```
/************************************************************
 * São Luís, 13 de julho de 2022.                           *
 * Universidade Estadual do Maranhão.                       *
 * Centro de Ciências Tecnológicas - CCT.                   *
 * Curso: Engenharia de Computação.                         *
 * Disciplina: Linguagem de Programação.                    *
 * Professora: Yonara Costa Magalhães.                      *
 *                                                          *
 * INTEGRANTES                                              *
 * Álex Dias - 20210024649                                  *
 * Enzo Maldinni Montanha Rodrigues - 20210002991           *
 * Ismael Freitas de Oliveira - 20210003095                 *
 * Lucas Andrade dos Santos - 20210024738                   *
 * Pedro Inácio Carvalho de Almeida Soares - 20210003059    *
 * Atividade de Programação - 3ª Avaliação                  *
 *                                                          *
 * Questão:                                                 *
 * Implemente um programa em linguagem C que crie um array  *
 * alocado dinamicamente, cuja tamanho será informado pelo  *
 * usuário em tempo de execução.    Este array também deve  *
 * armazenar  valores  inteiros quaisquer  informados pelo  *
 * usuário em tempo de execução. Todas as  manipulações  e  *
 * acessos devem ser feitos de forma indireta.              *
 *                                                          *
 * Após o armazenamento dos valores, calcule e mostre:      *
 *     a. Identificar o maior e o menor número;             *
 *                                                          *
 *     b.Quantas vezes o maior  número se  repete           *
 *       e quantas vezes o menor número se repete;          *
 *                                                          *
 *     c.A  soma de  todos os números  que estão nas        *
 *       posições pares e a soma de todos os números        *
 *       que estão nas posições ímpares;                    *
 *                                                          *
 *     d.Verificar se a soma dos números das posições       *
 *       pares  é maior,  menor ou  igual a soma  dos       *
 *       números nas posições ímpares.                      *
 *                                                          *
 * O programa  deve  fazer  uso de pelo  menos uma  função  *
 * /procedimento para realizar uma  destas ações descritas  *
 * nas alíneas.  É necessário  implementar um  menu com as  *
 * seguintes opções:   criar     array,   liberar   array,  *
 * inicializar array,  sair  do  programa e uma opção para  *
 * cada um dos processamentos das alíneas, acima.           *
 *                                                          *
 *                                                          *
 *                                                          *
 ************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
// Todas as bibliotecas usadas no programa.

// Todas as variáveis globais que foram utilizadas.
int i;
int *p;
int n, opcao, maior_num, menor_num, inicializado = 0, criado = 0, op_inicial,
op_criar;
int *pn = &n, *p_opcao = &opcao, *p_maior_num = &maior_num, *p_menor_num =
&menor_num, *p_inicializado = &inicializado, *p_criado = &criado, *p_op_inicial
= &op_inicial, *p_op_criar = &op_criar;
int repete_maior = 0, repete_menor = 0, soma_par = 0, soma_impar = 0;
int *p_repete_maior = &repete_maior, *p_repete_menor = &repete_menor,
```

```
*p_soma_par = &soma_par, *p_soma_impar = &soma_impar;

// Imprime a introdução do trabalho.
void imprime_intro(){

printf("================================================================================
=========\n");
    printf("* d8888b.      d88888b      .d8888.      db    db      .88b  d88.
.d88b.  *\n");
    printf("* 88  `8D      88'          88'  YP      88    88      88'YbdP`88
.8P  Y8. *\n");
    printf("* 88oobY'      88ooooo      `8bo.        88    88      88  88  88
88    88 *\n");
    printf("* 88`8b        88~~~~~        `Y8b.      88    88      88  88  88
88    88 *\n");
    printf("* 88 `88.      88.          db   8D      88b  d88      88  88  88
`8b  d8' *\n");
    printf("* 88   YD      Y88888P      `8888Y'      ~Y8888P'      YP  YP  YP
`Y88P'  *\n");

printf("================================================================================
=========\n");

    //Introdução
    printf("*
*\n");
    printf("* CABEÇALHO
*\n");
    printf("*
*\n");
    printf("* São Luís, 9 de julho de 2022.
*\n");
    printf("* Universidade Estadual do Maranhão.
*\n");
    printf("* Centro de Ciências Tecnológicas - CCT.
*\n");
    printf("* Curso: Engenharia de Computação.
*\n");
    printf("* Disciplina: Linguagem de Programação.
*\n");
    printf("* Professora: Yonara Costa Magalhães.
*\n");
    printf("*
*\n");
    printf("* INTEGRANTES
*\n");
    printf("*
*\n");
    printf("* Álex Dias - 20210024649
*\n");
    printf("* Enzo Maldinni Montanha Rodrigues - 20210002991
*\n");
    printf("* Ismael Freitas de Oliveira - 20210003095
*\n");
    printf("* Lucas Andrade dos Santos - 20210024738
*\n");
    printf("* Pedro Inácio Carvalho de Almeida Soares - 20210003059
*\n");

printf("================================================================================
=========\n");

    system("pause");
}
```

```
// Imprime todo o menu do programa.
void imprime_menu(){

printf("================================================================================
==========\n");
    printf("*                            __  __   ______   _   _   _    _
*\n");
    printf("*                           |  \\/  | |  ____| | \\ | | | |  | |
*\n");
    printf("*                           | \\  / | | |__    |  \\| | | |  | |
*\n");
    printf("*                           | |\\/| | |  __|   | . ` | | |  | |
*\n");
    printf("*                           | |  | | | |____  | |\\  | | |__| |
*\n");
    printf("*                           |_|  |_| |______| |_| \\_|  \\____/
*\n");

printf("================================================================================
==========\n");

    printf("* 1 - Criar array.
*\n");
    printf("* 2 - Inicializar array.
*\n");
    printf("* 3 - Liberar array.
*\n");
    printf("* 4 - Exibe o maior e o menor número.
*\n");
    printf("* 5 - Exibe quantas vezes o maior e o menor número se repetem.
*\n");
    printf("* 6 - Exibe a soma de todos os números nas posições pares e ímpares.
*\n");
    printf("* 7 - Compara as somas das posições pares e ímpares.
*\n");
    printf("* 8 - Sair do programa.
*\n");
    printf("*
*\n");
    printf("* Obs: Caso digite um número decimal em qualquer interação com
*\n");
    printf("* o programa, a parte inteira do número será considerada!
*\n");

printf("================================================================================
==========\n");
    printf("Escolha a opção desejada: ");
}

// Testa se os inputs do usuario são numeros inteiros.
int validar_inteiro()
{
    setlocale(LC_ALL, "Portuguese");
    int opc;
    int chk;
    char tmp[20];
    while(scanf("%d", &opc) != 1)
    {
        if (!fgets(tmp, sizeof tmp, stdin));
        if (sscanf(tmp, "%d", &opc) != 1);
        int chk = sscanf(tmp, "%d", &opc);
        if (chk == 0)
        {
```

```
            printf("====================================\n");
            printf("* Dado inválido! Digite novamente:  *\n");
            printf("====================================\n");
        }
    }
    return(opc);
}

// Aloca dinamicamente memória para um array de tamanho N.
int criar_array()
{
    system("cls");
    setlocale(LC_ALL, "Portuguese");
    if(*p_criado == 1)
    {
        printf("================================\n");
        printf("* O vetor já foi criado antes! *\n");
        printf("================================\n");
        do
        {
            fflush(stdin);
            printf("============================================\n");
            printf("* Escolha uma opção:                       *\n* 1 - Inserir
o tamanho do vetor novamente *\n* 2 - Voltar ao menu                          *\
n");
            printf("============================================\n");
            *p_op_criar=validar_inteiro();
            switch(*p_op_criar)
            {
            case 1:
                break;
            case 2:
                return 0;
            default:
                printf("==================\n");
                printf("* Opção inválida! *\n");
                printf("==================\n");
            }

        }
        while((*p_op_criar!=1) && (*p_op_criar!=2));
    }
    fflush(stdin);
    printf("================================\n");
    printf("* Digite o tamanho do vetor:  *\n");
    printf("================================\n");
    *pn = validar_inteiro();

    if(*pn == 0)
    {
        printf("====================================\n");
        printf("* Vetor não pode ter tamanho zero! *\n");
        printf("====================================\n");
        fflush(stdin);
        system("pause");
        return 0;
    }else{
    p = (int *) malloc((*pn) * sizeof(int));
    if (p == NULL)
    {
        printf("========================\n");
        printf("* Memória indisponível! *\n");
        printf("========================\n");
    }else{
```

```
        printf("============================\n");
        printf("* Vetor criado com sucesso! *\n");
        printf("============================\n");
        *p_criado = 1;
    }
    *p_inicializado = 0;
    }
    system("pause");
}

// Inicializa um array de tamanho N e computa a soma das posições pares e
ímpares como qual o menor e maior valor do array.
int inicializar_array()
{
    if(*p_criado == 1){
        system("cls");
        setlocale(LC_ALL, "Portuguese");
        if(p != NULL)
        {
            if(*p_inicializado == 1)
            {
                printf("====================================\n");
                printf("* O vetor já foi inicializado antes! *\n");
                printf("====================================\n");
                do
                {
            fflush(stdin);
            printf("==================================\n");
            printf("* Escolha uma opção:              *\n* 1 - Inserir os
valores novamente *\n* 2 - Voltar ao menu              *\n");
            printf("==================================\n");
                    *p_op_inicial = validar_inteiro();
                    switch(*p_op_inicial)
                    {
                    case 1:
                        break;
                    case 2:
                        return 0;
                    default:
                        printf("==================\n");
                        printf("* Opção inválida! *\n");
                        printf("==================\n");
                    }
                }
                while((*p_op_inicial != 1) && (*p_op_inicial != 2));
            }

            for(i = 0; i < *pn; i++)
            {
                fflush(stdin);
                printf("========================================\n");
                printf("* Digite o valor da posição %d do vetor:  *\n", i);
                printf("========================================\n");
                p[i] = validar_inteiro();
            }
        }
        printf("======================================\n");
        printf("* O vetor foi inicializado com sucesso! *\n");
        printf("======================================\n");
        *p_inicializado = 1;
        system("pause");
    }else{
        system("cls");
        printf("==============================\n");
```

```
        printf("* O array ainda não foi criado *\n");
        printf("================================\n");
        system("pause");
        return 0;
    }
    return p[0];
}

// Imprime uma mensagem de erro se tentar manipular um que vetor não foi
inicializado.
int vetor_inicializado(){
        system("cls");
        printf("======================================\n");
        printf("* O vetor ainda não foi inicializado! *\n");
        printf("======================================\n");
        system("pause");

        return 0;
}

// Libera a memória que foi alocada para a criação do array.
int liberar_array()
{
    if(*p_criado == 1)
    {
        free(p);
        p = NULL;
        *p_criado = 0;
        *p_inicializado = 0;
        *p_soma_impar = *p_soma_par = 0;
        *p_repete_maior = *p_repete_menor = 0;

        system("cls");
        printf("==============================\n");
        printf("* Array liberado com sucesso *\n");
        printf("==============================\n");
        system("pause");
    }else{
        system("cls");
        printf("================================\n");
        printf("* O array ainda não foi criado *\n");
        printf("================================\n");
        system("pause");
    }
}

// Imprime o maior e o menor número do array.
void maior_menor()
{
    system("cls");
    if(*p_inicializado != 1)
    {
        vetor_inicializado();
    }else{

        *p_maior_num = p[0];
        *p_menor_num = p[0];

        for(i = 0; i < *pn; i++){
            if(p[i] > *p_maior_num)
            {
                *p_maior_num = p[i];
            }
            else if(p[i] < *p_menor_num)
```

```
            {
                *p_menor_num = p[i];
            }
        }
        if(*p_maior_num == *p_menor_num){

printf("================================================================\n");
            printf("* O maior e o menor número são o mesmo número que é igual a
%i *\n", *p_maior_num);

printf("================================================================\n");
            system("pause");
        }else{
            printf("====================================\n");
            printf("* O maior número é %i             *\n", *p_maior_num);
            printf("* Já, o menor número é igual a %i *\n", *p_menor_num);
            printf("====================================\n");
            system("pause");
        }
    }
}

// Conta e imprime quantas vezes o maior e o menor número do array se repetem.
void repeticao_maior_menor()
{
    system("cls");
    if(*p_inicializado != 1)
    {
        vetor_inicializado();
    }else{
        // computa a quantidade de vezes que o maior e menor n?meros se repetem
no array
        for(i = 0; i < *pn; i++)
        {
            if(p[i] == *p_maior_num)
            {
                (*p_repete_maior)++;
            }
            if(p[i] == *p_menor_num)
            {
                (*p_repete_menor)++;
            }

        }
        if(*p_maior_num != *p_menor_num){
        printf("============================================\n");
        printf("* O maior número se repete %i vezes          *\n",
*p_repete_maior);
        printf("* Enquanto o menor número se repete %i vezes *\n",
*p_repete_menor);
        printf("============================================\n");
        }else{
        printf("================================================\n");
        printf("* O maior e o menor número são o mesmo número *\n");
        printf("* E se repete %i vezes                          *\n",
*p_repete_maior);
        printf("================================================\n");
        }
        *p_repete_maior = *p_repete_menor = 0;
        system("pause");
    }
}

// Soma todos os números nas posições pares e ímpares e imprime o resultado de
```

```
ambas.
void soma_posicaopar_posicaoimpar()
{
    if(*p_inicializado != 1)
    {
        vetor_inicializado();
    }else{

        *p_soma_impar = 0;
        *p_soma_par = 0;

        for(i = 0; i < *pn; i++){
            if((i % 2) == 0)
            {
                *p_soma_par += p[i];
            }else{
                *p_soma_impar += p[i];
            }
        }

        if(*pn == 1){
            system("cls");
            printf("=========================================================\
n");
            printf("* A soma dos números nas posições pares é igual a %i   *\n",
*p_soma_par);
            printf("* E a soma dos números nas posições ímpares não existe *\
n");
            printf("=========================================================\
n");
            system("pause");
          }
      else if(*pn >= 2){
            system("cls");
            printf("=========================================================\n");
            printf("* A soma dos números nas posições pares é igual a %i *\n",
*p_soma_par);
            printf("* E a soma dos números nas posições ímpares é %i     *\n",
*p_soma_impar);
            printf("=========================================================\n");
            system("pause");
      }
    }
}

// Compara as somas das posições pares com as ímpares e retorna se é maior,
menor ou igual.
void comaparar_somas()
{
    if(*p_inicializado != 1)
    {
        vetor_inicializado();
    }else{
    system("cls");
    if(*p_soma_par == *p_soma_impar)
    {

printf("=========================================================================
==================\n");
        printf("* A soma dos números nas posições pares é igual a soma dos
números nas posições ímpares. *\n");

printf("=========================================================================
==================\n");
```

```
    }
    else if(*p_soma_par > *p_soma_impar)
    {

printf("================================================================================
======================\n");
        printf("* A soma dos números nas posições pares é maior que a soma dos
números nas posições ímpares. *\n");

printf("================================================================================
======================\n");
    }else{

printf("================================================================================
======================\n");
        printf("* A soma dos números nas posições pares é menor que a soma dos
números nas posições ímpares. *\n");

printf("================================================================================
======================\n");
    }
    system("pause");
    }
}

// Função main que executa todo o código.
int main ()
{
    setlocale(LC_ALL, "Portuguese");

    imprime_intro();

    do
    {
        system("cls");
        fflush(stdin);
        imprime_menu();
        *p_opcao = validar_inteiro();

        // Switch case para o usuário interagir com o programa.
        switch(*p_opcao)
        {
        case 1:
            criar_array();
            break;
        case 2:
            inicializar_array();
            break;
        case 3:
            liberar_array();
            break;
        case 4:
            maior_menor();
            break;
        case 5:
            repeticao_maior_menor();
            break;
        case 6:
            soma_posicaopar_posicaoimpar();
            break;
        case 7:
            comaparar_somas();
            break;
        case 8:
```

```
            *p_opcao=8;
            break;
        default:
            system("cls");
            printf("====================\n");
            printf("* Opção inválida! *\n");
            printf("====================\n");
            system("pause");
            break;
        }
    }
    while(*p_opcao != 8);
    printf("====================================\n");
    printf("* Obrigado por usar nosso programa! *\n");
    printf("====================================\n");
    // Fim do programa.
}
```