

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

#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);

**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:**

//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);

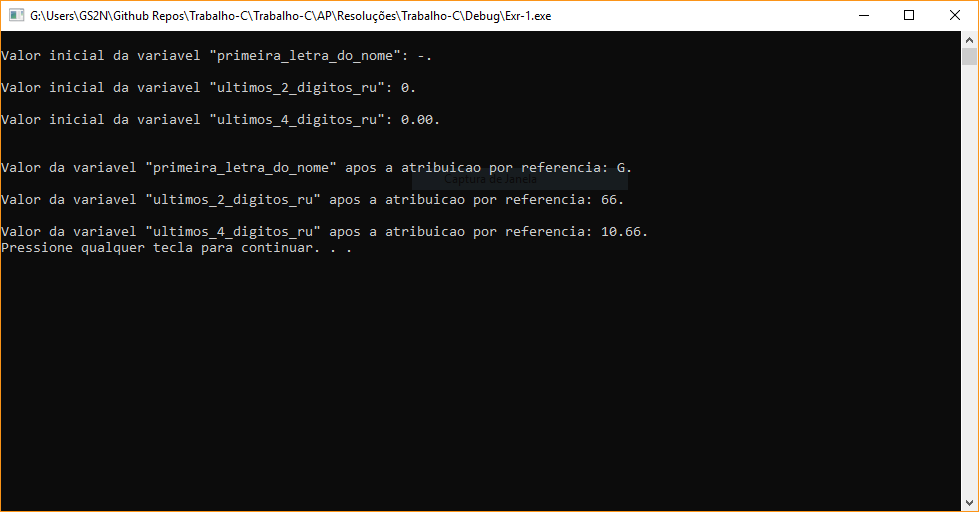
printf("\nValor da variavel \"ultimos\_2\_digitos\_ru\" apos a atribuicao por referencia: %d.\n", ultimos\_2\_digitos\_ru);

printf("\nValor da variavel \"ultimos\_4\_digitos\_ru\" apos a atribuicao por referencia: %.2f.\n", ultimos\_4\_digitos\_ru);

system("pause");

return 0;

}



**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 e maior que o valor digitado: RU -> %d | Valor -> %d \n", \*p\_ru, \*p\_valor) :

printf("\nO valor digitado e maior que o seu RU: RU -> %d | Valor -> %d \n", \*p\_ru, \*p\_valor);

system("pause");

return 0;

}

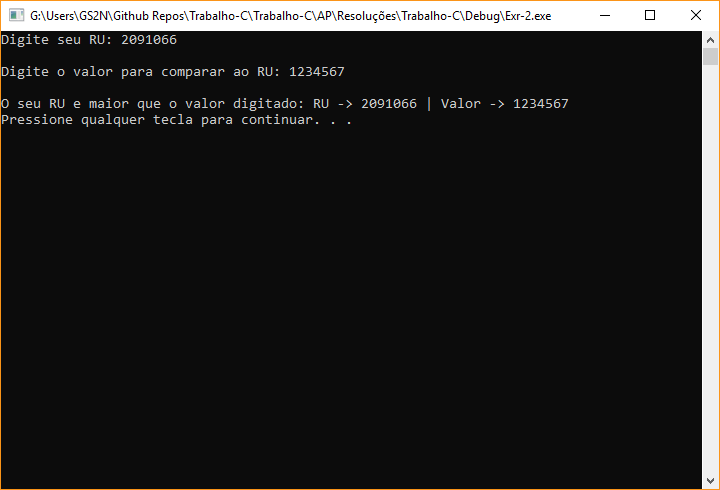
// método responsável pela limpeza do buffer do teclado

void flush() {

char c;

while ((c = getchar()) != '\n' && c != EOF) {}

}



**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);

// soliciação 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;

}

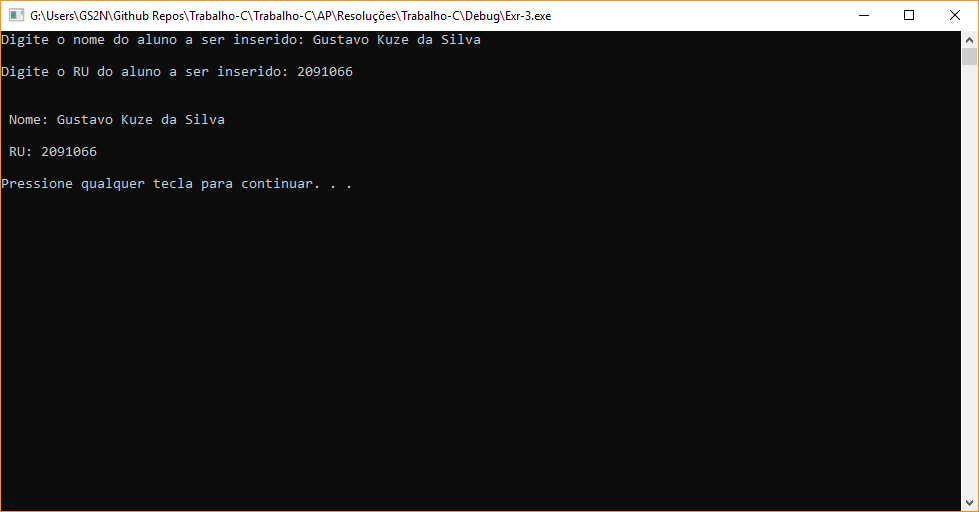
// método para a limpeza de buffer

void flush() {

char c;

while ((c = getchar()) != '\n' && c != EOF) {}

}



**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);

// soliciação 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;

}

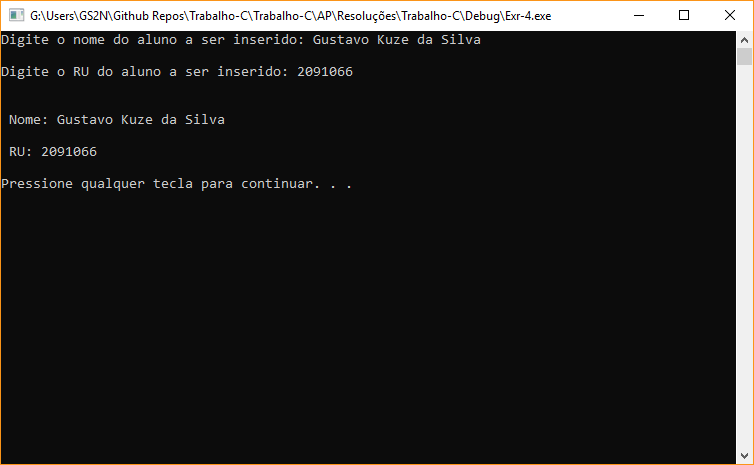
// método para a limpeza de buffer do teclado

void flush() {

char c;

while ((c = getchar()) != '\n' && c != EOF) {}

}



**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 inicio 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) {}

}

