

```
//Inclusão de Bibliotecas
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <string.h>

#define tamanhoVetor 10

void trataMemoriaNaoSuficiente(int *ponteiro)
{
    if (ponteiro == NULL){

        printf("\n\nNao existe memoria suficiente.\nEncerrando programa.\n\n");

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

        exit(EXIT_FAILURE);
    }
}

int* valores_entre(int *v, int n, int min, int max, int *qtd)
{
    *qtd = 0;
    int *vr;

    for (int i = 0; i < n; i++)
    {
        if (v[i] >= min && v[i] <= max)
        {
            (*qtd)++;
        }
    }

    if (*qtd > 0)
    {
        vr = (int *)malloc((*qtd) * sizeof(int));
        trataMemoriaNaoSuficiente(vr);

        int j = 0;

        for (int i = 0; i < n; i++)
        {
            if (v[i] >= min && v[i] <= max)
            {
                vr[j] = v[i];
                j++;
            }
        }

        return vr;
    }
    else
    {
        return NULL;
    }
}

//Método Main - Entry Point do Programa
int main()
{
    //Declaração de variáveis locais

    //Título do Programa e autor
    printf(" << UEFS - PGCA 2014.1 - Programa funcao valores_entre>>");
    printf("\n\n ## Autor: Leonardo Melo\n\n");

    int vetorPrincipal[tamanhoVetor] = { 1, 2, 3, 4, 5, 0, 7, 8, 9, 10 };
```

```
printf("\n\nVetor principal:\n\n");

for (int i = 0; i < tamanhoVetor; i++)
{
    if (i == 0)
    {
        printf("|");
    }

    printf(" %d |", vetorPrincipal[i]);
}

int min;
int max;

do{
    printf("\n\nDigite o valor minimo: ");
    scanf("%d", &min);
    fflush(stdin);

    printf("\nDigite o valor maximo: ");
    scanf("%d", &max);
    fflush(stdin);

    if (min > max)
    {
        printf("\n\nO valor minimo nao pode ser maior que o valor maximo.\n\n");
    }
    else
    {
        break;
    }
} while (true);

int tamanhoVetorResultante;
int *vetorResultante = valores_entre(vetorPrincipal, tamanhoVetor, min, max, &tamanhoVetorResultante);

printf("\n\nVetor resultante:\n\n");

for (int i = 0; i < tamanhoVetorResultante; i++)
{
    if (i == 0)
    {
        printf("|");
    }

    printf(" %d |", vetorResultante[i]);
}

free(vetorResultante);

//Pula duas linhas e Pausa a Tela (Utilizando comandos DOS)
printf("\n\n\n");
system("pause");

//Retorno do método main
return(EXIT_SUCCESS);
}
```

```
C:\Users\leo\Desktop\Introducao_Programacao_C\Respostas\ProjetoPGCA028Exercicios\Debug\Pr...
<< UEFS - PGCA 2014.1 - Programa funcao valores_entre>>
## Autor: Leonardo Melo

Vetor principal:
1 2 3 4 5 0 7 8 9 10

Digite o valor minimo: 4
Digite o valor maximo: 8

Vetor resultante:
4 5 7 8

Press any key to continue . . .
```

```
//Inclusão de Bibliotecas
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

#define QTD_MIN_NUM_APOSTA 1
#define QTD_MAX_NUM_APOSTA 20

#define NUM_APOSTA_MIN 0
#define NUM_APOSTA_MAX 100

#define QTD_NUM_SORTEADOS 20

void trataMemoriaInsuficiente(int *ponteiro)
{
    if (ponteiro == NULL){

        printf("\n\nNao existe memoria suficiente.\nEncerrando programa.\n\n");

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

        exit(EXIT_FAILURE);
    }
}

void ordenaVetor(int n, int* v)
{
    int aux;

    for (int i = n - 1; i > 0; i--)
    {
        for (int j = 0; j < i; j++)
        {
            //se antecessor > sucessor entao inverta
            if (v[j] > v[j + 1])
            {
                //inversao de
                aux = v[j];
                v[j] = v[j + 1];
                v[j + 1] = aux;
            }
        }
    }
}

int inteiroAleatorio(int a, int b)
{
    double r, x, R = RAND_MAX;
    int i;

    r = rand();
    x = r / (R + 1);
    i = x * (b - a + 1);

    return a + i;
}

void mostraVetor(int *v, int tam)
{
    for (int i = 0; i < tam; i++)
    {
        if (i == 0)
        {
            printf("|");
        }
    }
}
```

```
        printf(" %3d |", v[i]);

        if ((i + 1) % 10 == 0)
        {
            printf("\n|");
        }
    }
}

bool validaQtdNumApostados(int n)
{
    if (n >= QTD_MIN_NUM_APOSTA && n <= QTD_MAX_NUM_APOSTA)
    {
        return true;
    }
    else
    {
        return false;
    }
}

bool validaNumApostado(int n)
{
    if (n >= NUM_APOSTA_MIN && n <= NUM_APOSTA_MAX)
    {
        return true;
    }
    else
    {
        return false;
    }
}

bool existeElementoVetor(int n, int tam, int *v)
{
    for (int i = 0; i < tam; i++)
    {
        if (v[i] == n)
        {
            return true;
        }
    }

    return false;
}

void ler_aposta(int *aposta, int n)
{
    for (int i = 0; i < n; i++)
    {
        do{
            printf("\n\nDigite o %d%c numero apostado (entre %d e %d): ", i + 1, 167, NUM_APOSTA_MIN,
NUM_APOSTA_MAX);
            scanf("%d", &aposta[i]);
            fflush(stdin);

            if (validaNumApostado(aposta[i]))
            {
                if (!existeElementoVetor(aposta[i], i, aposta))
                {
                    break;
                }
                else
                {
                    printf("\n\nEscolha um numero diferente.\nNumeros ja apostados:\n");
                    mostraVetor(aposta, i);
                }
            }
        }
        else
    }
}
```

```
        {
            printf("\n\nNumero invalido.\nDigite um numero entre %d e %d.\n\n", NUM_APOSTA_MIN,
NUM_APOSTA_MAX);
        }
    } while (true);
}

void sorteia_valores(int *sorteio, int n)
{
    srand(time(NULL));

    for (int i = 0; i < n; i++)
    {
        do{
            sorteio[i] = inteiroAleatorio(NUM_APOSTA_MIN, NUM_APOSTA_MAX);

            if (!existeElementoVetor(sorteio[i], i, sorteio))
            {
                break;
            }
        } while (true);
    }
}

int* compara_aposta(int *aposta, int *sorteio, int *qtdacertos, int na, int ns)
{
    *qtdacertos = 0;
    int *acertos = NULL;

    for (int i = 0; i < na; i++)
    {
        if (existeElementoVetor(aposta[i], ns, sorteio))
        {
            (*qtdacertos)++;
        }
    }

    if (*qtdacertos > 0)
    {
        acertos = (int *)malloc((*qtdacertos) * sizeof(int));
        trataMemoriaInsuficiente(acertos);

        if (acertos != NULL){

            int j = 0;

            for (int i = 0; i < na; i++)
            {
                if (existeElementoVetor(aposta[i], ns, sorteio))
                {
                    acertos[j] = aposta[i];
                    j++;
                }
            }
        }

        return acertos;
    }
}

//Método Main - Entry Point do Programa
int main()
{
    //Declaração de variáveis locais

    //Título do Programa e autor
```

```
printf(" << UEFS - PGCA 2014.1 - Programa Bingo>>");
printf("\n\n ## Autor: Leonardo Melo\n\n");

int qtdNumApostados;

do{

    printf("\nInforme a quantidade de numeros a serem apostados (entre %d e %d): ", QTD_MIN_NUM_APOSTA, QTD_MAX_NUM_APOSTA);
    scanf("%d", &qtdNumApostados);
    fflush(stdin);

    if (validaQtdNumApostados(qtdNumApostados))
    {
        break;
    }
    else
    {
        printf("\n\nQuantidade invalida.\nDigite um numero entre %d e %d.\n\n", QTD_MIN_NUM_APOSTA, QTD_MAX_NUM_APOSTA);
    }

} while (true);

int *numerosApostados = (int *)malloc(qtdNumApostados * sizeof(int));
trataMemoriaInsuficiente(numerosApostados);

ler_aposta(numerosApostados, qtdNumApostados);

int *numerosSorteados = (int *)malloc(QTD_NUM_SORTEADOS * sizeof(int));
trataMemoriaInsuficiente(numerosSorteados);

printf("\n\nSorteando...\n\n");
sorteia_valores(numerosSorteados, QTD_NUM_SORTEADOS);

int quantidadeAcertos;

int *numerosAcertados = compara_aposta(numerosApostados, numerosSorteados, &quantidadeAcertos, qtdNumApostados, QTD_NUM_SORTEADOS);

printf("\n\nRESULTADO\n");
printf("\nNumeros apostados:\n");

ordenaVetor(qtdNumApostados, numerosApostados);
mostraVetor(numerosApostados, qtdNumApostados);

printf("\n\nNumeros sorteados:\n");

ordenaVetor(QTD_NUM_SORTEADOS, numerosSorteados);
mostraVetor(numerosSorteados, QTD_NUM_SORTEADOS);

printf("\n\nQuantidade de acertos: %d\n", quantidadeAcertos);

if (quantidadeAcertos > 0){
    printf("\nNumeros acertados:\n");

    ordenaVetor(quantidadeAcertos, numerosAcertados);
    mostraVetor(numerosAcertados, quantidadeAcertos);
}

free(numerosApostados);
free(numerosSorteados);
free(numerosAcertados);

//Pula duas linhas e Pausa a Tela (Utilizando comandos DOS)
printf("\n\n\n");
system("pause");
```

```
    //Retorno do método main  
    return(EXIT_SUCCESS);  
}
```



```
C:\Users\leo\Desktop\Introducao_Programacao_C\Respostas\ProjetoPGCA028Exercicios\Debug\Pr...
<< UEFS - PGCA 2014.1 - Programa Bingo>>
## Autor: Leonardo Melo

Informe a quantidade de numeros a serem apostados <entre 1 e 20>: 5

Digite o 1º numero apostado <entre 0 e 100>: 1

Digite o 2º numero apostado <entre 0 e 100>: 2

Digite o 3º numero apostado <entre 0 e 100>: 3

Digite o 4º numero apostado <entre 0 e 100>: 4

Digite o 5º numero apostado <entre 0 e 100>: 5

Sorteando...

RESULTADO
Numeros apostados:
| 1 | 2 | 3 | 4 | 5 |
Numeros sorteados:
| 2 | 5 | 8 | 12 | 25 | 27 | 29 | 32 | 36 | 39 |
| 57 | 59 | 60 | 67 | 68 | 77 | 84 | 94 | 96 | 99 |
|
Quantidade de acertos: 2
Numeros acertados:
| 2 | 5 |
Press any key to continue . . .
```