Área 1

1a

Escreva um programa que imprima na tela o nome da disciplina “Algoritmos e Programação” e o seu nome completo (cada um em uma linha diferente), separados por uma linha em branco.

Envie o código-fonte (arquivo com extensão .c).

1b

Escreva um programa que imprima o resultado da multiplicação de três números inteiros.

Envie o código-fonte (arquivo com extensão .c).

2a

Faça um algoritmo que lê o peso (em kg) e a altura (em m) de uma pessoa e determina (e escreve) o imc (índice de massa corporal), que é dado pela fórmula: IMC = peso/(altura²).

2b

Faça um algoritmo que leia um valor inteiro positivo de 3 dígitos, armazene-o em uma variável inteira e determine a soma dos dígitos que formam o valor. Exemplo: o valor 453 tem soma dos dígitos igual a 12 (4 + 5 + 3).

2c

Faça um programa que calcula a quantidade de latas de tinta necessária e o custo para pintar uma quantidade de chapas de madeira retangulares. O algoritmo deve ler:

- a quantidade de chapas que devem ser pintadas
- as dimensões de cada chapa (largura e comprimento)
- o custo da lata de tinta
- a quantidade de metros quadrados que podem ser pintados com uma lata de tinta.

3a

Faça um programa que leia o número da conta bancária e o saldo de um cliente. Caso a conta tenha saldo negativo, o algoritmo deve enviar a seguinte mensagem: CONTA NEGATIVA, caso contrário NORMAL.

3b

Faça um programa que lê o peso de 3 pessoas e um valor que indica um peso de referência. O programa deve informar quantas pessoas possuem um peso superior ou igual ao peso de referência.

3c

Escreva um programa que leia duas notas referentes a atividades práticas de um aluno (AP1 e AP2), e a nota do trabalho final (TF). Calcule a média final (MF) conforme a ponderação usada na disciplina INF01202 neste semestre, ou seja, MF = 0,35 * AP1 + 0,45 * AP2 + 0,20 * TF, e determine o conceito final do aluno da seguinte forma:

- MF >= 8,5 : Conceito A
- 8,5 > MF >= 7,5 : Conceito B
- 7,5 > MF >= 6,0 : Conceito C
- MF < 6,0: Conceito D

Ao final, imprima na tela a nota final (com uma casa decimal) e o conceito.

4a

O cardápio de uma lancheria é o seguinte:

Especificação	Código	Preço
Dog	100	13,00
Dog especial	101	17,00
X Salada	104	18,00
X Lombinho	105	21,00
Bauru	110	24,00
Refrigerante lata	120	4,00

Escrever um programa que leia o **código** do item pedido, a **quantidade** e **calcule o valor a ser pago** por aquele lanche. O programa deve avisar caso o código seja inválido. Considere que a cada execução somente será calculado um item.

4b

Escreva um programa que leia do teclado dois valores inteiros. Seu programa deve, como resultado, exibir a **soma dos números pares** (considere que um número inteiro é par se o resto da divisão por 2 é zero) entre eles, incluindo os extremos do intervalo. A ordem dos números de entrada pode ser tanto crescente quanto decrescente. Ou seja, você não pode assumir que o primeiro número informado é menor que o segundo.

Exemplo de execução:

Entre com dois valores: 10 -7

Soma dos valores pares entre estes dois numeros: 18

// a conta feita foi 10 + 8 + 6 + 4 + 2 + 0 + -2 + -4 + -6 = 18

4c

Escreva um programa que **leia um valor inteiro positivo N**, que representa o número de valores que deverão ser lidos. A seguir, o programa deve **ler N valores reais**. Ao fim, o programa deve informar o **menor** e o **maior** valor dentre os informados. O programa deve funcionar para qualquer valor de N.

Exemplo de execução:

Informe a quantidade de valores considerados: 5

Informe o valor 1: 10.2

Informe o valor 2: 12.4

Informe o valor 3: 20.82

Informe o valor 4: 30.12

Informe o valor 5: 8.0

Menor valor: 8.0

Maior valor: 30.12

5a

Faça um programa que leia um número inteiro N maior ou igual a 1 (o programa deverá repetidamente pedir um novo número caso o usuário digite um número inválido). Calcule e imprima todos os números primos menores que N. Lembre que número primo é um inteiro só com 2 divisores: o número 1 e ele próprio.

5b

Você deve escrever um programa para auxiliar uma campanha para arrecadação de doações. O programa deve inicialmente **ler o valor que se deseja arrecadar** com a campanha. A seguir, o programa deve **ler valores** doados, até que o valor arrecadado seja **igual ou superior** ao valor almejado pela campanha. Ao fim do programa, ele deve informar o **total arrecadado** e a **média dos valores** arrecadados.

OBS: O programa deve realizar verificação de consistência dos valores informados (i.e. devem ser sempre positivos)

Exemplo de execução:

Informe o valor que se deseja arrecadar: 2500,00

Informe o valor da doação: 100,00

Informe o valor da doação: 300,00

Informe o valor da doação: 50,00

Informe o valor da doação: 1000,00

Informe o valor da doação: 1300,00

Valor total arrecadado: 2750,00

Média das doações: 550,00

5c

Leia um determinado **número inteiro *M***. A seguir **leia um número indeterminado** de valores inteiros até que o usuário digite o valor 0 (zero é sinal de parada de leitura de números). O programa deve exibir o **número lido da lista que mais se aproxima do número *M*** dado no início. Note que o número zero, usado para marcar o fim da leitura de valores, não deve ser considerado para definir o número mais próximo de *M*.

Exemplo de execução:

Entre com o numero M: 5

Entre com o numero da lista: -1

Entre com o numero da lista: 2

Entre com o numero da lista: -3

Entre com o numero da lista: -4

Entre com o numero da lista: 15

Entre com o numero da lista: 0

Numero mais proximo de 5: 2

6a

Faça um programa que lê e armazena 5 valores em um arranjo de inteiros **arr1**, depois lê e armazena 5 valores em outro arranjo de inteiros **arr2**.

Este programa deve criar um terceiro arranjo de inteiros **arr3** de 10 posições, que deve armazenar os valores dos outros 2 arranjos, de modo que os 5 primeiros elementos são de **arr2** e os demais são elementos de **arr1**. No fim, imprima todos os elementos de **arr3**.

Ex:

Digite elemento 0 de arr1: 3

Digite elemento 1 de arr1: 80

Digite elemento 2 de arr1: -2

Digite elemento 3 de arr1: 51

Digite elemento 4 de arr1: 12

Digite elemento 0 de arr2: 68

Digite elemento 1 de arr2: -45

Digite elemento 2 de arr2: 21

Digite elemento 3 de arr2: 2

Digite elemento 4 de arr2: 3

Valores de arr3: 68 -45 21 2 3 3 80 -2 51 12

6b

Faça um programa para simular a corrida de 5 cavalos.

No início do programa peça para o usuário **informar os nomes** de cada cavalo. O programa também deve **ler um valor inteiro D**, referente ao comprimento da pista; e **um valor inteiro DMP**, referente à distância máxima do passo de um cavalo.

A posição de cada cavalo na pista é dada por uma variável inteira que começa em 0, sobre a qual serão somadas as distâncias das passadas em cada instante. (Inclusive você pode usar um vetor de 5 inteiros para representar tais posições).

A cada passo de repetição, deve-se **sortear um valor aleatório entre 0 e DMP** para cada cavalo, que vai indicar o quanto cada cavalo andou naquele passo. Siga atualizando as distâncias percorridas por cada cavalo, até que algum cavalo tenha alcançado o fim da pista.

Quando isso acontecer, **indique o cavalo vencedor** e imprima os **nomes** e as **distâncias percorridas** por todos.

6c

Em um campeonato de futebol foram informados o **nome de 6 atletas** e o **número de gols** que cada um marcou. Faça um programa que, a partir da **leitura destas informações**, forneça o **nome do goleador** do campeonato, e o **número de gols feitos por ele**.

Observação: Se houver mais de um goleador com a mesma quantidade de gols, considere que o primeiro nome a aparecer será o indicado.

Ex:

Digite nome do atleta: Badico

Digite gols do atleta : 21

Digite nome do atleta: Gustavo Papa

Digite gols do atleta : 16

Digite nome do atleta: Sandro Sotilli

Digite gols do atleta : 22

Digite nome do atleta: Ernestina

Digite gols do atleta : 16

Digite nome do atleta: Claudio Milar

Digite gols do atleta : 17

Digite nome do atleta: João Paulo

Digite gols do atleta : 19

Goleador foi Sandro Sotilli com 22 gols

7a

Faça um programa que lê 2 vetores de 5 posições: v1,v2. O programa deve criar uma matriz 4X5, de tal modo que:

- os elementos da 1ª linha da matriz sejam os elementos do vetor v1
- os elementos da 2ª linha sejam os quadrados dos elementos do vetor v2
- os elementos da 3ª linha sejam a multiplicação dos elementos de v1 pelos elementos de v2
- os elementos da 4ª linha sejam:
    - 0 quando os elementos dos dois vetores forem pares

        - 1 quando os dois elementos forem ímpares
        - 2 quando um deles for par e o outro ímpar
- Ao final, imprima a matriz.

Exemplo de execução:

v1: 5, 3, -7, 2, 6

v2: 2, 5, 3, 4, 1

matriz:

5, 3, -7, 2, 6

4, 25, 9, 16, 1

10, 15, -21, 8, 6

2, 1, 1, 0, 2

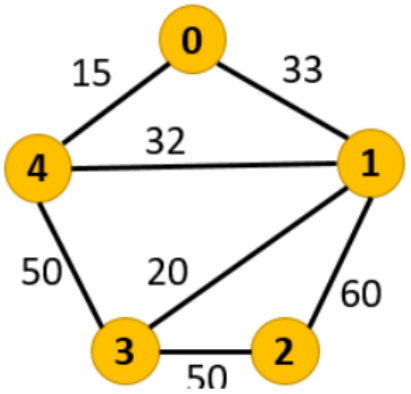
7b

Considere N cidades (utilize #define N 5). Faça um programa que utilize uma matriz para representar as distâncias entre essas cidades, de tal modo que a célula (i,j) da matriz representa a distância entre a cidade i e a cidade j. Nesta matriz, distâncias menores que zero indicam que não há via direta que conecta duas cidades.

- OBS: Para facilitar inicialize a matriz com os valores fixos mostrados no exemplo abaixo (você pode alterar os valores da matriz para testar, mas envie o exercicio com estes valores). Não é para pedir para o usuário informar os valores dela (a matriz deve ser inicializada no código).

Considere a seguinte matriz de distâncias:

0	33	-1	-1	15
33	0	60	20	32
-1	60	0	50	-1
-1	20	50	0	50
15	32	-1	50	0



A seguir, o programa deve pedir para o usuário uma sequência de N valores (cada valor entre 0 e N-1), que indica um trajeto que alguém gostaria de realizar para visitar um conjunto dessas cidades. O programa deve informar se o trajeto pretendido é possível ou não. Um trajeto não é possível se ele incluir uma sequência de cidades que não estão conectadas entre si.

Exemplo de execução:

Informe um trajeto:

Cidade 1: 0

Cidade 2: 1

Cidade 3: 2

Cidade 4: 3

Cidade 5: 4

O trajeto é possível

Exemplo de execução:

Informe um trajeto:

Cidade 1: 0

Cidade 2: 2

Cidade 3: 3

Cidade 4: 1

Cidade 5: 4

O trajeto não é possível

Área 2

9a

Faça uma função que recebe como parâmetro três números inteiros, hora, min e seg, representando um dado horário de um dia. A função deve computar e retornar a quantidade total de minutos desde 00:00:00 (em valor decimal). Caso o horário informado seja inválido, a função deverá retornar um valor negativo. O programa principal deverá ler os três números inteiros, chamar a função e, caso tenha sido verificado que o horário informado é válido, imprimir o total de horas computado. Caso contrário, deve imprimir “Horario invalido”.

Exemplo de execução:

Digite hora: 2

Digite min: 30

Digite seg: 25

Total de minutos decorridos: 150,42

Exemplo de execução:

Digite hora: 14

Digite min: 85

Digite seg: 10

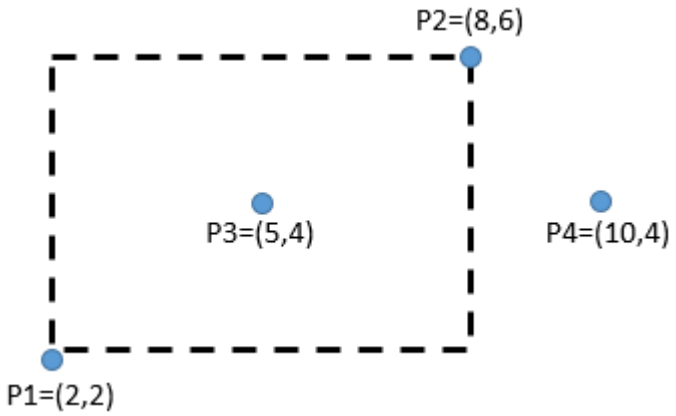
Horario invalido!

9b

Faça uma função chamada **menor2** que recebe como parâmetros dois números inteiros e retorna o menor deles. A seguir, faça uma função chamada **menor3** que recebe como parâmetro três números inteiros e retorna o menor deles. A função **menor3** deve necessariamente usar a função **menor2** em sua lógica interna. Escreva um programa principal que leia 3 números, computa qual deles é o menor usando a função **menor3** e imprima na tela qual dos números é o menor.

9c

Faça uma função tipada que recebe como entrada 6 números (3 pares ordenados) como parâmetros: x_1, y_1, x_2, y_2, x_3, y_3. Nesta sequência de parâmetros, os primeiros 2 pares ordenados representam coordenadas de 2 pontos (P_1 e P_2) no plano cartesiano que definem 2 cantos de um retângulo: P_1 representa o canto inferior esquerdo e P_2 representa o canto superior direito. O último par ordenado representa um outro ponto P_3 arbitrário. A função deve verificar se o ponto P_3 está dentro da área do retângulo definido pelos 2 primeiros pontos. Caso o ponto esteja dentro do retângulo, a função deve retornar 1, caso contrário, deve retornar 0 (zero). Escreva um programa principal que leia 6 números, chame a função desenvolvida e imprima na tela se o ponto P_3 está dentro do retângulo ou não.



Exemplo de execução:

Entre as coordenadas x e y do ponto 1: 2 2

Entre as coordenadas x e y do ponto 2: 8 6

Entre as coordenadas x e y do ponto 3: 5 4

O ponto 3 esta dentro do retângulo.

Abaixo, veja uma representação gráfica dos dados acima, incluindo também um quarto ponto (10,4), que não estaria dentro do retângulo.

10a

Faça uma função que recebe 3 vetores (v1, v2 e v3) de números inteiros de mesmo tamanho, e um valor inteiro que representa o tamanho deles. A função deve retornar em v3 a intersecção dos elementos em v1 e v2, ou seja, os elementos que estão simultaneamente em ambos os vetores. Para facilitar, considere que dentro de um mesmo vetor não há elementos repetidos. A função deve também retornar (através de return) o número de elementos da intersecção. O programa principal deve ler os vetores v1 e v2, chamar a função criada e depois imprimir v3 (somente os elementos relevantes dele, ou seja, sem imprimir “lixo” de memória).

Exemplo de execução:

V1: 12 20 3 18 90

V2: 90 65 12 36 20

A intersecção dos conjuntos possui 3 elementos.

V3: 12 20 90

10b

Faça uma função que receba uma matriz de inteiros positivos (m_in), uma matriz de floats (m_out) de mesmo tamanho de m_in, e a quantidade de linhas (L) e colunas (C) das matrizes. A função deve colocar em m_out os elementos normalizados de m_in, ou seja, cada elemento de m_in deve ser dividido pela soma de todos os elementos dela (portanto, a soma de todos os elementos de m_out deve ser igual a 1). O programa principal deve ler a matriz de inteiros positivos (se o usuário informar um número inválido, peça para ele informar novamente), chamar a função e depois imprimir a matriz de floats, com duas casas decimais.

OBS: a passagem de matrizes (arrays multidimensionais) por parâmetro em uma função permite que a primeira dimensão da matriz seja variável, mas que as demais dimensões sejam fixas.

ex: void funcao(int matriz[][COLS], ...) // onde COLS vem de um define

Exemplo de execução:

m_in:

5 10 2 4

6 9 8 6

m_out:

0.10 0.20 0.04 0.08

0.12 0.18 0.16 0.12

11a

Faça um programa que defina uma estrutura **jogador** contendo nome (string) e pontuação (int). O programa principal deverá criar um vetor de 5 jogadores e pedir para o usuário informar os nomes e pontuações dos mesmos. Depois deverá chamar uma função que **ordena o vetor** de jogadores por pontuação usando o algoritmo Bubblesort. Por fim, imprima na main os nomes e pontuações dos jogadores (**da maior** pontuação **para a menor**).

Exemplo de execução:

Digite nome do jogador 1: Jeff

Digite pontuacao do jogador 1: 60

Digite nome do jogador 2: Bob

Digite pontuacao do jogador 2: 120

Digite nome do jogador 3: George

Digite pontuacao do jogador 3: 105

Digite nome do jogador 4: Tom

Digite pontuacao do jogador 4: 90

Digite nome do jogador 5: Roy

Digite pontuacao do jogador 5: 100

Ordenacao:

Bob - 120

George - 105

Roy - 100

Tom - 90

Jeff - 60

11b

Faça um programa que defina uma estrutura **carta** contendo número (int) e naipe (int ou char). Crie na main um vetor de 52 cartas. Crie uma função **inicializa_baralho** (do tipo void) que recebe um vetor de 52 cartas e o inicializa com todos os valores possíveis de cartas: ou seja, cada carta do vetor deve ter um par diferente de valor (dentre 13 possíveis) e naipe (dentre 4 possíveis). Crie outra função **imprime_cartas** (do tipo void) que receba um vetor de cartas e a quantidade de elementos, e imprime todas as cartas. O programa principal deve chamar as duas funções.

Sugestão: use defines para definir número de valores, naipes, total de cartas...

11c

Aproveitando o que foi desenvolvido no exercício 11.b, faça um programa que defina um vetor **de ponteiros** da estrutura **carta** (definida anteriormente). O tamanho deste vetor também será 52 elementos. Todos os elementos desse vetor devem ser inicializados com valor NULL (isto é, originalmente não apontam para nenhum endereço de memória). Crie uma função **verifica_baralho** que recebe um **vetor de cartas** v_in, um **vetor de ponteiros de cartas** v_out e o tamanho dos vetores. A função deverá copiar para o vetor de saída o endereço das cartas do vetor de entrada que possuem valor par e naipe de copas. Na main imprima todas as cartas apontadas pelo vetor de ponteiros de cartas (não imprima elementos com valor NULL).

→ DICA de como copiar endereço de elemento de um vetor para um vetor de ponteiros:

v_out[indice_vout] = &(v_in[indice_vin])

→ DICA de como inicializar todos elementos de um vetor de ponteiros como NULL:

tipo* vetor_de_ponteiros[TAMANHO] = {NULL}

12a

Faça um programa que manipule um arquivo binário. O programa deverá pedir que o usuário indique o nome do arquivo binário (por exemplo, teste.bin). Na primeira execução (considerando um nome específico de arquivo), o programa criará o arquivo, nas demais execuções apenas o lerá.

- Se o arquivo não existir, o programa criará este arquivo. Em seguida, pedirá para o usuário digitar um **int**, um **float** e um **char**. E escreverá estes valores no arquivo (nesta respectiva ordem).

- Se o arquivo existir, o programa deverá abrí-lo. Depois, deverá-ler um **int**, um **float** e um **char** (nesta respectiva ordem), e imprimir os valores na tela.

12b

Faça um programa que defina e inicialize um vetor de elementos da estrutura **carta** (conforme desenvolvido nos exercícios da semana anterior). Crie uma função **salva_estado** que recebe um vetor de cartas e o tamanho do mesmo. Nesta função, peça para o usuário informar um nome de arquivo, depois abra-o para **escrita** (criando-o se ele não existir, ou sobrescrevendo-o se existir). Escreva no arquivo um **int** indicando a quantidade de elementos do vetor. Depois, escreva nele todos os elementos do vetor, ou seja, todas as cartas.

12c

Faça um programa que defina um vetor de elementos da estrutura **carta** (conforme desenvolvido nos exercícios da semana anterior). Crie uma função **carrega_estado** que recebe este vetor de cartas e o tamanho do mesmo e retorna um **int**. Nesta função, peça para o usuário informar um nome de arquivo, depois abra-o para **leitura**. Se o arquivo não existir, a função deve retornar **-1**, senão deverá ler os dados do arquivo e retornar a **quantidade de elementos lidos**. Leia do arquivo um **int** indicando a quantidade de elementos disponíveis. Depois, preencha o vetor com os elementos lidos do arquivo. Na main, imprima o vetor de cartas.

OBS: Se a quantidade de elementos no arquivo for maior do que o tamanho do vetor, leia somente a quantidade de elementos que caiba no vetor.

DICA: Para imprimir as cartas, use a função **imprime_cartas** desenvolvida nos exercícios da semana anterior.

13a

Faça um programa no qual o usuário informa o nome de um arquivo texto e uma palavra, e imprima todas as linhas do arquivo que contém a palavra informada (indicando na frente o número da linha).

Dicas: O comando **fgets** (para ler uma linha de texto) retorna NULL quando o arquivo de onde ele está lendo se encerra. O comando **strstr** pode ser usado para buscar uma string em outra string (ele retorna um ponteiro para a primeira ocorrência encontrada, ou NULL caso não encontre.

13b

Faça um programa que lê um arquivo de texto (cujo nome é informado pelo usuário) contendo números inteiros (um por linha). O programa deve ler esse arquivo e gerar dois novos arquivos textual: impares.txt e pares.txt. No arquivo impares, o programa deve armazenar os números ímpares e no arquivo pares, o programa deve armazenar os números pares. O programa deve armazenar um número por linha.

13c

Faça um programa que lê um arquivo texto informado pelo usuário, em que cada linha possui **dois números reais** separados por **ponto-e-vírgula**, que representam as dimensões de um retângulo. O programa deve imprimir, uma linha por vez, a **área de cada retângulo** representado pelas informações lidas. Considere o uso das funções strtok e atof.

Arquivo de entrada:

12.45;34.78

20.89;34.87

100.56;200.45

Arquivo de saída:

433.011

728.4343

20157.252

14a

Faça um programa contendo uma função recursiva que recebe como parâmetro um vetor de números inteiros, um valor N que representa o número de elementos que ele possui e que retorna a quantidade de números pares contidos no vetor. Na main peça para o usuário informar os valores de um vetor e execute a função criada para contar os pares contidos nesse vetor.

DICA: Faça a função verificar se um valor limítrofe do vetor (i.e. primeiro ou último elemento) satisfaz a condição buscada e, baseado no resultado, incremente (ou não) o resultado da função computada recursivamente com o restante dos elementos do vetor.

OBS: O vetor definido na main pode ter um tamanho fixo pré-definido (ex: 6), mas a função recursiva criada deve ser genérica para qualquer tamanho N.

Exemplo de execução:

Suponha o vetor: 4,5,3,5,2,7

O vetor possui 2 números pares.