se não faz parte. Dica:Considere que números "repetidos" fazem parte da sequência proposta, ou seja, devem ser analisados individualmente como os demais números fornecidos.
59) Faça um programa que leia um termo n que define o tamanho máximo que os lados de um triangulo retângulo podem assumir, sendo que os lados começam com tamanho 3, 4 e 5 (a,b,c). Em sequencia escreva todos os triângulos retângulos em que o valor máximo do lado é n e os três lados são inteiros. Cada triangulo deve ser escrito uma única vez em ordem crescente dos lados. Dica:O triangulo retângulo é dado como valido pela formula ($a^2 = b^2 + c^2$)

Lista de problemas iterativos por categorias:

- somadores: 1100, 1300,2000,2100,2300,2700,2750,2800,4600
- contadores: 600, 700, 1000, 1610, 1950, 2200, 3500, 3700, 5700, 6800
- divisores: 900, 1000,1100, 1150, 1155, 1300, 2300
- primos: 1200, 1900, 1950, 2000, 2100, 2200, 2250, 2260, 2270, 2280, 2550, 4500, 6700
- datas: 3800, 3900, 3950, 3960, 3970, 5200
- separação de dígitos: 3050, 3100, 3200, 3300, 3400, 3500, 5000, 5100, 5120, 5150, 5160, 5170, 5300, 5600, 6800
- maior/menor: 1400, 1500, 1600, 1610, 1700, 1750, 1800, 6800
- sequências de valores: 4700, 4800, 4850, 5400, 6200, 6201, 6400
- somatório de séries: 4000, 5500
- manter últimos valores: 2500, 2525, 2550, 3350, 4500
- flags:

Funções

- Funções são trechos de código que podem ser chamados a partir de diversos pontos diferentes do programa.
- Ao invés de escrever um trecho de código diversas vezes, escreve-se o código apenas uma vez e ele é chamado diversas vezes.
- A linguagem C possui diversas funções prontas. Essas funções encontram-se em bibliotecas, declaradas nas cláusulas #include.
- Assim #include $<$ *math.h* $>$ informa ao compilador que o programa utilizará funções matemáticas da biblioteca math.h.

- Além das funções pré-definidas na linguagem, o programador pode especificar suas próprias funções. Algumas vantagens do uso de funções são:
  - Reduzem o tamanho do programa, eliminando a repetição de código;
  - Melhoram a legibilidade do programa, "quebrando" um programa grande em partes menores, individualmente mais compreensíveis.
- Ao especificar uma função, o programador deve especificar que valores que ela irá receber, o tipo de valores que irá receber, e o tipo de valor que será retornado por ela (se for o caso). Os valores passados para uma função, que ela utilizará para efetuar os cálculos necessários, se chamam "parâmetros" ou "argumentos" da função.
- Os parâmetros enviados à função devem ser em mesma quantidade, tipo e ordem dos parâmetros definidos no cabeçalho da função.

- O formato geral de especificação de uma função é assim:

```
<tipo de valor de retorno> <nome da função> (<lista de parâmetros e tipo de cada um>)
    {
    <comandos>
    }
```

- Por exemplo, uma função que receba um valor inteiro e retorne o fatorial desse número pode ser escrita assim:

```
int fat (int n)
        {
        int i,f=1;
        for (i=1;i<=n;i++)
              f=f*i; // cálculo do fatorial
        return f; // especifica o valor que será retornado
        }
```

- E na chamada da função deve ser passado o parâmetro esperado.
- No caso da função fat, deve ser passado o valor de quem será calculado o fatorial.
- Uma chamada de função que retorne um valor deve ser feita sempre dentro de um comando onde o valor de retorno será imediatamente utilizado.
- Exemplos da chamada da função dentro do programa seriam algo como:

```
a=fat(4);
a=fat(b);
printf("O fatorial de %d = %d\n",b,fat(b));
```

- Funções podem chamar outras funções. Nesse caso, a função chamada deve estar declarada antes do ponto onde ela é chamada.
- Caso isso não seja possível, pode-se usar um **protótipo** que é simplesmente a linha do cabeçalho, sem o corpo da função, que deve estar desenvolvido mais adiante.

- Exemplo:

```
int poi()
        {
        printf("Oi");
        ptchau();
        }
int ptchau()
        {
        printf("Tchau");
        poi();
        }
```

- Como a função poi chama a ptchau, e a ptchau chama a poi, não é possível ordená-las de modo que a chamada da função fique após a declaração da mesma. A solução para isso é o uso de um protótipo de uma das funções:

- Exemplo:

```
int ptchau(); // protótipo da ptchau
int poi()
        {
        printf("Oi");
        ptchau();
        }
int ptchau()
        {
        printf("Tchau");
        poi();
        }
```

- Uma função pode utilizar variáveis declaradas dentro dela, chamadas **variáveis locais**, mas pode também utilizar variáveis declaradas fora de qualquer função. Essas variáveis são chamadas de **variáveis globais** e podem ser utilizadas para passar informações entre funções.
- Variáveis locais só podem ser referenciadas dentro da função onde são declaradas.
- Ex:

```
#include <stdio.h>
int i,b; // variáveis globais
int fat(int n)
{
int f=1; // f é local a fat
for (i=1;i<=n;i++) f=f*i;
// como i não foi declarado em fat,
// é utilizado o i global
return f;
}
int main()
{
int x; // x é local a main
scanf("%d",&a); // a é global
printf("O fatorial de %d = %d\n",a,fat(a));
}
```

- O comando **return** é utilizado para retornar a execução para o ponto onde a função foi chamada.
- Ele pode ser executado em qualquer ponto da função e causa o encerramento imediato da execução da função.
- Se a função deve retornar algum valor, o valor a ser retornado deve ser especificado no comando return.
- Em algumas situações uma função não deve retornar nenhum valor.
- Nesse caso ela deve ser especificada como função do tipo **void**.

- Por exemplo, uma função que receba um valor N e escreva os números de 1 a N:

```
void esc1aN (int N)
{
        int i;
        for (i=1;i<=N;i++)
                printf("%d ",i);
        return;
}
```

- E a chamada de uma função sem valor de retorno é feita como se fosse um comando da linguagem. Por exemplo, a função anterior seria chamada

```
esc1aN(5);

for (j=1;j<5;j++) esc1aN(j);
```

Exercícios:

- 1) Faça uma função que receba o dia, mês e ano de nascimento de uma pessoa, e o dia, mês e ano atual, e retorne sua idade.
- 2) Faça uma função que receba um valor N e retorne o primeiro divisor maior que 1 do valor N.
- 3) Faça uma função que receba um valor N e retorne a quantidade de divisores do valor N.
- 4) Usando uma das duas funções anteriores, faça uma função que recebe um valor N e verifique se ele é primo, retornando 1 se ele é primo, e 0 se não for primo. Use essa função para escrever um programa que escreva os 100 primeiros números primos.

- 5) Usando a função anterior, faça uma função proxprimo(int N) que receba um valor N e retorne primeiro valor primo maior que N. Use essa função para escrever os primeiros 100 números primos.
- 6) Faça um programa que escreva os 50 primeiros números primos.
- 7) Faça um programa que leia 2 números N1 e N2 e escreva a soma dos números primos entre N1 e N2 (incluindo N1 e N2 se algum deles for primo)..
- 8) Faça um programa que leia 2 números N1 e N2 e escreva o produto dos números primos entre N1 e N2 (incluindo N1 e N2 se algum deles for primo).

- 9) Faça um programa que leia um número N e escreva os N primeiros números primos maiores que 100.
- 10) Faça um programa que leia um número inteiro N e escreva o maior número primo menor do que N.
- 11) Faça um programa que leia um número inteiro N e escreva o menor número primo maior do que N.
- 12) Um número primo é um número natural maior que 1, que é divisível somente por 1 e por ele mesmo. Faça um programa que leia um número inteiro N e escreva o número primo mais próximo a ele. Se N for primo, considere que o mais próximo é o próprio N. Se houver dois números à mesma distância, escreva os dois em ordem crescente.

- 13) A conjectura de Goldbach diz que todo número par maior que 2 pode ser representado como a soma de dois números primos. Assim, 4=2+2, 6=3+3, 8=3+5... Faça um programa que leia um número N, par, e escreva, em ordem crescente, os dois números primos que o compõem. No caso de haver mais de um par de números (p.ex:20=3+17 e 20=7+13) escreva o par que tiver o menor número primo.
- 14) Faça uma funçao que receba o valor de dois resistores e um terceiro parâmetro definindo o tipo de associação dos resistores (0 para em série, 1 para em paralelo) e retorne o valor da associação dos dois resistores.

- 15) Faça um programa que escreva os primeiros 100 dígitos cuja soma dos dígitos formantes é 10.
- 16) O número de combinações de N diferentes objetos em grupos de P objetos é dado por $\frac{N!}{P!(N-P)!}$. Faça uma função que receba uma quantidade N de objetos e o tamanho P dos grupos a serem formados, e calcule e retorne a quantidade de grupos que podem ser formados.
- 17) O MDC (máximo divisor comum) entre dois números n1 e n2 é o maior número que é divisor de n1 e de n2. Faça uma função que receba dois números e escreva seu MDC.

Vetores

- Um vetor é um conjunto de variáveis independentes, onde a cada elemento é atribuído um índice que permite referenciá-la individualmente.
- A forma de referenciar um elemento de um vetor na linguagem C é nome_do_vetor[índice_do_elemento].
- O índice do primeiro elemento de um vetor na linguagem C é 0.
- Um elemento de um vetor e uma variável, e pode ser utilizado em qualquer situação que uma variável simples, especificando o nome do vetor e o índice do elemento a ser referenciado.

Ex. Considere um vetor V de 5 elementos do tipo int (V[0], V[1], V[2], V[3], V[4]):

```
V[0]=0;
V[1]=V[0]+3;
printf("%d\n",V[2]);
```

- Na declaração de um vetor deve ser especificado o tipo de cada elemento e o número de elementos:
- Ex: int V[5]; /* vetor de 5 elementos do tipo int, de índices 0 a 4 */
- Obs: Na declaração especifica-se o número de elementos, assim, como o índice do primeiro elemento é 0, o índice do último elemento do vetor é de um a menos do que o tamanho declarado.
- Pode-se inicializar os elementos de um vetor já na declaração:
- int V[5]={1,2,3,4,5}; // declara um vetor de 5 elementos (com índices de 0 a 4) do tipo int, inicializados com os valores 1,2,3,4,5.

Leitura e escrita de um vetor

```
int v[10],i;
for (i=0;i<10;i++)
      scanf("%d",&v[i]);
for (i=0;i<10;i++)
      printf("%d ",v[i]);
```

- V00001100 - Escrever um algoritmo que lê um vetor N(10) e o escreve. Troque, a seguir, o 1° elemento com o último, o 2° com o penúltimo, etc até o 5° com o 6° e escreva o vetor N assim modificado.
- V00001200 - Escrever um algoritmo que lê um vetor N(10) e o escreve. Troque, a seguir, cada elemento que estiver em posição ímpar (o primeiro, o terceiro...) pelo elemento da posição imediatamente a seguir. Escreva o vetor modificado.

- V00001250 - Escreva um algoritmo que lê em um vetor de 20 posições números positivos, até que o vetor esteja completo ou que tenha sido fornecido o valor 0 (zero). Após, escreva o vetor. A seguir, o algoritmo deve ler um número positivo qualquer e, caso ele se encontre no vetor, deve remover todas suas ocorrências, através de um deslocamento para a esquerda dos elementos que encontram-se à direita daquele a ser removido. Escreve o vetor modificado.

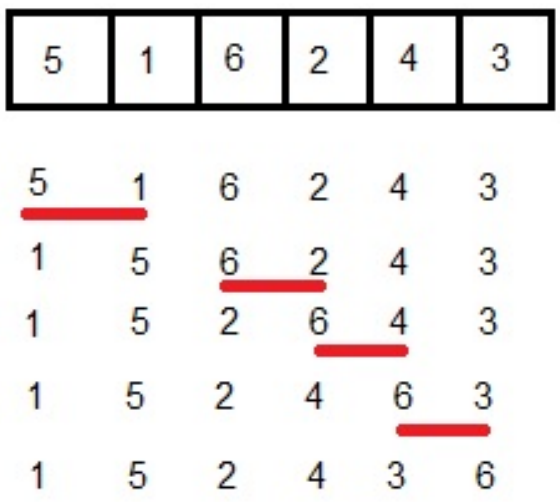
Índice Lista de Tópicos

- V00001650 - Faça um algoritmo que leia um número N e escreva os N primeiros números primos em ordem decrescente. Considere que N é no máximo igual a 100. Ex: Se N=5, escreva 11,7,5,3,2.
- V00001790 - Faça um algoritmo que leia valores inteiros entre 1 e 10 até que seja digitado um valor igual a 0, e escreva, ao final, quais dos valores entre 1 e 10 que não foram digitados nenhuma vez.

Índice Lista de Tópicos

Ordenação de Vetores

- V00001800 - Faça um algoritmo que leia um vetor X(10). Compare a seguir cada elemento com o elemento da posição seguinte, trocando-os entre si se o elemento de maior valor estiver antes do elemento de menor valor. O que se pode dizer sobre a ordem dos elementos dentro do vetor após essa operação?




```
for (i=0;i<9;i++)
    if (v[i]>v[i+1]){
        aux=v[i];
        v[i]=v[i+1];
        v[i+1]=aux;
    }
```

Ordenação de Vetores

- V00001900 - Faça um algoritmo que leia um vetor X(10) e ordene seus elementos em ordem crescente. Escreva o vetor ordenado. Dica: o que acontece se o procedimento descrito no exercício V00001800 for executado repetidamente sobre o vetor?

```
for (j=0;j<9;j++)
      for (i=0;i<9;i++)
            if (v[i]>v[i+1]){
                  aux=v[i];
                  v[i]=v[i+1];
                  v[i+1]=aux;
            }
```

Método da Bolha (Bubblesort)

- O algoritmo mostrado acima chama-se Método da Bolha, e normalmente é o primeiro ensinado em cursos de programação.
- Não é muito eficiente ou intuitivo, mas pela simplicidade é o mais usado para ordenar poucos valores.
- Pode-se obter uma melhora no seu desempenho substituindo o **for** externo por um **while**, para que ele termine assim que o vetor já estiver ordenado, o que pode ser identificado por um flag para sinalizar se foi feita alguma troca na passagem pelo vetor.

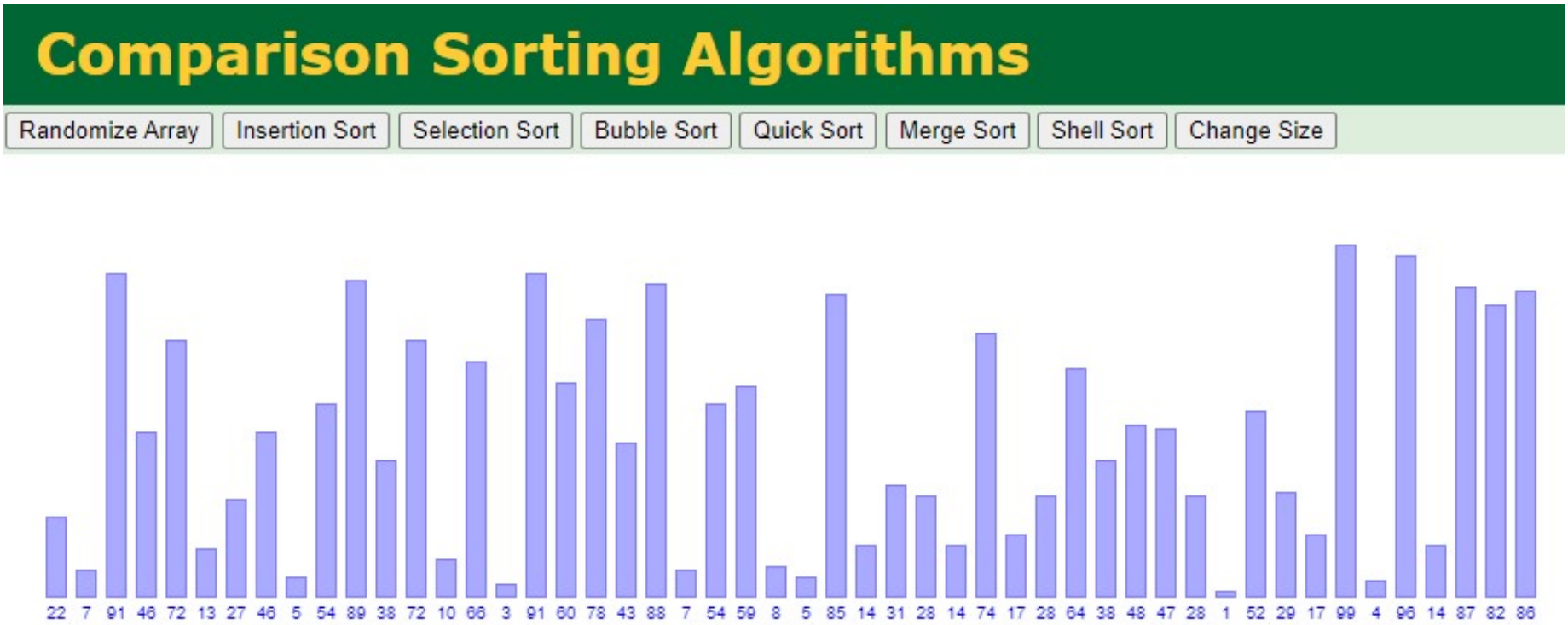
Índice Lista de Tópicos

- Algo como:

```
trocou=1;
while (trocou==1){
        trocou=0;
        for (i=0;i<9;i++)
                if (v[i]>v[i+1]){
                        aux=v[i];
                        v[i]=v[i+1];
                        v[i+1]=aux;
                        trocou=1;
                }
        }
```

- A animação abaixo disponível em ▸ bubblesort apresenta o *bubblesort* como uma dança húngara, com algumas otimizações. É interessante observar a animação e tentar identificar as otimizações:

- Outros algoritmos de ordenação da mesma ordem de custo computacional que o bubble sort são a ordenação por seleção e a ordenação por inserção.
- A animação abaixo, em ▸ Algoritmos de ordenação ilustra o funcionamento de alguns deles:



- V00002000 - Faça um algoritmo que leia, para cada pessoa de um conjunto de 10 pessoas, o seu peso (inteiro) e altura (real), e escreva a lista de pesos e alturas em ordem crescente de altura. Os dois dados referentes a cada pessoa devem ser lidos juntos (ou seja, o algoritmo não deve ler todos os pesos e em seguida todas as alturas)
  - Nesse exercício, pode-se manter um vetor para o peso e um para a altura, sendo que o que relaciona a altura e o peso à mesma pessoa é o fato de ocuparem a mesma posição nos dois vetores.
  - Para manter essa relação, a cada vez que duas alturas forem trocadas entre si, deve ser feita a mesma troca entre os pesos correspondentes

Ordenação por mais de uma chave

- Em algumas situações, os valores a serem ordenados são compostos por mais de um elemento, e mais de um elemento deve ser considerado para identificar a ordem relativa entre os valores.
- Identificando-se claramente a relação de ordem entre os valores, pode-se utilizar o Método da Bolha ou qualquer algoritmo de ordenação

Índice Lista de Tópicos

Ordenação por mais de uma chave

- V00003550 - Faça um algoritmo que leia 10 datas de aniversário, cada uma composta por dia e mes, e escreva as 10 datas em ordem cronológica crescente.
- Na comparação entre duas datas de aniversário, uma data é maior que a outra (ocorre após a outra) se o mês é maior ou, quando os meses são iguais, se o dia é maior.
  - Algo como: **if (m1>m2 || m1==m2 && d1>d2)**
  - E caso as datas não estiverem na ordem desejada, deve-se trocar todos os elementos que compõem a data (no caso, o dia e o mês)

- Algo como:

```
trocou=1;
while (trocou==1){
        trocou=0;
        for (i=0;i<9;i++)
                if (m[i]>m[i+1] || m[i]==m[i+1] && d[i]>d[i+1]){
                        aux=m[i];
                        m[i]=m[i+1];
                        m[i+1]=aux;
                        aux=d[i];
                        d[i]=d[i+1];
                        d[i+1]=aux;
                        trocou=1;
                }
        }
```

Ordenação por mais de uma chave

- V00003600 - Faça um algoritmo que leia 10 datas, cada uma composta por dia, mês e ano, e escreva as 10 datas em ordem cronológica crescente.
- V00003700 - Faça um algoritmo que para 10 cartas de baralho leia o seu valor (entre 1 e 13) e naipe (1 - ouros, 2 - copas, 3 - paus, 4 - espadas), e ordene-as em ordem crescente de naipe e, para cada naipe, em ordem crescente de valor. Escreva a lista de cartas ordenada.

Índice Lista de Tópicos

Vetores de contadores

- V00002500 - Faça um algoritmo que leia 30 valores inteiros entre 1 e 10 e escreva o número de vezes que ocorre cada valor.
- Esse exercício pode ser resolvido contando inicialmente os 1s, depois os 2s e assim por diante, até os 10s.

```
for (n=1;n<=10;n++){
        cont=0;
        for (i=0;i<30;i++){
                if (v[i]==n)
                        cont=cont+1;
        printf("O número %d ocorre %d vezes",n,cont);
        }
```

- Um problema da solução anterior é que, como é utilizado apenas um contador, é necessário repassar 10 vezes por todos os dados
- Uma alternativa seria a utilização de 10 contadores, um para cada número a ser contado.
- Dessa forma todas as contagens poderiam ser feitas simultaneamente

```
scanf("%d",&N);
if (N==1) cont1++;
else if (N==2) cont2++;
...
```

- Pode-se agrupar esses contadores em um vetor de contadores:

```
scanf("%d",&N);
if (N==1) cont[1]++;
else if (N==2) cont[2]++;
...
```

Índice Lista de Tópicos

- Como N "diz" qual a posição que deve ser incrementada, pode-se usar N diretamente como índice do vetor de contadores:

```
for (i=1;i<=10;i++)
    cont[i]=0;
for (i=1;i<=30;i++){
    scanf("%d",&N);
    cont[N]++;
    }
for (i=1;i<=N;i++)
    printf("␣%d␣ocorre␣%d␣vezes",N,cont[N]);
```

- V00003250 - Faça um algoritmo que leia, para 5 candidatos a deputado, o seu cadastro. Leia, a seguir, um número não determinado de votos (termina ao ser digitado 0) e escreva ao final o cadastro e número de votos de cada candidato (em ordem crescente de cadastro). Escreva também, ao final, o número de votos nulos (ou seja, votos que não são iguais ao cadastro de nenhum candidato).

Índice Lista de Tópicos

- V00003300 - Faça um algoritmo que leia, para 20 candidatos a deputado, seu cadastro, partido (inteiro entre 1 e 10) e número de votos, e escreva o total de votos de cada partido. Os 3 dados correspondentes a cada candidato devem ser lidos juntos, antes de passar para o próximo candidato.
- V00003350 - Faça um algoritmo que leia, para 20 candidatos a deputado, seu cadastro, partido (inteiro entre 1 e 10) e número de votos, e escreva, para os 10 partidos, o número do partido e o total de votos, em ordem decrescente de votação. Os 3 dados correspondentes a cada candidato devem ser lidos juntos, antes de passar para o próximo candidato.

- V00003400 - Faça um algoritmo que leia, para 20 candidatos a deputado, seu cadastro, partido (inteiro entre 1 e 10) e número de votos, e escreva o cadastro, partido e número de votos do candidato mais votado de cada partido, em ordem crescente de partido.
- V00003410 - Faça um algoritmo que leia, para 20 candidatos a deputado, seu cadastro, partido (inteiro entre 1 e 10) e número de votos, e escreva o cadastro, partido e número de votos do candidato mais votado de cada partido, em ordem decrescente de votação.

- V00003450 - Faça um algoritmo que leia, para 20 candidatos a deputado, seu cadastro, partido (inteiro entre 1 e 10) e número de votos, e escreva o cadastro, partido e número de votos de todos os candidatos, em ordem crescente de partido e, para o mesmo partido, em ordem crescente de número de votos.
- V00003500 - Faça um algoritmo que leia 20 números reais. Escreva a quantidade mínima de elementos cuja soma seja maior ou igual à metade da soma total. Dica: Classifique o vetor e some os primeiros (ou últimos) elementos até atingir a metade da soma total.

Lista de problemas de vetores por categorias (1):

- Preenchimento de vetores Preenchimento: 320, 1500, 1650, 2200
- Encontrar maior/menor elemento em vetor Maior/Menor: 800, 900, 1010, 3350
- Pesquisa de valor em vetores Pesquisa: 1050, 1400, 1450, 1600, 3250
- Operações sobre conjuntos em vetores Conjuntos: 1500, 1501, 1700, 1710, 1720, 1730
- Trocas de elementos em vetores: 1100, 1200, 1800

Lista de problemas de vetores por categorias (2):

- Ordenação Ordenação: 1900, 2000, 3250, 3350
- Ordenação por várias chaves Ordenação 2: 2550, 3550, 3600, 3700
- Vetores de contadores Contadores: 1790, 2500, 3250, 3300, 3350, 3750, 250
- Identificação de sequências: 3750, 4000
- Identificação dos últimos valores: 6200
- Deslocamento de valores em vetores: 2600, 6900, 7300, 1250

Busca de um valor em um vetor

- Uma operação comum em exercícios com vetores é a busca de um valor em um vetor. Para fazer isso pode-se comparar o valor a ser pesquisado com todos os valores do vetor e contar quantas vezes ele ocorre, ou usar um **flag**. Por exemplo, para procurar N em um vetor V[100].

```
tem=0;
for (i=0;i<100;i++)
      if (N==v[i])
            tem=1;
```

Flags

- Um **flag** ou variável sinalizadora é uma variável utilizada para identificar, em um ponto do algoritmo, se alguma situação ocorreu antes no algoritmo.
- É como se fosse um contador, mas limitado ao valor 1.
- Da mesma forma que um contador, ele é inicializado com 0 e, quando ocorre a situação que se busca identificar, ele recebe 1 e mantem esse valor.
  - Posteriormente testando-se o seu valor (0 ou 1) pode-se saber se a situação buscada ocorreu.
- No exemplo anterior, o flag **tem** é utilizado para testar mais adiante se N foi encontrado no vetor ou não.

Índice Lista de Tópicos

- Um problema da versão anterior é que o algoritmo continua procurando o valor mesmo após ter encontrado.
- Uma alternativa é sair do laço quando o valor for encontrado, o que pode ser feito alterando a condição do for, fazendo com que a procura termine assim que o valor for encontrado ou quando chegar no final.
- Algo como:

```
tem=0;
for (i=0;i<10 && tem==0;i++)
    if (N==v[i])
        tem=1;
```

- Outra alternativa é colocar a comparação de N e cada elemento do vetor direto na condição do **enquanto**
  - Deve-se tomar cuidado para que a condição não teste uma posição inexistente do vetor, como a posição v[10].
- Ao sair do enquanto, pode-se testar se o valor foi encontrado testando-se o valor de i.

```
for (i=1;i<10 && N!=v[i]; i++);
if (i<10) printf("Achei!!!");
```

Como descobrir se um valor **não** está em um vetor

- O processo é o mesmo de descobrir se o valor **está** no vetor. A diferença é que, ao sair do laço, testa-se se o flag está desligado, significando que nenhum valor igual foi encontrado no vetor.
- Soluções envolvendo testes do tipo "se n¡¿v[i]..." em geral não funcionam porque testam se há no vetor **um** elemento diferente de n, e não se n é diferente de **todos** os elementos do vetor

```
tem=0;
for (i=0;i<10 && tem==0;i++)
    if (N==v[i])
        tem=1;
if (tem==0)
    printf("O valor não está no vetor");
```

- V00001050 - Faça um algoritmo que leia um vetor de 10 elementos. Leia, a seguir, um valor N e verifique se o valor aparece no vetor escrevendo:

  0 - se o valor N não aparece no vetor;

  1 - se o valor N aparece no vetor
- V00001450 - Faça um algoritmo que leia um vetor de 10 elementos. Leia, a seguir, 10 valores N e, para cada um, verifique se o valor aparece no vetor escrevendo:

  0 - se o valor N não aparece no vetor;

  1 - se o valor N aparece no vetor

- V00001300 - Escrever um algoritmo que lê um vetor G(10) contendo o gabarito de uma prova. Leia a seguir, para cada aluno de um conjunto de 5 alunos suas 10 respostas e verifique quantas questões acertou, escrevendo para cada aluno o número de acertos.
- V00001400 - Faça um algoritmo que leia 10 valores e verifique se algum dos valores aparece repetido. Escreva 1 se algum valor aparece repetido e 0 se não houver nenhum valor repetido.

- V00001600 - Faça um algoritmo que leia um vetor V(5), com os valores sorteados em um sorteio de Loto. Leia, a seguir, para um conjunto de 5 apostadores, seus 5 palpites e escreva, para cada um, o número de acertos que teve.
- V00001500 - Escrever um algoritmo que lê um vetor X(10) e, após, leia um vetor Y(10). Crie, a seguir, um terceiro vetor Z com os elementos que aparecem nos dois vetores (intersecção). Os elementos devem aparecer no vetor Z na mesma ordem em que aparecem no vetor X. Considere que não há repetição de valores dentro do mesmo vetor. Escreva o vetor Z (apenas as posições que foram preenchidas).

- V00001501 - Escrever um algoritmo que lê um vetor X(10) e, após, leia um vetor Y(10). Crie, a seguir, um terceiro vetor Z com os elementos que aparecem em um X ou em Y (união); elementos que aparecem em X e Y simultaneamente devem aparecer apenas uma vez em Z. Os elementos devem aparecer no vetor Z na mesma ordem em que aparecem no vetor X e Y. Considere que não há repetição de valores dentro do mesmo vetor. Escreva o vetor Z (apenas as posições que foram preenchidas).

Índice Lista de Tópicos

- V00001700 - Escrever um algoritmo que lê 2 vetores X(10) e Y(10), e escreva os elementos que aparecem no vetor X e não aparecem no vetor Y (diferença de conjuntos). Escreva os valores na ordem em que eles aparecem no vetor X. Os dois vetores devem ser lidos separadamente (em primeiro lugar, todo o vetor X, após, o vetor Y).

Índice Lista de Tópicos

- V00001710 - Escrever um algoritmo que lê 3 vetores A[1..10], B[1.10] e C[1..10] e escreve os elementos que estão em A e B (interseção) mas não estão em C. Escreva os valores na ordem em que eles aparecem no vetor A. Os três vetores devem ser lidos separadamente (em primeiro lugar, todo o vetor A, após, o vetor B e por fim o vetor C).

- V00001720 - Escrever um algoritmo que lê 3 vetores A[1..10], B[1.10] e C[1..10] e escreve os elementos que são comuns aos três vetores (intersecção). Escreva os valores na ordem em que eles aparecem no vetor A. Os três vetores devem ser lidos separadamente (em primeiro lugar, todo o vetor A, após, o vetor B e por fim o vetor C).

Índice Lista de Tópicos

- V00001730 - Escrever um algoritmo que lê 2 vetores A[1..10] e B[1.10] e escreve os elementos que estão somente em A ou somente em B. Escreva os valores na ordem em que eles aparecem no vetor A, e em seguida os que aparecem no vetor B. Os dois vetores devem ser lidos separadamente (em primeiro lugar, todo o vetor A e, após, o vetor B).

Índice Lista de Tópicos

Preenchimento de vetores

- Valores são colocados no vetor só quando atendem a uma condição. O algoritmo deve controlar qual a próxima posição disponível do vetor ou qual a última que foi preenchida até o momento.
- Exemplo: Leia valores até serem digitados 10 valores maiores ou iguais a zero. Os negativos devem ser descartados.

```
ult_ocup=-1;
while (ult_ocup<9) {
    scanf("%d",&n);
    if (n>=0){
        ult_ocup=ult_ocup+1;
        v[ult_ocup]=n;
    }
}
```

```
prox_disp=0;
while (prox_disp<=9){
    scanf("%d",&n);
    if (n>=0){
        v[prox_disp]=n;
        prox_disp=prox_disp+1;
    }
}
```

- Exercícios 320, 1500, 1650, 2200

Encontrar maior/menor elemento de vetor

- Como em todos os problemas envolvendo encontrar o maior valor de um conjunto, deve-se manter uma variável para guardar o maior valor e comparar cada elemento do vetor com essa variável, substituindo-a a cada vez que aparecer um maior
- Pode-se inicializar essa variável com 0, se os valores são positivos, ou com o primeiro valor do vetor, caso não se tenha informação sobre o intervalo dos valores no vetor

```
maior=v[0];
pos_maior=0; (caso queira-se a posição do maior)
for (i=0;i<10;i++)
    if (v[i]>maior){
        maior=v[i];
        pos_maior=i;
    }
```

Encontrar **menor** elemento de vetor

- Análogo ao problema de encontrar o maior valor. A principal diferença é que, a variável que guardará o menor deve ser inicializada com um valor grande (maior do que os valores do vetor) ou com o primeiro valor do vetor
- E a comparação entre o elemento do vetor e o menor é trocada. O menor é substituído a cada vez que o elemento do vetor for **menor** que ela.

```
menor=v[0];
pos_menor=0; (caso queira-se a posição do menor)
for (i=0;i<10;i++)
    if (v[i]<menor){
        menor=v[i];
        pos_menor=i;
    }
```

- Exercícios 800, 900, 1010, 3350

Operações sobre conjuntos de valores

- De modo geral verifica-se para cada elemento de um vetor se ele está ou não está (slide 317) em outro vetor.
- Ex: diferença entre vetores A e B (elementos que estão em A e não estão em B):

```
for (i=0;i<10;i++){
        tem=0;
        for (j=0;j<10;j++)
              if (A[i]==B[j])
                    tem=1;
        if (tem==0) printf("%d",A[i]);
}
```

Mensagens de erro do Dev C++

- **'a' undeclared (first use this function)**
  - A variável 'a' não foi declarada
- **expected ';' before "b"**
  - Faltou um ponto-e-vírgula no comando anterior
- **Warning - Converting to 'int' from 'double'**
  - Está sendo atribuído um valor real a uma variável inteira
  - Um 'Warning' é uma mensagem de advertência, mas que não impede o programa de ser executado. Normalmente indicam situações de erro em potencial.

O que é Arduino

- É uma plataforma de software e hardware livre.
- Foi desenvolvida em 2005 com o objetivo de criar uma plataforma para o desenvolvimento de projetos educativos.
- É uma placa baseada no uso do microprocessador ATMEL.
- Para sua programação utiliza um ambiente próprio, cuja linguagem é baseada em C e algumas extensões de C++.

Por que Arduino?

- É uma solução de baixo custo em comparação a outras placas existentes no mercado.
- Podem ser desenvolvidos projetos de eletrônica, robótica e automação.
- Todos seus esquemáticos e componentes eletrônicos se encontram na Internet permitindo uma fácil implementação na criação de projetos.
- É um sistema multiplataforma rodando em Linux, MacOS e WindosNa.

Família Arduino

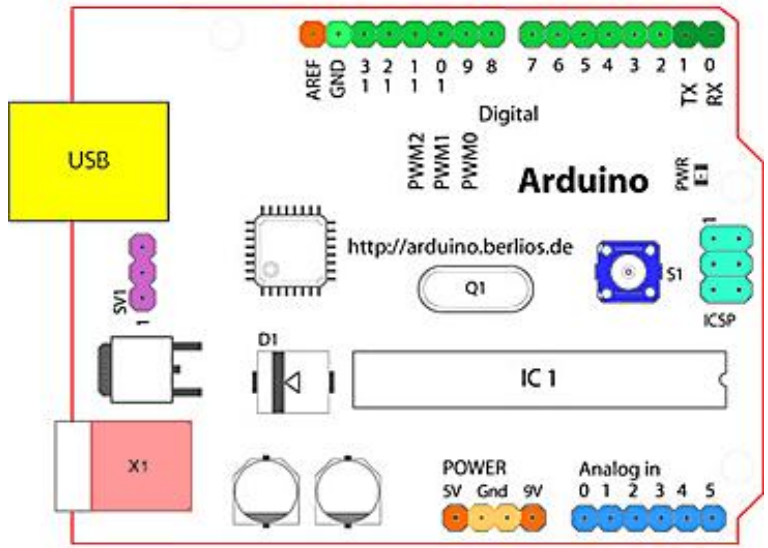
- Há diversos tipos de hardware e Shields (componentes para auxílio ou para incrementar o projeto), cada uma delas tem um "codinome" e existe uma variedade que se adapta de acordo com a necessidade dos projetos (colocar figurinhas e exemplos de shields).
- Algumas delas são:
  - Arduino UNO
  - Arduino Duemilanove
  - Arduino Ethernet
  - Arduino Mega

Estrutura do Hardware

- Arduino UNO
  - 32 Kb de memória flash (mantém o programa quando desligada)
  - 2 Kb de memória SRAM (variáveis)
  - 1 kb de memória EEPROM

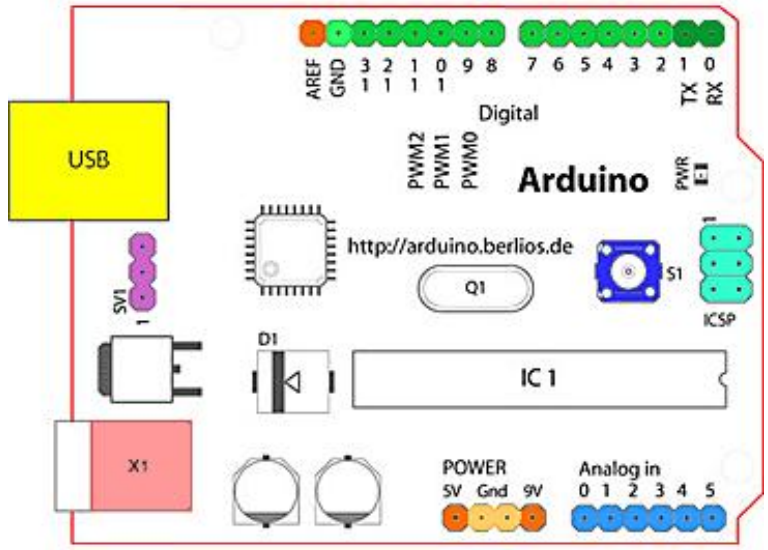
Estrutura do Hardware

- Digital Ground (verde claro)
- E/S digitais 2 a 13 (verde)
- Digital Pins 0-1/Serial In/Out - TX/RX (Verde Escuro) - Estes Pinos não podem ser usados para digital i/o (digitalRead e digitalWrite) se estiver usando comunicação serial.
- Botão de Reset - S1 (Azul Escuro)



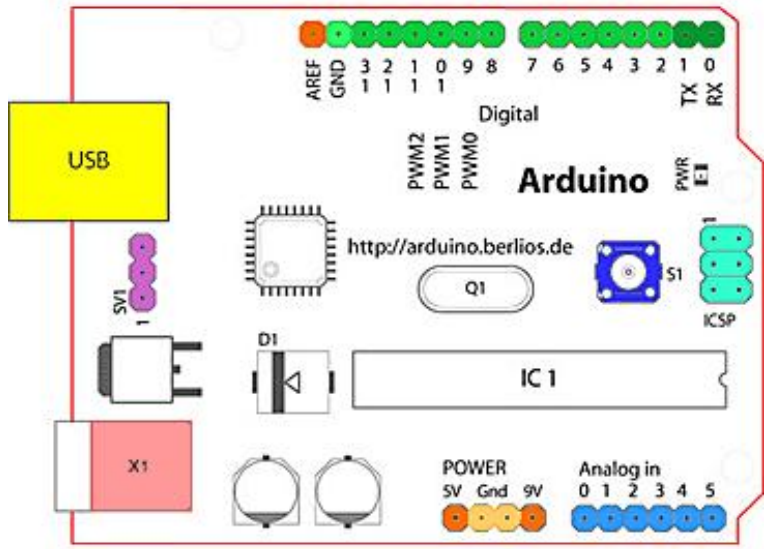
Estrutura do Hardware

- Entradas analógicas nos pinos 0-5 (Azul Claro)
- Pinos de saída de alimentação (laranja escuro: 5 v e 9 v) e Ground (Laranja claro)
- Suprimento Externo de energia In (9- 12VDC) - X1 (Rosa)



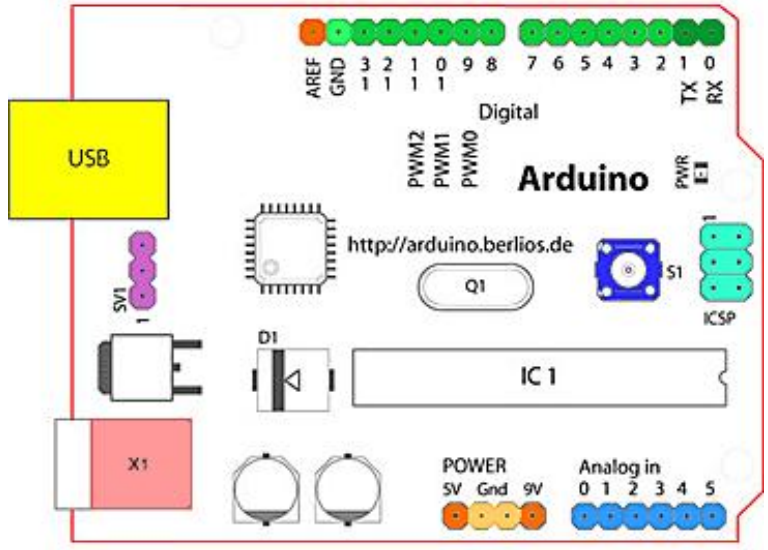
Estrutura do Hardware

- SV1- Jumper: Determina alimentação da USB ou Externa X1 (Roxo)
- USB: Usada para gravar os programas; Comunicação serial entre placa e computador; Alimentação da placa (Amarelo)



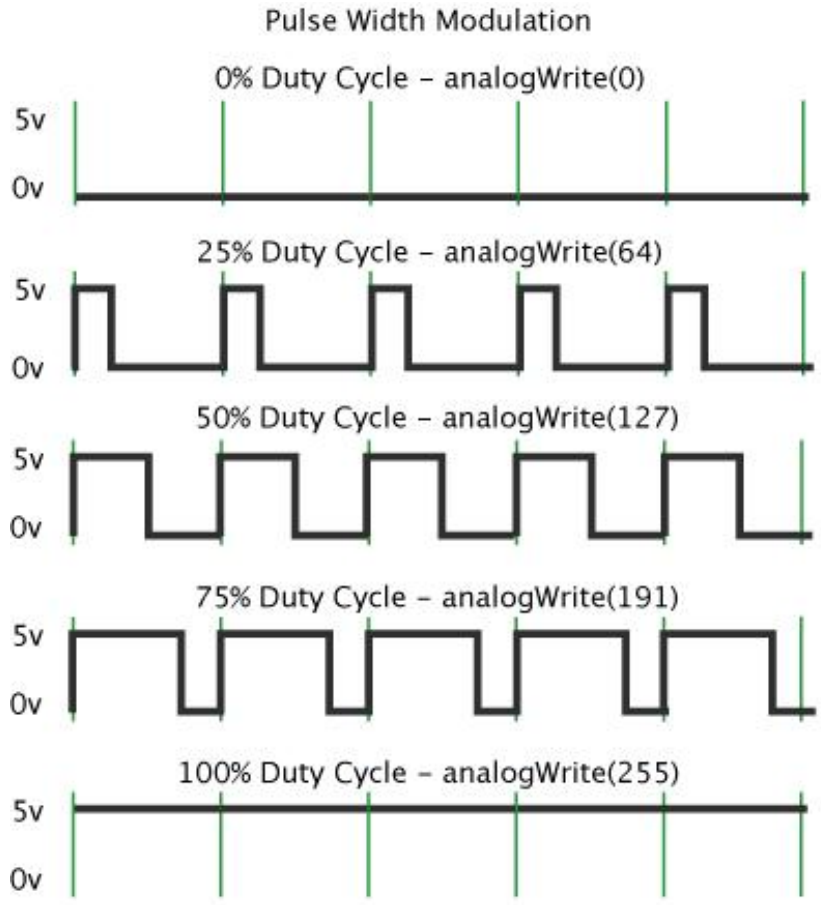
Estrutura do Hardware

- SV1- Jumper: Determina alimentação da USB ou Externa X1 (Roxo)
- USB: Usada para gravar os programas; Comunicação serial entre placa e computador; Alimentação da placa (Amarelo)

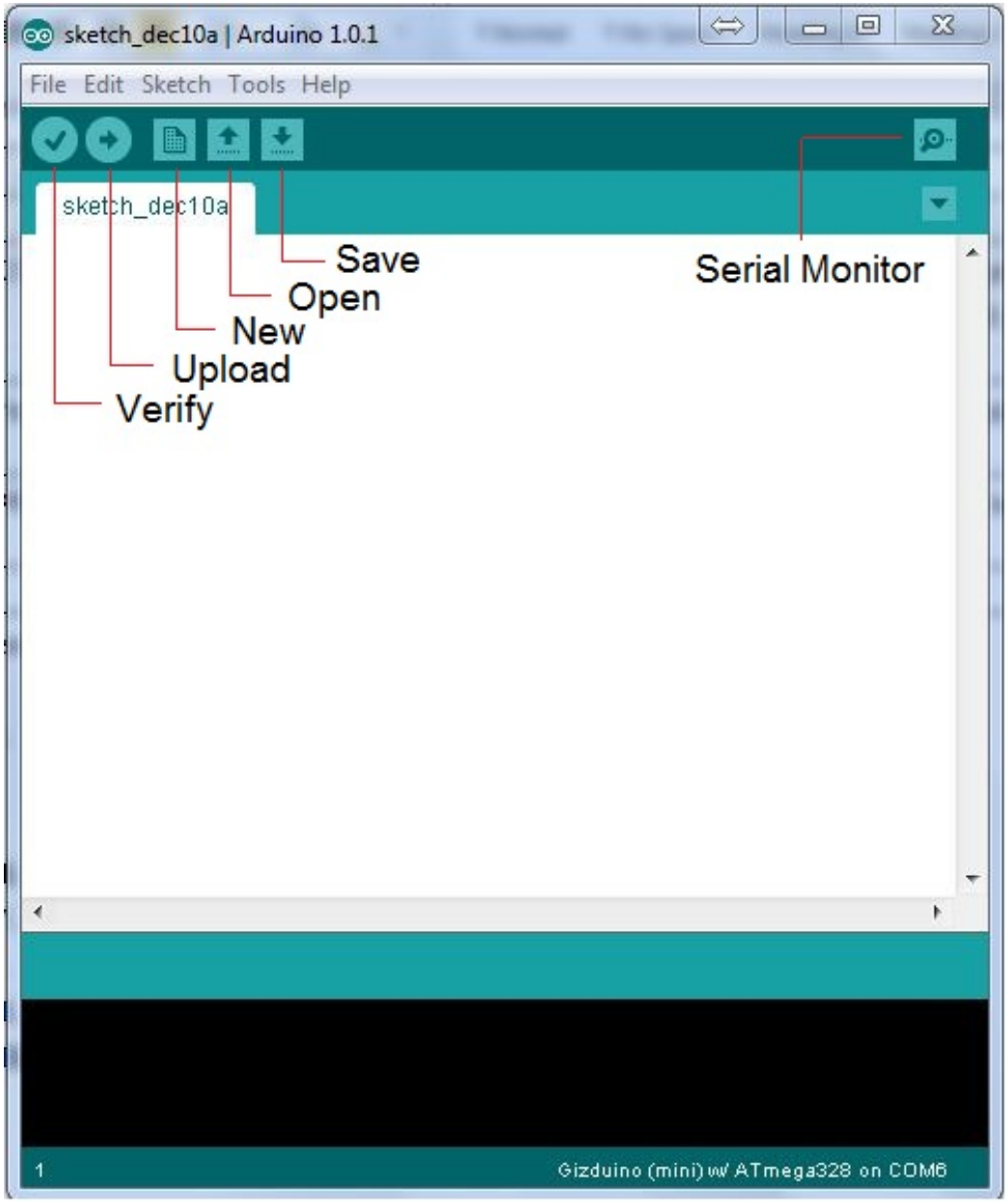


"Saídas analógicas"

- O arduino pode simular saídas analógicas através de 6 das saídas digitais através de pwm



Ambiente de Programação do Arduino



- Um programa para o arduino é chamado de *sketch*
- Um *sketch* é formado de duas funções principais:
  - setup( )
    - é executada uma única vez ao iniciar a execução do programa
    - é utilizada para inicialização de portas e variáveis
  - loop( ) - é chamada repetidamente enquanto o arduino estiver alimentado

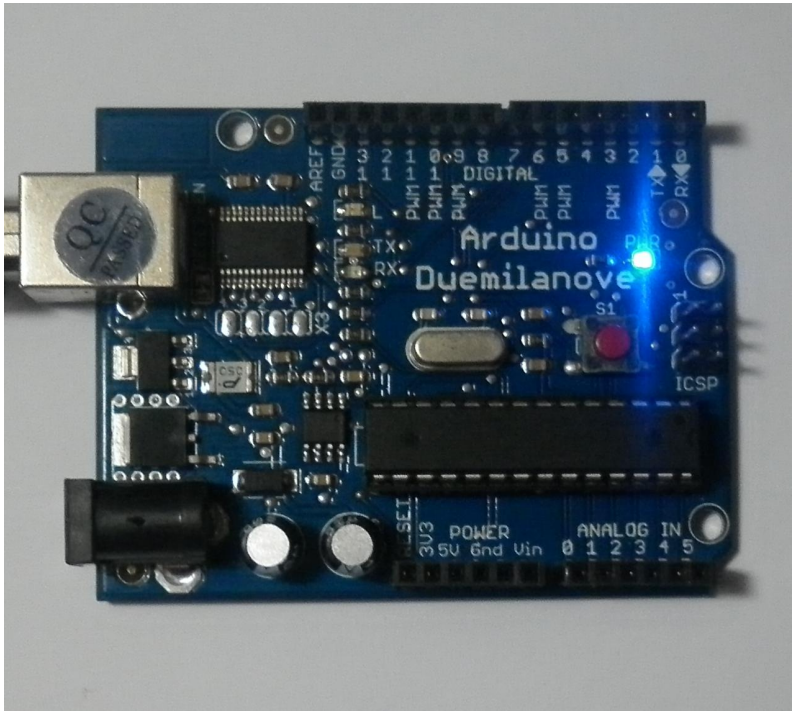
Primeiro exemplo - Led piscante

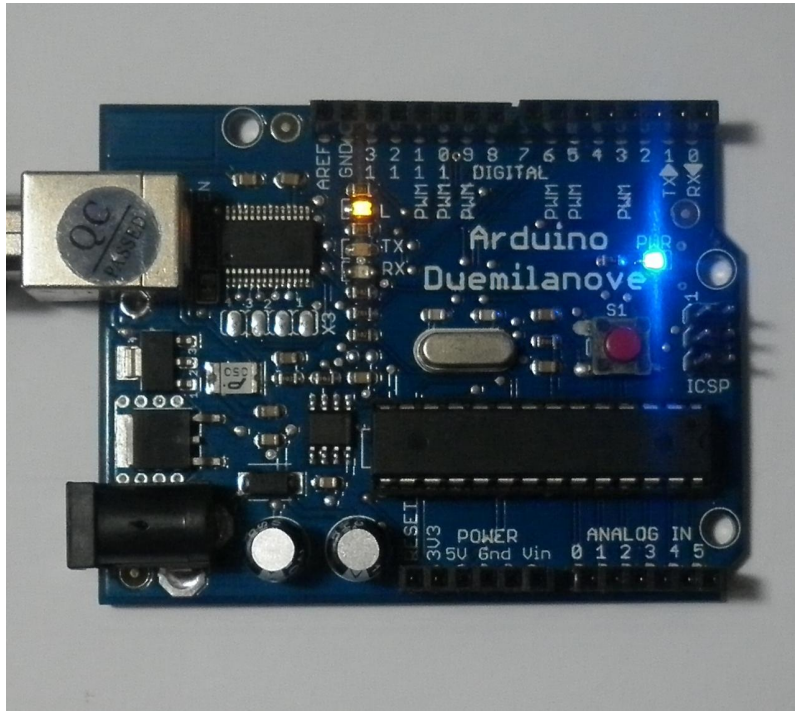
- Mantem o led aceso por um segundo e apagado por um segundo alternadamente.
- Utiliza a porta 13 que já possui um led conectado internamente a ela.
- Abra a IDE e digite o seguinte código:

```
void setup() {
pinMode(13, OUTPUT);
}

void loop() {
digitalWrite(13, HIGH);
delay(1000);
digitalWrite(13, LOW);
delay(1000);
}
```

Resultado obtido



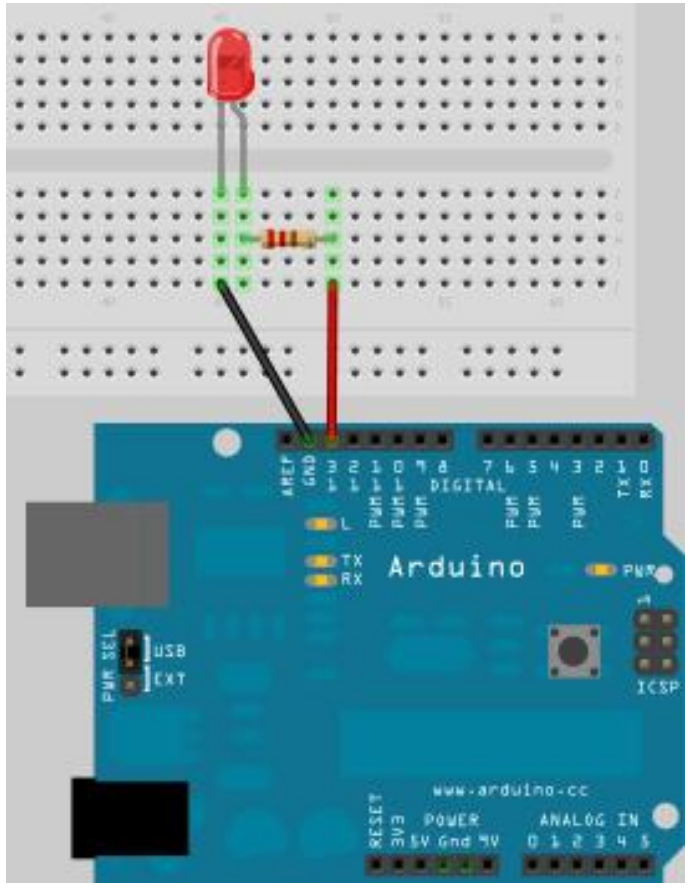


Funções utilizadas

- pinMode(pino, INPUT/OUTPUT)
  - configura o pino para entrada ou saída
- digitalWrite(pino, HIGH/LOW)
  - liga/desliga o pino de saída
- delay(tempo)
  - congela a execução por "tempo" milisegundos

Uso de led externo

- Pode-se também ligar um led externo no mesmo pino 13 através de um resistor de 120 ohms (por que 120?)
  - Tensão de saída no pino digital: 5 v
  - Tensão do diodo vermelho: 2.2 v
  - Corrente do diodo: 20 mA



Funções analógicas

- Gera e recebe valores de 0 a 1023.
- Permite medir além do LIGADO e DESLIGADO.
- AnalogRead() - Faz leitura do pino analógico. Valor entre 0 a 1023.
- AnalogWrite() - Gera valor analógico entre 0 e 1023. Onda PWM.

Porta serial

- É a forma de comunicar computador com Arduino em tempo real.
- Serial.begin(9600);
  - Ajusta velocidade em 9600 bps
- Serial.println("TEXTO");
  - Para escrever na tela.

Primeiro programa no uso da serial

```
void setup() // roda apenas uma vez, ao iniciar o pro
{
Serial.begin(9600); // configura a serial para 9600
Serial.println("Olá! Sou o Arduino!"); // envia a men
}
void loop() // fica rodando continuamente
{
// não faz nada
}
```

Versão 2

```
void setup() // roda apenas uma vez, ao iniciar o pro
{
Serial.begin(9600); // configura a serial para 9600
}
void loop() // fica rodando continuamente
{
Serial.println("Olá! Sou o Arduino!"); // envia a men
}
```

Versão 3

```
void setup() // roda apenas uma vez, ao iniciar o pro
{
Serial.begin(9600); // configura a serial para 9600
}
void loop() // fica rodando continuamente
{
Serial.println("Olá! Sou o Arduino!"); // envia a men
delay(1000); // fica parado 1000 milisegundos
}
```