



CENTRO UNIVERSITÁRIO INTERNACIONAL UNINTER
ESCOLA SUPERIOR POLITÉCNICA
TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS
DISCIPLINA DE LINGUAGEM DE PROGRAMAÇÃO

ATIVIDADE PRÁTICA

GUSTAVO KUZE DA SILVA – RU: 2091066
PROF. VINICIUS POZZOBON BORIN

PORTÃO – RIO GRANDE DO SUL
2018

Exercício 1:

Escreva um algoritmo em linguagem C com as seguintes instruções:

1. Declare três variáveis (inteiro, real e char);
2. Declare três ponteiros;
3. Associe as variáveis aos ponteiros;
4. Modifique os valores de cada variável indiretamente usando os ponteiros associados. Para armazenar os valores nas variáveis, armazene na variável char a primeira letra do seu nome, na variável inteira os dois últimos dígitos do seu RU e na variável real os 4 últimos dígitos do seu RU, sendo os 2 últimos os valores com virgula;
5. Imprima na tela os valores das variáveis antes e após a modificação.

Resolução:

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

int main() {

    /*Declaração da variável que irá conter a primeira letra do nome,
    do tipo char, inicializada com o sinal de subtração (-) */
    char primeira_letra_do_nome = '-',

        /*declaração do ponteiro que irá armazenar o endereço de memória da variável
    primeira_letra_do_nome*/
        *p_primeira_letra_do_nome;

    /*Declaração da variável que irá conter os dois ultimos digitos do RU, do tipo int, inicializada com
    o valor zero*/
    int ultimos_2_digitos_ru = 0,

        /*declaração do ponteiro que irá armazenar o endereço de memória da variável
    ultimos_2_digitos_ru*/
        *p_ultimos_2_digitos_ru;

    //Declaração da variável que irá conter os quatro ultimos digitos do RU, do tipo float, inicializada
    com o valor zero
    float ultimos_4_digitos_ru = 0,

        /*declaração do ponteiro que irá armazenar o endereço de memória da variável
    ultimos_4_digitos_ru*/
        *p_ultimos_4_digitos_ru;

    /*Aqui é feita a atribuição do endereço de memória das variáveis a seus respectivos ponteiros*/
    p_primeira_letra_do_nome = &primeira_letra_do_nome;
    p_ultimos_2_digitos_ru = &ultimos_2_digitos_ru;
    p_ultimos_4_digitos_ru = &ultimos_4_digitos_ru;

    /*Aqui é mostrado o valor inicial das variáveis na tela*/
    printf("\nValor inicial da variavel \"primeira_letra_do_nome\": %c.\n", primeira_letra_do_nome);
    printf("\nValor inicial da variavel \"ultimos_2_digitos_ru\": %d.\n", ultimos_2_digitos_ru);
    printf("\nValor inicial da variavel \"ultimos_4_digitos_ru\": %.2f.\n", ultimos_4_digitos_ru);
```

```

//atribuição dos valores às variáveis através dos ponteiros
*p_primeira_letra_do_nome = 'G';
*p_ultimos_2_digitos_ru = 66;
*p_ultimos_4_digitos_ru = 10.66;

/* Aqui são mostrados os valores das variáveis após a atribuição por referência*/
printf("\n\nValor da variavel \"primeira_letra_do_nome\" apos a atribuicao por referencia: %c.\n",
primeira_letra_do_nome);
printf("\nValor da variavel \"ultimos_2_digitos_ru\" apos a atribuicao por referencia: %d.\n",
ultimos_2_digitos_ru);
printf("\nValor da variavel \"ultimos_4_digitos_ru\" apos a atribuicao por referencia: %.2f.\n",
ultimos_4_digitos_ru);

system("pause");
return 0;
}

```

```

G:\Users\GS2N\Github Repos\Trabalho-C\Trabalho-C\AP\Resoluções\Trabalho-C\Debug\Exr-1.exe
Valor inicial da variavel "primeira_letra_do_nome": -.
Valor inicial da variavel "ultimos_2_digitos_ru": 0.
Valor inicial da variavel "ultimos_4_digitos_ru": 0.00.

Valor da variavel "primeira_letra_do_nome" apos a atribuicao por referencia: G.
Valor da variavel "ultimos_2_digitos_ru" apos a atribuicao por referencia: 66.
Valor da variavel "ultimos_4_digitos_ru" apos a atribuicao por referencia: 10.66.
Pressione qualquer tecla para continuar. . .

```

Exercício 2:

Escreva um algoritmo em LINGUAGEM C que armazene na memória o seu RU e o valor 1234567, ambos digitados pelo usuário na tela.

Em seguida, imprima na tela ambos RU usando ponteiros. O algoritmo também vai ter que comparar os dois RU usando ponteiros e imprimir na tela qual é o maior.

Resolução:

```

#include <stdlib.h>
#include <stdio.h>

// prototipagem
void flush();

int main() {

    /* Declaração das variáveis e dos ponteiros */
    int ru, valor, *p_ru, *p_valor;

    /* atribuição do endereço de memória das variáveis aos respectivos ponteiros*/
    p_ru = &ru;
    p_valor = &valor;

    /* requisição ao usuário pelo RU */
    printf("Digite seu RU: ");

    /* atribuição do valor digitado à variável através do ponteiro
       NOTA: o Visual Studio não permite a utilização da função
       scanf, então utilizei o scanf_s que tem o funcionamento equivalente */
    scanf_s("%d", p_ru);

    /* limpeza do buffer */
    flush();

    /* abaixo é feito exatamente o mesmo processo, mas dessa vez para o valor ao invés do RU */
    printf("\nDigite o valor para comparar ao RU: ");
    scanf_s("%d", p_valor);
    flush();

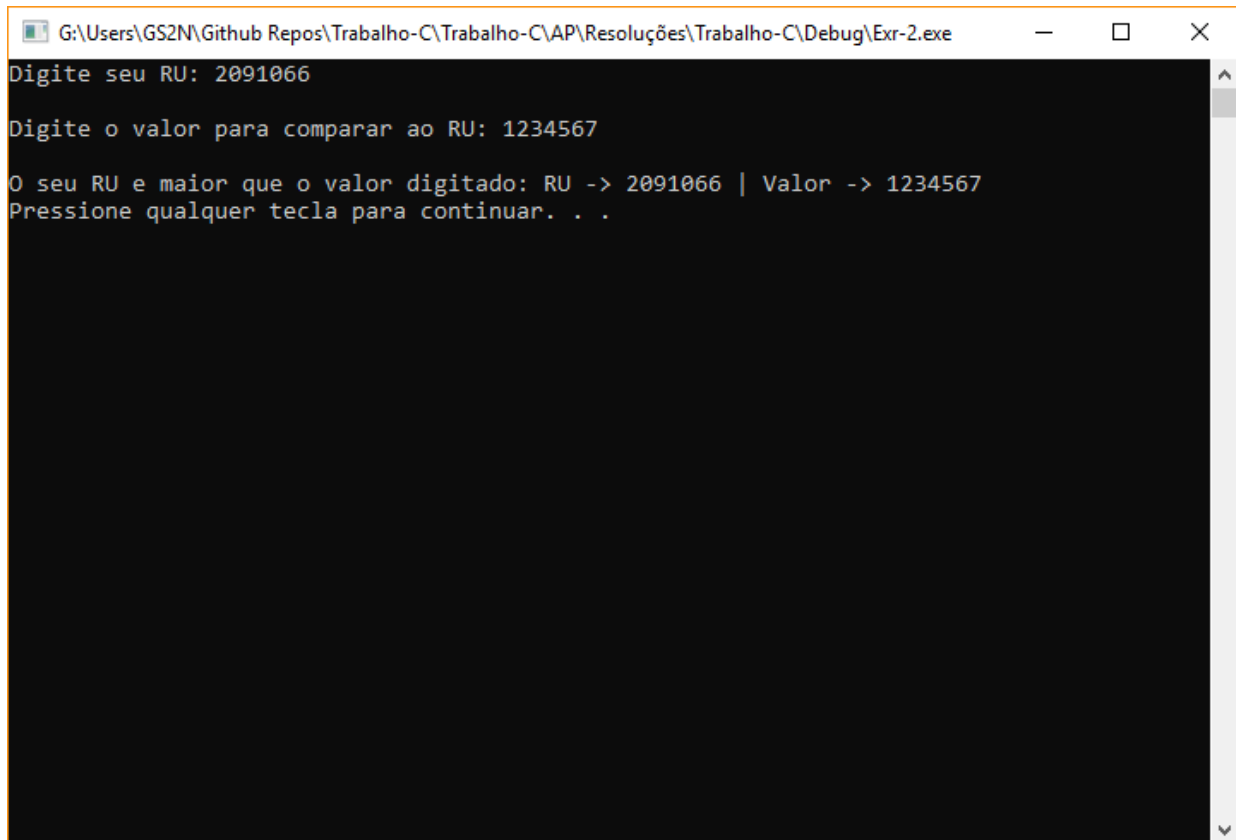
    /* comparação do valor com o RU através dos ponteiros utilizando operador ternário */
    (*p_ru > *p_valor) ?
        printf("\nO seu RU é maior que o valor digitado: RU -> %d | Valor -> %d \n", *p_ru, *p_valor) :
        printf("\nO valor digitado é maior que o seu RU: RU -> %d | Valor -> %d \n", *p_ru, *p_valor);

    system("pause");

    return 0;
}

// método responsável pela limpeza do buffer do teclado
void flush() {
    char c;
    while ((c = getchar()) != '\n' && c != EOF) {}
}

```



```
G:\Users\GS2N\Github Repos\Trabalho-C\Trabalho-C\AP\Resoluções\Trabalho-C\Debug\Exr-2.exe
Digite seu RU: 2091066
Digite o valor para comparar ao RU: 1234567
O seu RU e maior que o valor digitado: RU -> 2091066 | Valor -> 1234567
Pressione qualquer tecla para continuar. . .
```

Exercício 3:

Faça um algoritmo em linguagem C com as seguintes funcionalidades:

- Receba um registro, com dois campos, como dados de entrada.
- O primeiro campo é um vetor que vai armazenar o nome do aluno.
- O segundo campo é uma variável do tipo inteiro que vai armazenar o RU do aluno.
- Imprime na tela os dados armazenados na estrutura.

Resolução:

```

#include <stdlib.h>
#include <stdio.h>

// Declaração da Struct Aluno com os campos Nome e RU
struct aluno {
    char nome[50];
    int ru;
};

typedef struct aluno Aluno; // definição do tipo "Aluno" para facilitar a chamada

//prototipagem
void flush();
Aluno criar_aluno(char *nome, int ru);

int main() {

    //Declaração das variáveis que conterão a entrada do usuário
    char nome[50];
    int ru;

    printf_s("Digite o nome do aluno a ser inserido: ");
    // Atribuição da entrada do usuário para a variável nome em formato de "string"
    fgets(nome, sizeof(nome), stdin);

    // solicitação ao usuário do RU e atribuição do mesmo para a variável responsável
    printf_s("\nDigite o RU do aluno a ser inserido: ");
    scanf_s("%d", &(ru));
    flush();

    // nova variável aluno retornada pela função de construção de alunos
    Aluno aluno = criar_aluno(nome, ru);

    // impressão na tela dos membros do registro aluno
    printf_s("\n\n Nome: %s \n RU: %d\n\n", aluno.nome, aluno.ru);
    system("pause");
    return 0;
}

//função responsável pela criação de novos alunos
Aluno criar_aluno(char *nome, int ru) {
    Aluno aluno;
    int i;
    for (i = 0; i < 50; i++)
    {
        aluno.nome[i] = nome[i];
    }

    aluno.ru = ru;
    return aluno;
}

// método para a limpeza de buffer
void flush() {
    char c;
    while ((c = getchar()) != '\n' && c != EOF) {}
}

```

```
G:\Users\GS2N\Github Repos\Trabalho-C\Trabalho-C\AP\Resoluções\Trabalho-C\Debug\Exr-3.exe
Digite o nome do aluno a ser inserido: Gustavo Kuze da Silva
Digite o RU do aluno a ser inserido: 2091066

Nome: Gustavo Kuze da Silva
RU: 2091066
Pressione qualquer tecla para continuar. . .
```

Exercício 4:

Replique o exercício 3. Porém, agora, declare um ponteiro para a estrutura de dados heterogênea. No momento da leitura dos dados e da impressão na tela, use o ponteiro para buscar o conteúdo dos campos. Imprima na tela também o seu RU na tela

Resolução:

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

// Declaração da Struct Aluno com os campos Nome e RU
struct aluno {
    char nome[50];
    int ru;
};
typedef struct aluno Aluno; // definição do tipo "Aluno" para facilitar a chamada

//prototipagem
void flush();
int main() {

    //Declaração da variável ponteiro pra Aluno
    Aluno *aluno = (Aluno *)malloc(sizeof(Aluno));

    printf_s("Digite o nome do aluno a ser inserido: ");
    // Atribuição da entrada do usuário para o membro nome de aluno em formato de "string"
    fgets(aluno->nome, sizeof(aluno->nome), stdin);

    // solicitação ao usuário do RU e atribuição do mesmo para o membro ru de aluno
    printf_s("\nDigite o RU do aluno a ser inserido: ");
    scanf_s("%d", &(aluno->ru));
    flush();

    // impressão na tela dos membros do registro aluno usando ponteiros
    printf_s("\n\n Nome: %s \n RU: %d\n\n", aluno->nome, aluno->ru);
    system("pause");
    return 0;
}

// método para a limpeza de buffer do teclado
void flush() {
    char c;
    while ((c = getchar()) != '\n' && c != EOF) {}
}
```

```
G:\Users\GS2N\Github Repos\Trabalho-C\Trabalho-C\AP\Resoluções\Trabalho-C\Debug\Exr-4.exe
Digite o nome do aluno a ser inserido: Gustavo Kuze da Silva
Digite o RU do aluno a ser inserido: 2091066

Nome: Gustavo Kuze da Silva
RU: 2091066

Pressione qualquer tecla para continuar. . .
```

Exercício 5:

Faça um algoritmo em linguagem C que contenha dois números inteiros digitados na tela pelo usuário:

- a. O primeiro número marca um início;
- b. O segundo número marca um fim;

O algoritmo vai contar quantos números existem entre o início (primeira entrada) e o fim (segunda entrada). A impressão na tela do usuário deve ser realizada de duas formas:

- a. Iterativa;
- b. Recursiva;

Ao colocar no seu relatório uma imagem do seu código funcionando, coloque ele rodando utilizando como valor de início os 2 últimos valores do seu RU e valor final o número 99

Resolução:

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

//prototipagem
void flush();
int contar_iterativamente(int ini, int fim);
int contar_recursivamente(int *ini, int fim, int *contador);

int main() {

    //Declaração das variáveis e dos ponteiros
    int inicio = 0,
        final = 0,
        contador = 0,
        resultado_itera = 0,
        resultado_recurs = 0,
        *p_contador,
        *p_inicio,
        *p_final;
```



```

//atribuição dos ponteiros aos endereços de memória das variáveis
p_contador = &contador;
p_inicio = &inicio;
p_final = &final;

//requisição dos valores ao usuário, recolhidos através dos ponteiros
printf("Digite o numero de inicio: ");
scanf_s("%d", p_inicio);
printf("\nDigite o numero final: ");
scanf_s("%d", p_final);

//atribuição do retorno da função iterativa à variável resultado_itera
resultado_itera = contar_iterativamente(inicio, final);
//atribuição do retorno da função recursiva à variável resultado_recurs
resultado_recurs = contar_recursivamente(p_inicio, final, p_contador);

//impressão de ambos os resultados na tela
printf("\n\n+++++\n\nResultado iterativo: %d\n", resultado_itera);
printf("\nResultado recursivo: %d\n\n", resultado_recurs);

system("pause");

return 0;
}

int contar_iterativamente(int ini, int fim) {
    //declaração das variáveis de iteração e contagem
    int i, contador = 0;

    //faz a iteração desde o valor da variável ini até a variável fim
    for (i = ini; i < fim; i++)
    {
        contador++; //Incrementa o valor de contador em um
        printf("\n Contador iterativo ==> %d ", contador); //imprime na tela o valor atual de contador
    }
    return contador; //retorna o valor de contador como sendo a diferença entre ini e fim
}

int contar_recursivamente(int *ini, int fim, int *contador) {

    // caso o ponteiro ini (já) tenha seu valor identico ao da variável fim, retorne o valor do ponteiro
    //contador
    if (*ini == fim)
        return *contador;
    *contador += 1; // acrescenta 1 ao valor do ponteiro
    *ini += 1;

    printf("\n Contador recursivo ==> %d ", *contador); //imprime na tela o valor atual de *contador

    return contar_recursivamente(ini, fim, contador); //retorna uma nova chamada à função, fazendo assim a
    //recursividade
}

// método para a limpeza de buffer do teclado
void flush() {
    char c;
    while ((c = getchar()) != '\n' && c != EOF) {}
}

```

```
Contador iterativo ==> 24
Contador iterativo ==> 25
Contador iterativo ==> 26
Contador iterativo ==> 27
Contador iterativo ==> 28
Contador iterativo ==> 29
Contador iterativo ==> 30
Contador iterativo ==> 31
Contador iterativo ==> 32
Contador iterativo ==> 33
Contador recursivo ==> 1
Contador recursivo ==> 2
Contador recursivo ==> 3
Contador recursivo ==> 4
Contador recursivo ==> 5
Contador recursivo ==> 6
Contador recursivo ==> 7
Contador recursivo ==> 8
Contador recursivo ==> 9
Contador recursivo ==> 10
Contador recursivo ==> 11
Contador recursivo ==> 12
Contador recursivo ==> 13
Contador recursivo ==> 14
Contador recursivo ==> 15
Contador recursivo ==> 16
Contador recursivo ==> 17
Contador recursivo ==> 18
Contador recursivo ==> 19
Contador recursivo ==> 20
Contador recursivo ==> 21
Contador recursivo ==> 22
Contador recursivo ==> 23
Contador recursivo ==> 24
Contador recursivo ==> 25
Contador recursivo ==> 26
Contador recursivo ==> 27
Contador recursivo ==> 28
Contador recursivo ==> 29
Contador recursivo ==> 30
Contador recursivo ==> 31
Contador recursivo ==> 32
Contador recursivo ==> 33
```

+++++

Resultado iterativo: 33

Resultado recursivo: 33

Pressione qualquer tecla para continuar. . .