Registros

59. Declare uma estrutura para um número complexo (parte real e parte imaginária do tipo `double`. Crie funções para ler, escrever e somar números complexos.

60. Considere uma estrutura para armazenar data (dia, mês e ano). Escreva uma função:

    (a) que recebe duas datas (verifique se as datas são válidas) e devolve o número de dias que decorreram entre as duas datas;

    (b) recebe uma data e retorna o dia do ano (1 a 366);

    (c) recebe duas datas `d1` e `d2` e compara-as retornando -1 se `d1` $<$ `d2`, 0 se `d1` $=$ `d2` e 1 se `d1` $>$ `d2`.

61. Defina um registro empregado para guardar os dados (nome, sobrenome, data de nascimento, RG, data de admissão, salário) de um empregado de sua empresa. Defina um vetor de empregados para armazenar todos os empregados de sua empresa.

62. Um racional é qualquer número da forma p/q, sendo p inteiro e q inteiro não-nulo. É conveniente representar um racional por um registro:

```
typedef struct {
   int p, q;
} racional;
```

    Vamos convencionar que o campo q de todo racional é estritamente positivo e que o máximo divisor comum dos campos p e q é 1. Escreva

    (a) uma função reduz que receba inteiros a e b e devolva o racional que representa a/b;

    (b) uma função neg que receba um racional x e devolva o racional -x;

    (c) uma função soma que receba racionais x e y e devolva o racional que representa a soma de x e y;

    (d) uma função mult que receba racionais x e y e devolva o racional que representa o produto de x por y;

    (e) uma função div que receba racionais x e y e devolva o racional que representa o quociente de x por y;

63. Implemente uma função para encontrar o ponto de sela de uma matriz. A rotina deverá retornar os índices da matriz cujo elemento é um ponto de sela.

64. Dada uma estrutura que armazena os dados de um ponto (coordenadas x e y), escreva um programa que leia a quantidade de pontos de um polígono, leia o vetor de pontos que representam o polígono (função leitura) e calcule a distância entre os pontos, dois a dois (distancia Euclidiana).

    Exemplo:

```
2    \\ número de pontos
2 6  \\ P1(x1, y1)
6 2  \\ P2(x2, y2)
Distancia entre os pontos 0 e 1: 5.66
```

65. Um número complexo é dividido em duas partes na forma $z = a + b.i$, onde a e b são números reais. Defina um tipo chamado Complexo usando a estrutura para contemplar estas duas partes. Além disso, faça funções para ler, somar, subtrair e multiplicar números complexos. Lembrando que, se $z_1 = a + bi$ e $z_2 = c + di$:

    a) $z_1 + z_2 = (a + c) + (b + d)i$;

    b) $z_1 - z_2 = (a - c) + (b - d)i$;

    c) $z_1.z_2 = (ac + bd) + (ad + bc)i$.

66. Escreva um programa para ler um número inteiro n que indicará a quantidade de alunos na turma. Após isso, ele deverá ler as notas das 3 provas de cada aluno e calcular a média de cada aluno. A média geral da turma em cada prova e o desvio padrão de cada prova também devem ser calculados e impressos. A função `main` é dada. Implemente as demais funções.

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

#define MAX 50

struct prova {
   float p[3];
   float media; };

typedef struct prova tp_prova;

struct aluno {
   int ra[6];
   tp_prova nota; };

typedef struct aluno tp_aluno;

// protótipos da funções
void leitura(tp_aluno alunos[], int n);
void calcula_media_aluno(tp_aluno alunos[], int n);
void calcula_media_desvio_turma(tp_aluno alunos[],
     int n, float medias_turma[], float desvio_turma[]);
void imprime_dados(tp_aluno alunos[], int n,
     float medias_turma[], float desvio_turma[]);

int main(void)
{
   tp_aluno alunos[MAX];
   int qtdAlunos;
   float medias_turma[3]; // média da turma em cada prova
   float desvio_turma[3]; // desvio da média da turma em cada prova

   printf("Informe a quantidade de alunos: ");
    scanf("%d", &qtdAlunos);
```

```
    leitura(alunos, qtdAlunos); // lê dados de cada aluno
    calcula_media_aluno(alunos, qtdAlunos); // cálculo da média de cada aluno
    calcula_media_desvio_turma(alunos, qtdAlunos, medias_turma, desvio_turma);
    imprime_dados(alunos, qtdAlunos, medias_turma, desvio_turma);

    return (0);
}
```

Exemplo de entrada:

```
3
030034
4.0 5.0 6.0
123456
0.0 5.0 10.0
987654
6.0 9.0 3.0
```

Exemplo de saída:

```
RA: 030034
Prova 1: 4.0
Prova 2: 5.0
Prova 3: 6.0
Média: 5.0
----------------
RA: 123456
Prova 1: 0.0
Prova 2: 5.0
Prova 3: 10.0
Média: 5.0
----------------
RA: 987654
Prova 1: 6.0
Prova 2: 9.0
Prova 3: 3.0
Média: 6.0
----------------
Média da turma P1: 3.3
Desvio padrão P1: 2.4
Média da turma P2: 6.3
Desvio padrão P2: 1.8
Média da turma P3: 6.3
Desvio padrão P3: 2.8
```

67. Um número racional é dividido em duas partes inteiras, o numerador e o denominador. Defina um tipo chamado `Racional` usando a estrutura para contemplar estas duas partes. Além disso, faça funções para ler, somar, subtrair, multiplicar e dividir números racionais.

68. Suponha um cadastro de alunos onde cada aluno contém os seguintes campos: Nome, Data de Nascimento (dia, mês, ano), RG, Sexo, Endereço (Rua, Cidade, Estado, CEP), RA (Registro de Aluno) e IRA (Índice de rendimento: número real no intervalo [0,1]). Faça um programa que realize o cadastro de alunos em um vetor com 100 posições. O programa deve manipular este cadastro com as seguintes opções:

    (a) Inserir um novo aluno no cadastro (Se o cadastro estiver cheio, avise que não há memória disponível).

    (b) Exibir o maior e menor IRA cadastrado.

    (c) Exibir o aluno mais novo e mais velho cadastrado.

    (d) Ler o valor de um RA e exibir os dados do aluno no cadastro com mesmo RA.

    (e) Exibir as fichas na ordem em que foram inseridas.

    (f) Sair do programa

69. Considere uma estrutura para armazenar uma data (dia, mês e ano). Escreva três funções, tais que:

    (a) uma recebe duas datas (verifica se as datas são válidas) e devolve o número de dias que decorreram entre as duas datas;

    (b) uma recebe uma data e retorna o dia do ano (1 a 366);

    (c) uma faz comparação entre duas datas $\texttt{data}_1$ e $\texttt{data}_2$ e retorna -1 se $\texttt{data}_1 < \texttt{data}_2$, 0 se $\texttt{data}_1 = \texttt{data}_2$ e 1 se $\texttt{data}_1 > \texttt{data}_2$.

70. Implemente um programa que analise as cartas de dois jogadores de pôquer e defina qual deles é o vencedor. O seu programa deverá distribuir as cartas e naipes aleatoriamente (utilize o comando `rand` da `stdlib.h`). Em resumo, as regras do pôquer são:

    (a) Ordem crescente de valor das cartas: 2, 3, 4, 5, 6, 7, 8, 9, 10, Valete, Damas, Reis e Ás;

    (b) Todos os naipes têm o mesmo valor;

    (c) Ranking de mãos (ordem crescente): carta mais alta, par, 2 pares, trinca, straight, flush, full house, quadra, straight flush e royal flush.

    Para obter detalhes sobre o jogo, consulte: `http://www.universidadedopoker.com/regras-de-poker/ranking-de-maos-de-poker/`.

    Obs: use registros e tipos enumerados.

    Note que o seu programa não terá nenhuma entrada. Ele deverá gerar as duas mãos automaticamente, exibir na tela e declarar o vencedor.

    Exemplo:

```
--------------------------------
Jogador1:
AP 3O 7E RC RE  ---> Par de Reis    \\ Ás de paus, 3 de ouros, 7 de espadas,
--------------------------------       Reis de copas e Reis de espadas
Jogador2:
6E 5P 6O 10C VE ---> Par de Seis
```

```
----------------------------
Vencedor:
Jogador1
```

Depois, flexibilize o seu programa para trabalhar com uma quantidade de jogadores definida pelo usuário.

71. Declare um tipo chamado `TipoReg`, definido como um **tipo de registro** contendo os seguintes campos: Nome, RG, Salario, Idade, Sexo, DataNascimento; onde Nome e RG são strings, Salario é float, Idade é inteiro, Sexo é char e DataNascimento é um registro contendo três inteiros, dia, mês e ano. Declare um **tipo de registro** chamado `TipoCadastro` que contém dois campos: um campo funcionário, contendo um vetor com 100 posições do tipo `TipoReg` e outro campo inteiro, `Quant`, que indica a quantidade de funcionários no cadastro.

*** Todos os exercícios seguintes fazem uso do tipo `TipoCadastro` ***

72. Faça uma rotina, `IniciaCadastro`, que inicia uma variável do tipo `TipoCadastro`. A rotina atribui a quantidade de funcionários como zero.

73. Faça um procedimento, `LeFuncionarios`, com uma variável do tipo `TipoCadastro` como parâmetro de entrada. A rotina deve ler os dados de vários funcionários e colocar no vetor do cadastro, atualizando a quantidade de elementos não nulos. A rotina deve retornar com o cadastro atualizado. Lembre que o cadastro não suporta mais funcionários que os definidos no vetor de funcionários.

74. Faça uma rotina, chamada `ListaFuncionarios`, que exibe os dados de todos os funcionários.

75. Faça duas rotinas para ordenar os funcionários no cadastro. Uma que ordena pelo nome, `OrdenaNome`, e outra que ordena pelo salário, `OrdenaSalario`.

    Para a função `OrdenaNome` você por utilizar o exemplo abaixo (adicionar biblioteca `<string.h>`):

```
void Sort_String(char vetorNomes[][MAX_SIZE], int qtdNome) {
   int i,j;
   char aux[MAX_SIZE];

   for(j=0; j<qtdNome; j++)
      for(i=0; i<qtdNome-1; i++){
         if(strcmp(vetorNomes[i], vetorNomes[i + 1]) > 0){
            strcpy(aux, vetorNomes[i]);
            strcpy(vetorNomes[i], vetorNomes[i + 1]);
            strcpy(vetorNomes[i + 1], aux);
         }
      }
}
```

    Para a função `OrdenaSalario` você por utilizar o exemplo abaixo:

```
void Sort_Number(int v[], int n)
{
   int i, j, chave;

   for(j=1; j<n; j++) {
      chave = v[j];
      i = j-1;
      while(i >= 0 && v[i] > chave) {
         v[i+1] = v[i];
         i--;
      }
      v[i+1] = chave;
   }
}
```

76. Faça uma rotina, `SalarioIntervalo`, que tem como parâmetros: um do tipo `TipoCadastro` e dois valores reais v_1 e v_2, sendo $v_1 \leq v_2$. A rotina lista os funcionários com salário entre v_1 e v_2. Depois de exibe a média dos salários dos funcionários listados.

77. Faça uma rotina que dado um cadastro (posição do vetor), exibe o nome do funcionário e o imposto que é retido na fonte. Um funcionário que recebe até R$1000,00 é isento do imposto. Para quem recebe mais de R$1000,00 e até R$2000,00 tem 10% do salário retido na fonte. Para quem recebe mais de R$2000,00 e até R$3500,00 tem 15% do salário retido na fonte. Para quem recebe mais de R$3500,00 tem 25% do salário retido na fonte.

78. Faça uma rotina, `BuscaNome`, que tem como entrada o cadastro e mais um parâmetro que é um nome de um funcionário. A rotina deve retornar um registro (tipo `TipoReg`) contendo todas as informações do funcionário que tem o mesmo nome. Caso a rotina não encontre um elemento no vetor contendo o mesmo nome que o dado como parâmetro, o registro deve retornar com nome igual a vazio.

79. Faça uma rotina, `AtualizaSalario`, que tem como parâmetro o cadastro de funcionários. A rotina deve ler do teclado o RG do funcionário a atualizar. Em seguida, a rotina lê o novo salário do funcionário. Por fim, a rotina atualiza no cadastro o salário do funcionário com o RG especificado.

80. Faça uma rotina, chamada `ListaChefe`, que tem como parâmetro o cadastro e devolve um registro contendo os dados do funcionário que tem o maior salário.

81. Faça uma rotina, `RemoveFuncionario`, que tem parâmetros o cadastro e o RG de um funcionário. A rotina deve remover do cadastro o funcionário que contém o RG especificado. Lembre-se que os elementos não nulos no vetor do cadastro devem estar contíguos. Além disso, caso um elemento seja removido, a variável que indica a quantidade de elementos deve ser decrementada de uma unidade. Caso não exista nenhum elemento no vetor com o RG fornecido, a rotina não modifica nem os dados do vetor nem sua quantidade.

82. Faça uma rotina, `ListaAniversarioMes`, que tem como entrada o cadastro e um número inteiro que corresponde a um mês do ano. A rotina deve exibir o nome dos funcionários que nasceram neste mês, assim como o dia do seu nascimento.

83. Faça uma rotina, `ListaAniversarioSexo`, que tem como entrada o cadastro, três inteiros: dia, mês e ano, que correspondem a uma data e um caractere (sexo) com valor 'F' ou 'M'. A rotina deve imprimir o nome dos funcionários que nasceram nesta data e com sexo igual ao definido pelo parâmetro.

84. Em grandes lojas de departamento é comum existirem sistemas computacionais para gerenciar os estoques e produtos facilitando a organização e evitando que produtos fiquem em baixa no estoque. Os grandes desafios para os desenvolvedores desses sistemas não estam em apenas armazenar, mas sim em encontrar maneiras de que as funções desse sistema sejam os mais eficientes possiveis.

    (a) Implemente uma estrutura que contenha os dados de um determinado produto (Código do produto, nome, categoria, quantidade, preço...) e uma função para a inserção de novos produtos, lembrando que não se pode cadastrar mais produtos que seu registro suporta (Segmentation Fault!). Exemplo: typedef struct dados int codigo; . . . char nome[50]; Produto; Produto item[100];

    (b) Implemente uma busca por nome, código e uma que exiba quando algum produto estiver acabando (assuma que Quantidade < 10) em versão recursiva e outra iterativa, admitindo que os registros não estão ordenados. Nesse caso, qual mais eficiente? Se os registros fossem ordenados pra cada busca implementada, qual as vantagens? E as desvantagens?

85. Implemente duas versões de uma estrutura que armazene os dados financeiros de clientes de um banco, onde contenha Conta(int) e senha, nome(array de caracteres), sexo (char), e saldo (double), insira e imprima na tela os dados de determinado cliente conforme a conta e senha entrados. A primeira versão APENAS com matrizes, a segunda pode ser usado os registros com structs. Qual teve a implementação mais simples? Qual ficou mais eficiente? justifique.

    Exemplo:

    Dados:

```
-----------------------------------------|
 Conta | Senha | Nome    | Sexo| Saldo    |
-----------------------------------------|
 0001  | 123   | Evellyn | F   | 1000,00  |
-----------------------------------------|
 0002  | 312   | Kaleb   | M   | 10,19    |
-----------------------------------------|
 0002  | 123   | Jéssica | F   | 5,30     |
-----------------------------------------|
```

```
-----------------------------------------------|
Entrada                 |     Saida            |
-----------------------------------------------|
Conta: 0001             |Nome : Evellyn        |
Senha: 123              |Sexo : F              |
                        |Saldo: 1000,00        |
-----------------------------------------------|
```

```
-------------------------------------------------|
Entrada                 |     Saida              |
```

```
-------------------------------------------------|
Conta: 0005              |Conta ou senha invalidos!|
Senha: 123               |                         |
-------------------------------------------------|
```

Dica: Para a primeira implementação podemos usar varias matrizes para tratar os diferentes tipos de dados, e para unir os dados podemos nos valer dos indices. imprimindo os dados das matrizes cujo o indice seja correspondente.

86. Crie uma estrutura para um número complexo (parte real e parte imaginária do tipo double. Crie funções para ler, escrever e somar números complexos.

87. Considere uma estrutura para armazenar uma data (dia, mês e ano). Escreva uma função que: (a) que recebe duas datas (verifique se as datas sao válidas) e devolve o número de dias que decorreram entre as duas datas; (b) recebe uma data e retorna o dia do ano (1 a 366); (c) recebe duas datas data1 e data2 e compara-as retornando -1 se data1 $<$ d2, 0 se data1 $=$ data2 e 1 se data1 $>$ data2.

88. Implemente uma estrutura que armazene cartas de um baralho, e que contenha informações do numero (considere ás como 1, Valete como 11...) e as cores. Crie funções para inserir e ordenar (considere a seguinte ordem como crescente Ás 2,3... 10,Valete,Damas, Reis e Ouros, Espadas, Copas e Paus) e imprimir a ordem em que se encontra o baralho.

89. Crie um registro para armazenar os dados de alunos (RA, Nome, Curso) e crie funções para inserir, remover imprimir os alunos de um curso especificado, ou seja, quando for selecionado a impressão, deverá ser retornado todos os alunos de um determinado curso

90. Considerando o exercício anteerior, crie uma função recursiva para ordenar, e para busca binária de alunos.

91. Implemente um aplicativo que armazene as informações de um catalogo de musicas usando registros, onde possa inserir nova musicas (Nome, Artista, Compositor, e Nota (0 5), remover musica, e que possa ser atualizado esses dados posteriormente, além de poder ordenar pela nota atribuida a ela.

92. Implemente um sistema de criptografia basico, onde o usuário tem a opção de codificar a mensagem ou decodificar. A frase codificada é uma sequencia numerica separada por (hífens) que cuja decodificação se baseia na tabela a seguir.

```
-------------------------------------------------------------------------------
|A |B | C|D |E |F |G |H |I |J |k |L |M |N |O |P |Q |R |S |T |U |V |W |X |Y |Z |   |
-------------------------------------------------------------------------------
|1 |2 |3 |4 |5 |6 |7 |8 |9 |10|11|12|13|14|15|16|17|18|19|20|21|22|23|24|25|26|27 |
-------------------------------------------------------------------------------
```

Exemplo:

Entrada: 13-5-27-7-21-19-20-1 Saida: ME GUSTA

a)Crie um registro para armazenar os valores dessa tabela, sendo as letras como chars, e os numeros como unsigned int.
b)Crie uma função que decodifique uma mensagem
c)Crie uma função que codifique uma mensagem

93. Declarar o registro cuja representação gráfica é dada a seguir:

```
 ------------------------------------
|Ficha                               |
|------------------------------------|
|NOME  caracter                      |
|SALÁRIO   numero                    |
|CPF  11 números                     |
|IDADE Numero                        |
|SEXO Caracter                       |
--------------------------------------
```

94. Utilizando o registro criado no exercício anterior, atribuir os valores apresentados abaixo aos campos correspondentes:

```
|------------------------------------|
|Ficha                               |
|------------------------------------|
|JOSÉ  DA  SILVA                     |
|850,00                              |
|531.987.001-41                      |
|32                                  |
|M                                   |
--------------------------------------
```

95. Monte um programa para cadastrar imóveis a serem alugados ou vendidos, contendo os seguintes dados: tipo (loja, apartamento, casa, kit), endereço, bairro, valor, situação (aluguel ou venda). Ao final, solicitar ao usuário a situação a ser pesquisada e mostrar todos os dados dos imóveis enquadrados na solicitação;

96. cadastrar o nome do município, seu estado e sua população. Mostrar todos os municípios cadastrados do estado do Acre e os dados do município que contém a maior população.

97. Crie um de registro para armazenar os dados de um pai, esse registro deve conter Nome, Idade, e um vetor de outro registro filhos, que contenha nome idade e sexo. Crie funções para inserir, remover e imprimir esses pais.

98. Para cada um dos seguintes itens, projete uma estrutura de registro apropriada para as informações dadas.
    a. As cartas em um maço de jogo de baralho.
    b. Tempo medido em horas, minutos e segundos.
    c. Os registros de uma lista telefônica.
    d. Descrição de um automóvel (Marca, Ano, Modelo, Tipo, Cor, Acessórios, Preço).
    e. Descrição de um livro na biblioteca (Autor, Título, Editora, etc.).
    f. Times de futebol no campeonato nacional (Time, grupo, número de partidas jogadas, etc.).
    g. Posição das pedras em um jogo de damas no tabuleiro.

99. Explique as vantagens sobre o uso de registros em relação a matrizes

100. Aponte os erros na declaração dessa estrutura

```
typedef struct data {
      unsigned short int dia;
      long int mes;
      double ano;
}Data;

int main (void) {
   struct data a, b;
   int d;
   scanf ("%d %d %d", &a.dia, &a.mes, &a.ano);
   scanf ("%d", &d);
   b = fim_evento (a, d);
   printf ("%d %i %d\n", b.Dia, b.mes, b.ano)
   return -1;
}
```

101. Baseado no exercício anterior, reescreva de forma a corrigir os problemas.

102. Crie um tipo registro chamado Endereco que contenha os campos Rua (Alfanumérico), Numero (Inteiro), Bairro (Alfanumérico). Em seguida declare uma variável do tipo Endereco

103. Adapte o exercício anterior para que o campo rua seja um registro que contenha o nome da rua e o CEP

104. Utilizando registro, construa um programa que permita a entrada de nome, endereço e telefone de 5 pessoas e os imprima em ordem crescente de nome.

Ponteiros

105. Considere o abaixo:

```
int main()
{
  int i, j, *p_1, *p_2, **p_p_1, **p_p_2;
  i = 4;
  j = 5;
  p_1 = &i;
  p_2 = &j;
  p_p_1 = &p_2;
  p_p_2 = &p_1;
  return 0;
}
```

Considere também a seguinte alocação de memória (variável - endereço alocado para a mesma):

- `i` - 1000
- `j` - 1004
- `p_1` - 1057
- `p_2` - 1042
- `p_p_1` - 1547
- `p_p_2` - 1147

Indique o valor de:

(a) `i`
(b) `j`
(c) `p_1`
(d) `p_2`
(e) `p_p_1`
(f) `p_p_2`
(g) `&i`
(h) `&j`
(i) `&p_1`
(j) `&p_2`
(k) `&p_p_1`
(l) `&p_p_2`
(m) `*p_1`
(n) `*p_2`
(o) `*p_p_1`
(p) `*p_p_2`
(q) `**p_p_1`
(r) `**p_p_2`
(s) `&*p_1`
(t) `*&p_2`

106. Qual a diferença dos programas abaixo?

```
int main() {
  int i, *p_1, *p_2, v[10];
  p_1 = v;
  p_2 = p_1;
  for (i = 0; i < 10; i++)
  { v[i] = (2*i)+1; p_2++; }
  for (i = 0; i < 10; i++)
  { p_2--; printf("  %d  ", *p_2);}
  return 0;
}
```

```
int main() {
  int i, *p_1, *p_2, v[10];
  p_1 = v;
  p_2 = v;
  for (i = 0; i < 10; i++)
  { v[i] = (2*i)+1; p_2++; }
  for (i = 0; i < 10; i++)
  {p_2--; printf("  %d  ", (int)(p_2 - p_1))
  return 0;
}
```

107. Escreva duas versões de um programa que lê uma mensagem e verfica se a mesma é um palíndromo. A primeira versão deve utilizar inteiros para controlar as posições do vetor. A segunda versão deve utilizar ponteiros.

Ignore os caracteres que não são letras de *a* a *z*. Exemplos.:

```
Enter a message: He lived as a devil, eh?
palindrome

Enter a message: Madam, I am Adam.
not palindrome
```

108. Qual o erro da função que deveria criar uma cópia de uma *string*?

```
char *duplicate(cont char *p){
  char *q;
  strcpy(q, p);
  return q;
}
```

109. Seja `f` a seguinte função:

```
int f(char *s, char *t){
  char *p1, *p2;
  for (p1 = s; *p1; p1++) {
    for (p2 = t; *p2; p2++) {
      if (*p1 == *p2) break;
    if (*p2 == '\0') break;
```

```
  }
  return p1-s;
}
```

(a) qual é o valor de `f("abcd", "babc")`?

(b) qual é o valor de `f("abcd", "bcd")`?

(c) em geral, qual o valor que `f` retorna ao passar duas *strings* `s` e `t`

110. Escreva um programa que lê uma data na forma dd/mm/aaaa e o mostra na forma mês dd, aaaa, onde mês é o nome dos meses:

Exemplo:
Entre com a data (dd/mm/aaaa): 10/09/2010 Setembro 10, 2010

Armazene o nome das datas em um vetor que contém ponteiros para *strings*.

111. Escreva uma função que cria uma cópia de uma *string*. Use alocação dinâmica.

112. Faça um programa para ordenar um conjunto de palavras dado pelo usuário. Considere que cada palavra tem no máximo 20 caracteres e o espaço para armazená-las deve ser alocado dinamicamente.

113. Crie uma função que receba um nome completo e retorne o nome e o sobrenome em variáveis distintas. Considere como sobrenome o restante da string após o primeiro espaço.

114. Seja o seguinte trecho de programa:

```
 int i = 3, j = 5;
int *p, *q;

p = &i;
q = &j;
```

Qual será o valor das seguintes expressões:

```
* p == &i
* *p - *q
* **&p
* 3* - *p/(*q)+7
```

115. Qual será a saída deste programa supondo que i ocupa o endereço 4094 na memória?

```
main () {
  int i = 5, *p;
  p = &i;
  printf ("%x %d %d %d %d \n", p, *p+2, **&p, 3**p, **&p+4);
}
```

116. Considere a seguinte declaração de variáveis.

```
int a[] = {5, 15, 34, 54, 14, 2, 52, 72};
int *p = &a[1], *q = &a[5];
```

(a) Qual é o valor de `*(p+3)`?

(b) Qual é o valor de `*(q-3)`?

(c) Qual é o valor de `q-p`?

(d) A condição `p < q` é verdadeira ou falsa?

(e) A condição `*p < *q` é verdadeira ou falsa?

117. Qual será o conteúdo do vetor `a` após a execução do seguinte bloco:

```
#define N 10

int a[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
int *p = &a[0], *q = &a[N-1], temp;

while (p < q) {
   temp = *p;
   *p++ = *q;
   *q-- = temp;
}
```

118. Efetue as seguintes ações:

(a) Definir um tipo de registro com três campos: um campo numérico real, um campo caractere e um campo dado por um vetor de inteiros.

(b) Declarar um registro do tipo acima.

(c) Declarar um ponteiro para esse tipo de estrutura e apontá-lo para o registro declarado no item anterior.

(d) Atribuir valores aos campos do registro utilizando o ponteiro.

119. Implemente uma função que calcule a área da superfície e o volume de uma esfera de raio r. A função deverá obedecer o protótipo:

`void esfera (float r, float *area, float *volume);`

A área da superfície e o volume são dados, respectivamente por $4\pi r^2$ e $\frac{4\pi r^3}{3}$.

120. Implemente uma função que receba como parâmetro um vetor de números inteiros (`vet`) de tamanho n e inverta a ordem dos elementos armazenados nesse vetor. A função deverá obedecer o protótipo:

`int inverte_vetor (int n, int *vet);`

121. Implemente uma função que receba como parâmetro um vetor de números inteiros (`vet`) de tamanho n e retorne quantos números pares, ímpares e primos estão armazenados no vetor. A função deverá obedecer o protótipo:

`int consulta_vet (int n, int *vet, int *pares, int *impares);`

122. Faça uma função que receba como parâmetros três ponteiros para vetores de inteiros, todos do mesmo tamanho, tamanho este que também é passado como parâmetro para a função. Essa função deverá fazer com que cada célula do terceiro vetor se torne igual à soma das células correspondentes dos outros dois.

123. Faça uma função que receba como parâmetro uma *string* (vetor de caracteres) e o respectivo número de caracteres válidos (tamanho da *string*) como parâmetros. Essa função deverá inverter a *string*.

124. Faça um programa que leia uma matriz de caracteres de dimensão 3 x 5, e imprima-a na ordem inversa, usando ponteiros.

125. Escreva um programa que lê uma frase e verifica se ela um palíndromo (sequência de caracteres ou símbolos que quando lidos de trás para frente obtém-se o conjunto original).

```
Frase.....: A droga da gorda
Palíndromo: Sim

Frase.....: Alg2 é legal
Palíndromo: Não

Frase.....: Anotaram a data da maratona
Palíndromo: Sim
```

Dica: ignore os caracteres não alfanuméricos como espaço e pontuações

126. Escreva um programa que contenha a seguinte estrutura:

```
typedef struct duracao{
   int horas;
   int minutos;
   int segundos;
} tpduracao;
```

O programa também deverá conter uma função com o seguinte protótipo:

```
int calc(tpduracao a, tpduracao b, tpduracao *total, tpduracao *dif);
```

A função deverá receber por valor a duração dos experimentos a e b, e deverá calcular a soma da duração dos dois experimentos e a diferença de tempo dos mesmos, através dos parâmetros `total` e `dif`, recebidos por referência. O programa deverá retornar 1, por meio do uso do comando `return`, se o experimento a é mais curto ou de igual duração a b e 0 caso contrário.

O programa deverá receber os valores das durações dos testes e informar ao usuário as duas medidas calculadas. O programa também deverá informar qual é o teste mais curto.

127. Historicamente, César é reconhecido o primeiro homem a codificar mensagens. Ele reorganizava o texto de suas mensagens de maneira que o texto parecia não ter sentido. Cada mensagem sempre possuía uma contagem de letras cujo total equivalia a um quadrado perfeito, dependendo de quanto César tivesse que escrever. Assim, uma mensagem com 16 caracteres usava um quadrado de quatro por quatro; se fossem 25 caracteres, seria cinco por cinco; 100 caracteres requeriam um quadrado de dez por dez, etc. Seus oficiais sabiam que deviam transcrever o texto preenchendo as casas do quadrado sempre que uma mensagem aleatória chegasse. Ao fazerem isso, podiam ler a mensagem na vertical e seu sentido se tornaria claro.

Escreva um programa que lê o tamanho de uma *string* e a *string* de um arquivo de texto "msg.in" (utilize o operador '`>`' do linux) e escreve a mensagem decifrada na tela. Exemplo:

```
36
AEEUMOLSHMSCGT*AGU2A***L:****T*****A
```

Esta mensagem pode ser transcrita em um quadrado perfeito 6x6. Exemplo:

```
A E E U M O
L S H M S C
G T * A G U
2 A * * * L
: * * * * T
* * * * * A
```

Lendo cada coluna da matriz (desconsiderando o caractere '*'), a saída será:

```
ALG2: ESTA EH UMA MSG OCULTA
```

128. Você foi encarregado de desenvolver um programa para uma grande companhia de petróleo. O programa deve fornecer alguns dados sobre as bacias da companhia após uma série de operações envolvendo o estoque das mesmas. Para fins de testes a companhia utilizará o programa em 5 bacias, identificadas por números de 0 a 4.

Um operador poderá entrar com as seguintes operações:

- **d**: descoberta (adiciona o petróleo descoberto ao estoque da bacia),
- **c**: consumo (utiliza o petróleo do estoque da bacia para fins internos),
- **v**: venda (vende o petróleo do estoque da bacia a um certo preço),
- **f**: fim das operações.

A companhia trabalha com dois tipos de petróleo: Pesado e Leve. Assuma que o preço da unidade de petróleo é R$100, tanto para o tipo Pesado quanto para o Leve. Assuma também que o estoque inicial de cada bacia é de 10 unidades de cada tipo de petróleo. Se ocorrer tentativa de venda ou consumo de mais petróleo do que o disponível, uma mensagem de erro deve ser exibida.

Após indicação do final das operações seu programa deverá retornar, para cada uma das bacias, o estoque total, a quantidade de petróleo Pesado, a quantidade de petróleo Leve e a renda (quantidade de dinheiro ganho com a venda do petróleo).

Além disso, a companhia também quer saber qual é a bacia com o maior estoque (se houver empate, as bacias com os maiores estoques) e qual a bacia com a maior renda (se houver empate, as bacias com as maiores rendas). Preços e quantidades são determinados por números inteiros.

É obrigatório o uso de vetores, registros e funções! Utilize também o comando `switch`.

Exemplo de E/S:

Entrada	Saída
`d 0 p 50` (descoberta 50 unid de petróleo pesado na bacia 0)	`0 70 60 10 0` (bacia total pesado leve renda)
`c 1 l 10`	`1 10 10 0 0`
`v 2 p 5`	`2 15 5 10 500`
`v 3 p 10`	`3 10 0 10 1000`
`d 4 l 30`	`4 50 10 40 0`
`f`	`0` (bacia com maior estoque)
	`3` (bacia com maior renda)

129. Criar um registro `Aluno` e um registro `Materia` tal como dado abaixo:

```
typedef struct Aluno{
   int matricula;
   float vNotas[5]; // Armazena as 5 notas de um aluno ao longo de um ano.
   char nome[100];
} Aluno;

typedef struct Materia{
   Aluno V[MAX]; // Armazena a informação de MAX alunos
   float media[5]; // Armazena as 5 médias do ano.
   int nAlunos; // Número de alunos matriculados no curso.
} Materia;
```

(a) Criar uma função `Aluno fillAluno(Aluno a1)` que preenche os campos de uma variável `a1` do tipo `Aluno` e retorna essa variável preenchida.

(b) Criar uma função `Materia fillMateria(Materia m1, int numAlunos)` que preenche os campos de uma variável `m1` do tipo `Materia` realizando chamadas a função `fillAluno` em um número de vezes igual ao número contido na variável `nAlunos`. Nota: número máximo de aluno por matéria (`MAX`) é igual a 60.

(c) Criar uma função `int mediaMateria(Materia m1)` que fornece a média de cada prova do ano para os alunos contidos na variável `m1` e assim preenche o campo `media` de `m1`.

(d) Criar uma função `void mostraMateria(Materia m1)` que mostra as informações dos alunos contidas na variável `m1`.

(e) Criar um programa que ilustra o funcionamento das funções anteriores. Para tanto, o programa deve possuir uma variável `pc1` do tipo `Materia` com 5 alunos. Depois, serão utilizadas as seguintes funções: `fillMateria` (que emprega `fillAluno`), `mediaMateria` e `mostraMateria`.

130. Criar um registro `Livro` e um registro `Biblioteca` tal como dado abaixo:

```
typedef struct Livro{
   int ano;
   char titulo[100];
   char autor[100];
   int nVolume; // Número de exemplares de um dado livro.
   float preco;
} Livro;

typedef struct Biblioteca{
   Livro V[MAX]; // Armazena a informação de MAX livros
   int nLivros // Número de livros existentes na biblioteca.
} Biblioteca;
```

(a) Criar uma função `Livro fillLivro(Livro l1)` que preenche os campos de uma variável `l1` do tipo `Livro` e retorna essa variável preenchida.

(b) Criar uma função `Biblioteca fillBiblioteca(Biblioteca b1, int numLivros)` que preenche os campos de uma variável `b1` do tipo `Biblioteca` realizando chamadas a função `fillLivro` em um número de vezes igual ao número contido na variável `nLivros`. Nota: número máximo de livros por Biblioteca (`MAX`) é igual a 1000.

(c) Criar uma função `int valorBiblioteca(Biblioteca b1)` que fornece o montante gasto para se comprar todos os exemplares existentes na biblioteca.

(d) Criar uma função `int maiorBiblioteca(Biblioteca b1)` que fornece o livro com maior número de exemplares existente na variável `b1`.

(e) Criar um programa que ilustra o funcionamento das funções anteriores. Para tanto, o programa deve possuir uma variável `bccs` do tipo Biblioteca com 5 livros. Depois, serão utilizadas as seguintes funções: `fillBiblioteca` (que emprega `fillLivro`), `valorBiblioteca` e `maiorBiblioteca`.

131. Faça um programa que leia um valor n, crie dinamicamente um vetor de n elementos e passe esse vetor para uma função que vai ler os seus elementos. Depois, através de outra função, o vetor preenchido deverá ser impresso. Além disso, antes de finalizar o programa, deve-se liberar a área de memória alocada.

132. Altere o programa do exercício anterior. Faça com que o vetor seja alocado dinamicamente dentro da função que lê os seus elementos. Nesse caso, a função `main` deverá conter apenas a leitura do valor de n, a chamada para a função de leitura e a chamada para a função de impressão. Além disso, antes de finalizar o programa, deve-se liberar a área de memória alocada.

133. Faça um programa que carregue um vetor de inteiros `v` com n elementos e que imprima o primeiro valor ímpar de v. Para isso, crie uma função `int *primeiro_impar(int *v, int n)` (passagem de ponteiro por valor).

134. Implemente o mesmo programa do exercício anterior. Entretanto, agora, o protótipo da função que deverá encontrar o primeiro número ímpar de `v` deverá ser igual a `void primeiro_impar(int **v, int n)` (passagem de ponteiro por referência).

135. Escreva um programa que solicita ao usuário um vetor de notas (números reais) e imprime a média aritmética das notas.

    Observações:

    a. Apesar de não ser necessário utilize um vetor.

    b. O programa não deve limitar o tamanho do vetor.

    c. Não deve ocorrer desperdício de memória.

    d. Após ser utilizada a memória deve ser devolvida.

136. Escreva um programa que solicita ao usuário o RA (inteiro) e a média final (real) de todos os seus alunos e imprime todos os alunos que estão de SAC ($5 \leq nota < 6$).

    Observações:

    a. Utilize um vetor de registros (estruturas) para armazenar os dados dos alunos.

    b. O programa não deve limitar o número de alunos.

    c. Não deve ocorrer desperdício de memória.

    d. Após ser utilizada a memória deve ser devolvida.

137. Utilizando alocação dinâmica, crie um programa que leia n idades (int), armazene-as e depois mostre-as em ordem crescente. O valor de n será informado pelo usuário. Crie funções e procedimentos para realizar as tarefas.

138. Escreva uma função que receba um ponteiro para uma *string* e troque todo o caractere após um espaço em branco pelo seu equivalente maiúsculo. Obs: como o tamanho da *string* é inicialmente desconhecido, aloque uma quantidade máxima de bytes, leia a *string* e depois realoque a variável para ocupar o seu tamanho exato.

139. Faça um programa que receba dois vetores: p de tamanho m e q de tamanho n. Primeiro, o usuário deverá informar o valor de m e os valores de cada elemento de p. Em seguida, o valor de n e os elementos de q. Finalmente, o programa deverá imprimir um novo vetor s no qual cada elemento corresponde a soma de $p[i] + q[i]$.

140. Faça programa que crie um registro `ficha` para armazenar os seguintes dados de uma pessoa: nome, endereço e telefone. Após ler as n fichas, ele deve imprimi-las em ordem lexicográfica.

    Observações:

    a. A quantidade de fichas deverá ser informada pelo usuário na chamada no programa em linha de comando;
    b. Crie uma função que receba um ponteiro e preencha os dados das `fichas`;
    c. Crie uma função que receba o ponteiro e imprime todas as `fichas`;
    d. A função principal deverá ter apenas chamadas para funções.

141. Implemente um programa que receba um vetor de inteiros com n posições e imprima:

    a. o maior elemento;
    b. o menor elemento;
    c. a média;

    Em seguida, expanda o tamanho do vetor para armazenar mais m elementos e imprima novamente as informações acima. Depois, ordene o vetor em ordem decrescente e redimensione-o para n posições iniciais. Imprima todas as informações novamente.

142. Crie um programa que leia um vetor de inteiros de tamanho n. Após cada valor lido, exiba o máximo, o mínimo e a média. Obs: use `calloc` para alocar o vetor e implemente uma função para cada operação.

    Exemplo:

```
3                           // tamanho de v
5                           // v[0]
max = 5 | min = 5 | med = 5  // saída 1
7                           // v[1]
max = 7 | min = 5 | med = 6  // saída 2
3                           // v[3]
max = 7 | min = 3 | med = 5  // saída 3
```

143. Escreva um programa que utiliza a estrutura abaixo para criar uma lista. Numa lista o campo `prox` recebe o endereço do próximo elemento da lista. Solicite ao usuário o número de elementos da lista e crie a mesma preenchendo o campo `val` com a posição do elemento na lista. Depois imprima a lista.

```
typedef struct Elemento {
   int val;
   struct Elemento *prox;
} Elemento;
```

144. Implemente um programa para alocar dinamicamente uma matriz de inteiros. Solicite o tamanho da matriz. Obs: Implemente duas funções:

   - `int *leMatriz(int l, int c)`
   - `void imprimeMatriz(int *m, int l, int c)`

   A alocação deverá ser feita na `leMatriz` e não deve ser usado nenhum ponteiro auxiliar para percorrer os índices.

145. Adapte o programa do exercício anterior para percorrer os índices da matriz usando um ponteiro auxiliar.

146. Implemente um programa para alocar dinamicamente uma matriz de inteiros como um vetor de ponteiros. Solicite o tamanho da matriz. Obs: Implemente duas funções:

   - `int **leMatriz(int nlin, int ncol)`
   - `void imprimeMatriz(int **m, int nlin, int ncol)`

   A alocação deverá ser feita na `leMatriz`.

147. Faça um programa que crie uma matriz, cujas dimensões deverão ser informadas pelo usuário, preencha esta matriz com valores inteiros digitados no teclado, porém o preenchimento deverá ser feito utilizando-se um ponteiro. Tal matriz deverá ser apresentada na tela, no formato linha x coluna, também utilizando ponteiro.

148. Fazer um programa que receba três nomes e as idades das respectivas pessoas (idades acima de 45 anos devem ser rejeitados) em um vetor de estruturas de dados, utilizando ponteiro. Após o recebimento, listar os 3 nomes e idades que nela foram armazenados, com ponteiros.

149. Crie uma estrutura, usando ponteiros, que registre as informações do Lápis que são: Dureza (inteiro), Fabricante (char) e Número (inteiro). Insira 3 registros nesta estrutura e os apresente na tela.

150. Qual o resultado apresentado pelo seguinte trecho de código. Justifique.

```
double d[ ] = { 5.4, 3.2, 1.0 };

double *pd = d[1];

++pd;

printf(?%f?, *pd);
```

151. Sejam as seguintes declarações:

```
int ai [] = {1, 2, 3, 4, 5};

int bi [5];
```

a) Justifique o resultado do seguinte segmento de código:

```
int *pi = &ai [2];

++ pi;

printf ("%d", *pi);
```

b) Escreva um segmento de código que coloque no array bi os elementos de ai invertendo a ordem, ie. Bi == 5, 4, 3, 2, 1.

152. Construa um programa que declare um vetor de strings com 10 elementos e o inicialize com nomes fornecidos pelo usuário através da entrada padrão e em seguida o retorne na saída padrão. A manipulação do vetor deve ser feita por meio de um ponteiro.

153. Dadas as declarações abaixo e o endereço de memória de cada variável, fornecer o valor de cada item.

```
int x = 10, *px = &x, **ppx = &px;
float y = 5.9, *py = &y, **ppy = &py;
```

x	FFA0
y	FFB4
px	FFF0
py	FFC6
ppx	FFA6
ppy	FFD4

Tabela 1: Endereços de memória

x		ppx		&py	
py		&x		(**ppy)++	
px		(*py)++		(**ppx)++	
&y		(*px)– –		&ppy	
y		**ppy		&ppx	

Tabela 2: Preencher os valores de cada ítem

154. Considerando as variáveis e ponteiros abaixo, identifique as atribuições permitidas.

```
int i, *pi, **ppi;
float f, *pf, **ppf;
```

i = f;		*pf = 10;		ppf = &pf;	
pi = &i;		f = i;		**ppi = 100;	
pf = &f		pi = &f;		**ppf = &pf;	
*ppi = π		*pi = 7.3		ppi = π	

Tabela 3: Coloque 'X' nas operações corretas

155. Seja o seguinte trecho de programa:

```
int i=3,j=5;
int *p, *q;
p = &i;
q = &j;
```

Qual é o valor das seguintes expressões?

(a) `p == &i;`

(b) `*p - *q;`

(c) `*p++;`

(d) `3 * - *p/((*q)-2);`

156. Qual será a saída do programa abaixo?

```
int main () {
   int i = 5, *p = &i;
   printf ("%p %d %d %d %d \n", p, *p+2, **&p, 3**p, **&p+4);
   return(0);
}
```

157. Se `i` e `j` são variáveis inteiras e `p` e `q` são ponteiros para inteiros, quais das seguintes expressões de atribuição são ilegais?

(a) `p = &i;`

(b) `*q = &j;`

(c) `p = &*&i;`

(d) `i = (*&)j;`

(e) `i = *&*&j;`

(f) `q = &p;`

(g) `i = (*p)++ + *q;`

158. Por que o código abaixo está errado?

```
void troca (int *i, int *j) {
   int *temp;
   *temp = *i; *i = *j; *j = *temp;
}
```

159. Declare os seguintes vetores:

```
int vi[5] = {2, 5, 1, 4, 0};
char c[5] = {'a', 'b', 'm', '4', '-'};
float vf[5] = {2.66, 0.125, 1.0, 4.99, 2.009};
```

Usando a função `printf`, com o argumento `%p`, e com ponteiros para cada um dos tipos de dados, descubra quantos bytes são alocados pelo seu compilador para cada vetor. Como se sabe que em um vetor as posições na memória são contínuas, se for impresso o endereço de duas posições, pela diferença entre os dois endereços pode-se descobrir quantos bytes são alocados.

160. Declare um vetor de inteiros com 300 posições, de forma que cada posição possua o valor igual ao índice da posição (logo, o vetor será ordenado de 0 a 299). Declare um ponteiro que aponte para a primeira posição deste vetor, ou seja, `p = &v[0]`. Usando a função `printf`, e o endereçamento do tipo `*(p ± n)`, imprima o valor de `*(p+n)` para `n` variando de 0 a 299.

    Obs: não confundir com o endereçamento `*p+n`, que imprime o conteúdo de `p` somado a `n`.

161. Um ponteiro pode ser usado para dizer a uma função onde ela deve depositar o resultado de seus cálculos. Escreva uma função `hm` que converta minutos em horas-e-minutos. A função recebe um inteiro `mnts` e os endereços de duas variáveis inteiras, digamos `h` e `m`, e atribui valores a essas variáveis de modo que `m` seja menor que 60 e que `60*h + m` seja igual a `mnts`.

162. Crie uma única função que retorne o valor máximo e o mínimo de um vetor, além de apontador para o valor mediano.

163. Implemente uma função que calcule a área da superfície e o volume de uma esfera de raio r. A função deverá obedecer o protótipo:

    ```
    void esfera (float r, float *area, float *volume);
    ```

    A área da superfície e o volume são dados, respectivamente por $4\pi r^2$ e $\frac{4}{3}\pi r^3$.

Arquivos

164. Faça um programa para ler um arquivo texto (o usuário deve fornecer o nome do arquivo junto com a chamada do programa) e imprimir o seu conteúdo na tela.

165. Faça um programa que copie o arquivo Texto T_1 para T_2 mantendo a estrutura de linhas.

166. Escreva um programa que conte o número de caracteres e de palavras de um arquivo texto.

167. Escreva um programa que lê um arquivo texto e copia apenas os caracteres alfabéticos (letras) para um arquivo de destino (ambos fornecidos na chamada do programa). Números e caracteres especiais devem ser desconsiderados.

168. Faça um programa que procura pelas ocorrências de uma *string* dentro de um arquivo texto e informa em quais posições (linha, coluna) foram encontradas tais ocorrências.