1- Faça uma função que recebe um vetor, o tamanho N de elementos desse um vetor e um valor de código. Se o código for 1, imprimir o vetor na ordem direta, se o código for 2, mostrar o vetor na ordem inversa.

2- Escreva um programa que leia e mostre um vetor de 20 elementos inteiros. A seguir, conte quantos valores pares existem no vetor.

3- Escreva um programa que leia um vetor de 50 posições de números inteiros e mostre somente os positivos.

4- Dado um vetor A de números inteiros, obter a maior diferença entre dois elementos consecutivos desse vetor. Imprimir a maior diferença e os índices dos respectivos elementos.

5- Dado um vetor A de números inteiros, faça uma função que calcula a média, e os valores máximo e mínimo presentes no vetor.

6. Faça um procedimento que recebe 2 vetores A e B de tamanho 10 de inteiros, por parâmetro. Ao final do procedimento, B deve conter o fatorial de cada elemento de A.

7. Dados dois vetores A e B de números inteiros, com k < 300, verifique se existem dois segmentos consecutivos iguais, isto é, se existem i e m, tais que para todo elemento A[x] e B[x], com i <= x <= m, A[x] == B[x]. Imprima a maior sequência.

8 – Crie uma função que receba como parâmetro um vetor estático de 20 posições para armazenar tipos floats. Crie ainda, duas outras funções que alimentem este vetor, e o exiba em tela. Como última tarefa, implemente uma função que receba esse vetor e retorne o maior elemento do mesmo. Obedeça os seguintes protótipos e NÃO use alocação dinâmica e nem ponteiros. Suas funções devem fazer parte de um programa completo.

```
void alimentar(float vetor[20], int n);
void exibir(float vetor[20], int n);
float maior(float vetor[20], int n);
```

9 – Modifique o programa acima, acrescentando uma nova função que retorne: o maior elemento, o menor elemento e a soma dos elementos do vetor. Note que para retornar 3 valores, você deve usar ponteiros para os elementos menor, maior e soma. Obedeça o seguinte protótipo.

```
void maior_menor_soma(float vetor[20], int n, float *maior, float *menor, float *
soma);
```

10 – Escreva uma função que receba um vetor de números reais e conte quantos números negativos estão no vetor. Obedeça o seguinte protótipo.

```
int negativos (float* vetor, int n);
```

11 – Escreva uma função que receba um vetor de números reais e inverta a ordem destes números no próprio vetor. Obedeça o protótipo:

```
int inverter (float* vetor, int n);
```

12 – Dado um vetor de inteiros, de tamanho n, escreva uma função que ordena os elementos do vetor na ordem crescente. Obedeça o protótipo:

```
int ordenar (int * vetor, int n);
```

Dica: implemente uma função que retorne o menor elemento de um vetor. Use essa função no seu algorítimo de ordenação.

13 – Implementação de um Tipo Abstrato de Dados pilha (TAD pilha). Uma pilha é um TAD que implementa a política de "último a entrar, primeiro a sair" (LIFO – Last in, first out). Sendo assim, um elemento "empilhado" só poderá deixar a pilha se não houver nenhum outro elemento "acima" dele. Em suma, apenas elementos do topo da pilha poderão ser retirados.
Implemente um programa completo em C que simule as principais operações de uma pilha descritas pelos seguintes protótipos:

```
void inicializar(int tamanho);
void empilhar(int elemento);
int desempilhar();
int tamanho();
int topo();
int cheia();
int vazia();
```

Para a sua implementação, use um vetor dinâmico como variável global, que é inicializado pelo método inicializar(), ou seja, usando o malloc. A função empilhar(), coloca um elemento no topo da pilha, caso a mesma não esteja cheia. A função desempilhar retira o elemento do topo da pilha, caso a mesma não esteja vazia. A função tamanho(), retorna quantos elementos existem na pilha em um determinado momento. A função cheia() retorna 1 ou 0 caso a pilha esteja cheia. De forma análoga é a função vazia().

14 - Adicione a sua implementação de pilha as seguintes funções, de acordo com os protótipos:

```
void inverter(int *pilha_invertida);
void clonar (int *pilha_clonada);
```

O método inverter deve retornar no ponteiro pilha_invertida, a sua pilha do exercício 6 invertida. Já a função clonar deve criar uma nova cópia da pilha no ponteiro pilha_clonada.