

CENTRO UNIVERSITÁRIO INTERNACIONAL UNINTER

ESCOLA SUPERIOR POLITÉCNICA

BACHARELADO EM ENGENHARIA DA COMPUTAÇÃO

DISCIPLINA DE LINGUAGEM DE PROGRAMAÇÃO

ATIVIDADE PRÁTICA

gesiane gonçalves ferreira pajarinen – RU: 2466550

Prof. winston sen lun fung

Ipatinga – minas gerais

2020

# EXERCÍCIO 1

ENUNCIADO: 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.

Solução do aluno:

#include <stdio.h>

#include <stdlib.h>

int main() {

// Declarando três variáveis (inteiro, real e char):

int RUint;

float RUfloat;

char letra;

//Declarando três ponteiros:

int \*pRUint;

float \*pRUfloat;

char \*pletra;

//Associando as variáveis aos ponteiros:

pRUint = &RUint;

pRUfloat = &RUfloat;

pletra = &letra; //OU pontc = c

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ANTES\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

//Imprimindo na tela os valores das variáveis antes da modificação:

// Vou atriburi diretamente às variáveis os valores:

RUint = 24; //Os dois primeiros dígitos do meu RU.

RUfloat = 24.66; //Os 4 priemeiros dígitos do meu RU com 2 casa decimais.

letra = 'E'; // A última letra do meu nome.

printf("valor de RUint = %d \n", RUint);

printf("valor de RUfloat = %.2f \n", RUfloat);

printf("valor da letra = %c \n", letra);

printf("\n");

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Modificando os valores de cada variável indiretamente, usando os \* ponteiros associados:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

\*pRUint = 50; // Armazenando na variável inteira os dois últimos dígitos do meu RU.

\*pRUfloat = 65.50; // Armazenando na variável real os 4 últimos dígitos do meu RU, sendo os 2 últimos os valores com virgula.

\*pletra = 'G'; //Armazenando na variável char a primeira letra do meu nome.

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_DEPOIS\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

//Imprimindo na tela os valores das variáveis após a modificação:

printf("valor de RUint = %d \n", RUint);

printf("valor de RUfloat = %.2f \n", RUfloat);

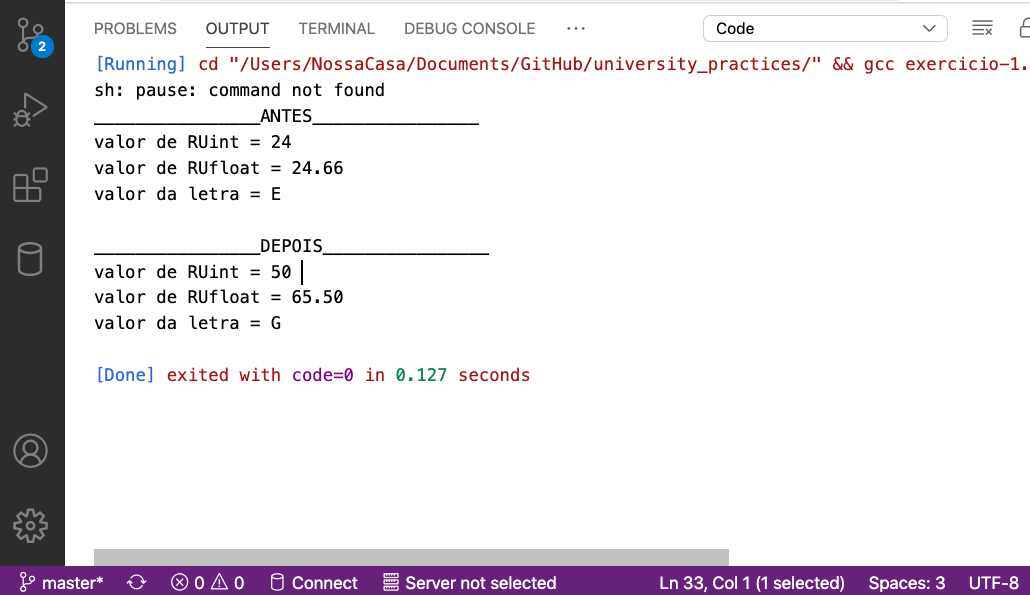
printf("valor da letra = %c \n", letra);

system("pause");

return 0;

}

Imagem do código funcionando:



# EXERCÍCIO 2

ENUNCIADO: 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.

Solução do aluno:

#include <stdio.h>

#include <stdlib.h>

int main() {

printf("\n\n- - - - - - - - - - - - - - - - - - - - \n\n"); //Para melhorar a aparência na execução do código.

//Declarando as variáveis e seus respectivos ponteiros:

int RU, valor, \*pRU, \*pvalor;

//Armazenando na memória o meu RU e o valor 1234567, ambos digitados pelo usuário na tela:

printf("Digite o seu RU: ");

scanf\_s("%d", &RU);

printf("Digite a sequência '1234567': ");

scanf\_s("%d", &valor);

pRU = &RU;

pvalor = &valor;

printf("\n");

//Imprimindo na tela ambos RU usando ponteiros:

printf("Imprimindo na tela ambos RU usando ponteiros: \n");

printf(">> Seu RU é: %d \n", \*pRU);

printf(">> A sequência digitada foi: %d \n", \*pvalor);

printf("\n");

//Comparando o RU e a sequência 1234567 usando ponteiros e imprimindo na tela qual é o maior:

if (\*pRU > \*pvalor){

printf("O seu RU %d é maio que a sequência %d.\n", \*pRU, \*pvalor);

}

else if(\*pRU < \*pvalor){

printf("A sequência %d é maio que seu RU %d. \n", \*pvalor, \*pRU);

}

else {

printf("O seu RU %d é igual à sequência %d.\n", \*pRU, \*pvalor);

}

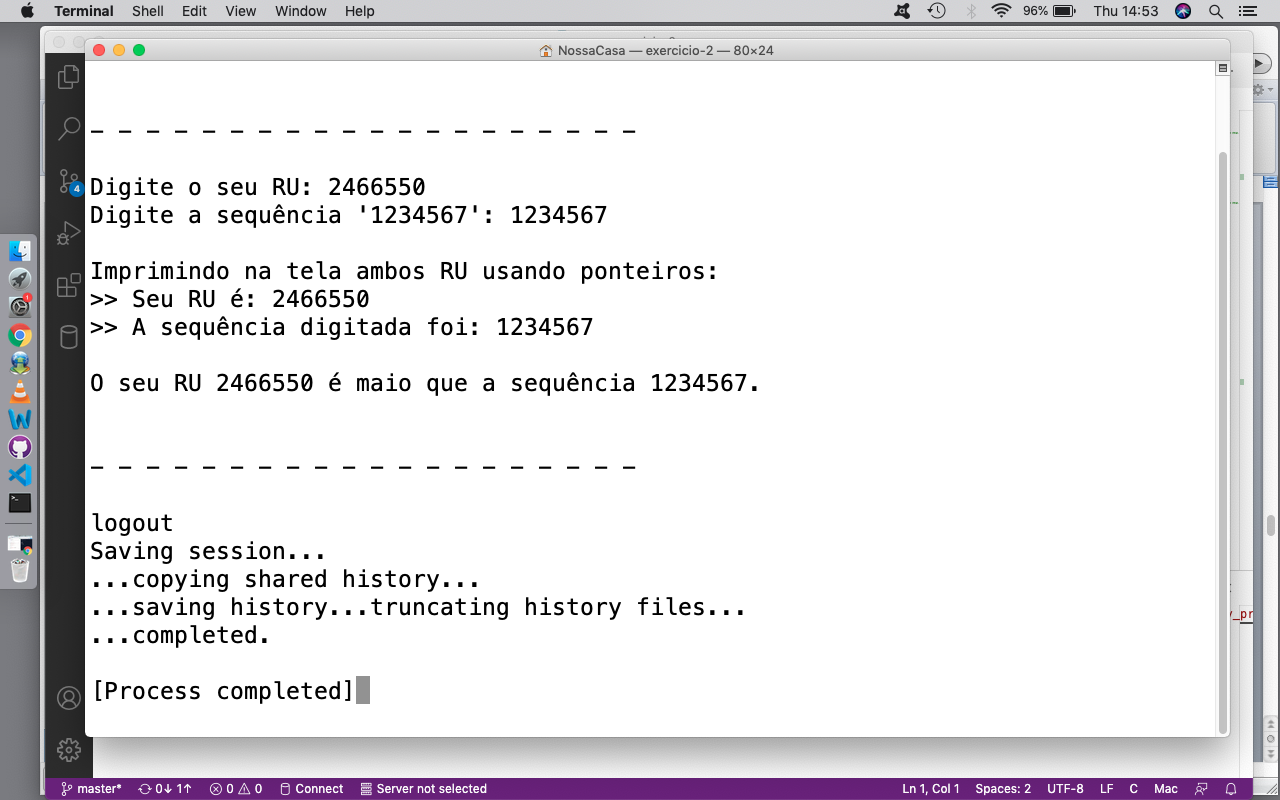
printf("\n\n- - - - - - - - - - - - - - - - - - - - \n\n"); //Para melhorar a aparência na execução do código.

system("pause");

return 0;

}

Imagem do código funcionando:



# EXERCÍCIO 3

ENUNCIADO: 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.

Solução do aluno:

#include <stdio.h>

#include <stdlib.h>

int main(){

//Registro, com dois campos, como dados de entrada:

struct registro\_aluno{

char nome[10]; //vetor que vai armazenar o nome do aluno.

int RU; //variável do tipo inteiro que vai armazenar o RU do aluno.

};

struct registro\_aluno aluno; //A struct agora vai ser chamada por aluno.

printf("\n\n- - - - - - - - - - - - - - - - - - - - \n\n"); //Para melhorar a aparência na execução do código.

//Pedindo as entradas do usuário:

printf("Digite seu NOME: \n >> ");

scanf\_s("%s", aluno.nome); //Salvando o nome na estrutura.

printf("Digite seu RU: \n >> ");

scanf\_s("%d", &aluno.RU); //Salvando o RU na estrutura.

printf("\n");

//Imprimindo na tela os dados armazenados na estrutura:

printf("Seu nome é: %s \n", aluno.nome);

printf("e seu RU é: %d \n", aluno.RU);

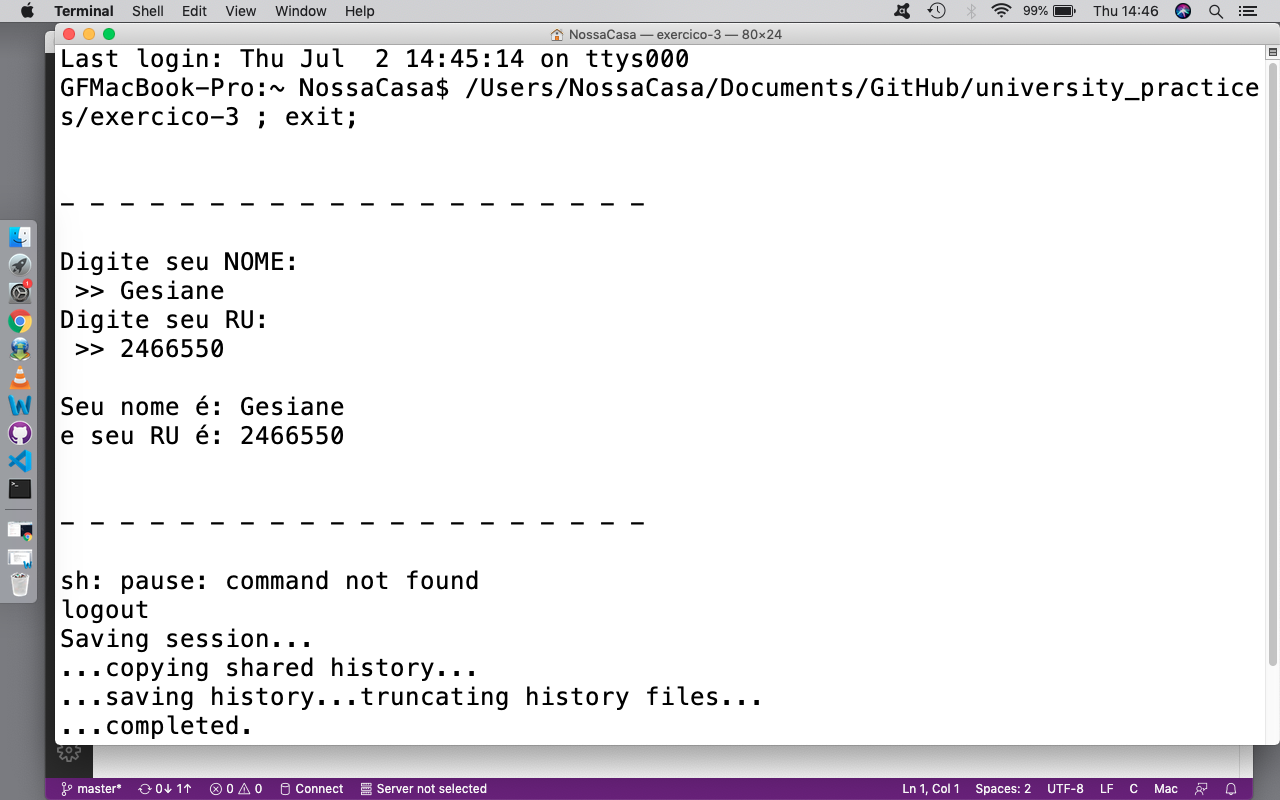
printf("\n\n- - - - - - - - - - - - - - - - - - - - \n\n"); //Para melhorar a aparência na execução do código.

system("pause");

return 0;

}

Imagem do código funcionando:



# EXERCÍCIO 4

ENUNCIADO: 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.

Solução do aluno:

#include <stdio.h>

#include <stdlib.h>

int main(){

//Registro, com dois campos, como dados de entrada:

struct registro\_aluno{

char nome[20]; //vetor que vai armazenar o nome do aluno.

int RU; //variável do tipo inteiro que vai armazenar o RU do aluno.

};

struct registro\_aluno aluno, \*p\_aluno; //Declarando um ponteiro para a estrutura de dados.

p\_aluno = &aluno; //Dando ao ponteiro o endereço da estrutura.

printf("\n\n- - - - - - - - - - - - - - - - - - - - \n\n"); //Para melhorar a apresentação na execução do código.

//Pedindo as entradas do usuário e colocando na estrutura através dos ponteiros:

printf("Digite seu NOME: \n >> ");

scanf\_s("%s", p\_aluno -> nome); //Salvando o nome na estrutura através do ponteiro.

printf("Digite seu RU: \n >> ");

scanf\_s("%d", &p\_aluno -> RU); //Salvando o RU na estrutura através do ponteiro.

printf("\n");

//Imprimindo na tela os dados armazenados na estrutura através dos ponteiros:

printf("\n Imprimindo os dados armazenados na estrutura através dos ponteiros:\n");

printf(" Seu nome é: %s \n", p\_aluno -> nome);

printf(" e seu RU é: %d \n", p\_aluno -> RU);

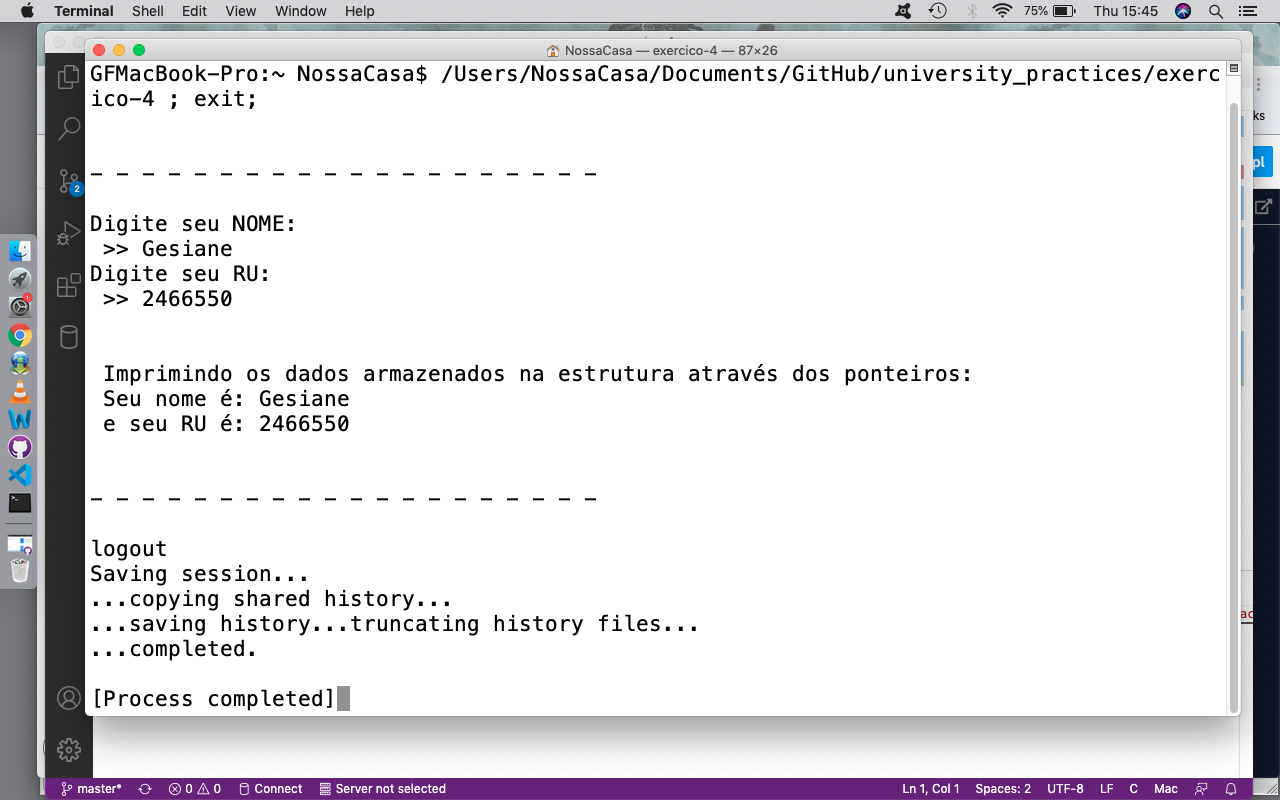
printf("\n\n- - - - - - - - - - - - - - - - - - - - \n\n"); //Para melhorar a apresentação na execução do código.

system("pause");

return 0;

}

Imagem do código funcionando:



# EXERCÍCIO 5

ENUNCIADO: 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.

Solução do aluno:

#include <stdio.h>

#include <stdlib.h>

int contandoNumeros (); //Anunciando a função que vai ser usada no main.

int main(){

//Declarando as variáveis:

int inicio, fim, resultado;

printf("\n\n- - - - - - - - - - - - - - - - - - - - \n\n"); //Para melhorar a aparência na execução do código.

//Pedindo as entradas do usuário:

printf("Digite o primeiro número: \n >> ");

scanf\_s("%d", &inicio); //ADICIONAR \_S //Número inicial.

printf("Digite o segundo número (maior que o primeiro): \n >> ");

scanf\_s("%d", &fim); //ADICIONAR \_S //Número final.

printf("Entre os números %d e %d, temos: \n\n", inicio, fim);

//Iterativa:

printf(" - Processando de forma Iterativa: \n");

for (int i= inicio+1; i<fim; i++){

printf("%d, ", i);

};

printf("\n");

//Recursiva:

printf("\n - Processando de forma Recursiva: \n");

//Chamando a função com seus argumentos:

return contandoNumeros (inicio, fim);

printf("\n\n- - - - - - - - - - - - - - - - - - - - \n\n"); //Para melhorar a aparência na execução do código.

sistem("pause");

return 0;

}

//Função Recursiva:

int contandoNumeros (int num1, int num2) {

num1++;

while (num1<num2) {//Enquanto o num1 for menor que num2:

printf("%d, ", num1); //Imprima o valor de num1 já com o acrescrécimo dado no início da função.

return contandoNumeros(num1, num2); //e refaça a função com o novo valor de num1.

};

printf("\n\n- - - - - - - - - - - - - - - - - - - - \n\n"); //Para melhorar a aparência na execução do código.

return 0;

}

Imagem do código funcionando:

