ANO
2025

UNINTER

ATIVIDADE PRÁTICA

LINGUAGEM DE PROGRAMAÇÃO

Prof. Winston Sen Lun Fung, Me.

INTRODUÇÃO

Olá a todos.

Sejam todos muito bem-vindos!

Esta avaliação foi planejada e preparada para as disciplinas de Linguagem de Programação Centro Universitário Internacional Uninter.

O objetivo desta atividade é fazer com que você, aluno, desenvolva os conhecimentos teóricos aprendidos na rota de maneira.

Ao longo desse roteiro serão passadas as orientações gerais para realização da avaliação bem como os seus critérios de correção.

No mais, desejo-lhe boa atividade prática em nome dos professores da disciplina de Linguagem de Programação.

SUMÁRIO

ORIENTAÇÕES GERAIS

ESTRUTURA DO CADERNO DE RESPOSTAS

1. **Capa**
    a. Inclua seu **nome completo** e seu **RU**.
    b. Caso não conste o nome e o RU, a atividade poderá ser zerada por falta de identificação.
2. **Organização Interna**
    a. Para cada exercício, siga a ordem:
    b. Enunciado (caso necessário, ou coloque apenas o número do exercício para referência).
3. **Código-Fonte Completo**
    a. Insira todo o código que você desenvolveu, desde a primeira até a última linha.
    b. O código deve estar indentado e organizado, facilitando a leitura.
    c. Adicione **comentários** no código para explicar, com suas palavras, o que cada parte ou trecho faz.
4. **Captura de Tela (Screenshot)**
    a. Após o código, inclua uma imagem mostrando o terminal em execução, exibindo o resultado do seu programa.
    b. Essa imagem serve para comprovar que seu código foi executado corretamente e está mostrando a saída solicitada.
5. **(Opcional) Breve Explicação (fora dos comentários)**
    a. Se achar necessário, você pode escrever, fora do código, alguma explicação adicional sobre a lógica, entradas, saídas ou possíveis casos de teste.

CUIDADO!

Em programação, não existem dois códigos exatamente iguais. Cada programador organiza seu código de uma forma diferente, declara variáveis com nomes diferentes, faz comentários diferentes, gera mensagens aos usuários distintas etc. Por este motivo, não serão aceitos dois algoritmos idênticos entre alunos (ou iguais à Internet). Caso o corretor observe respostas iguais, elas serão consideradas como PLÁGIO e será atribuída a NOTA ZERO na questão.

FORMATO DE ENTREGA

- ✓ Você deverá utilizar o **Caderno de Respostas** este arquivo vai unir todo o conteúdo (capa, códigos e capturas de tela) em um único arquivo.
- ✓ A Atividade Prática, em formato PDF, deve ser enviado no AVA-Univirtus, no campo/ícone disponibilizado para entrega de **Trabalhos**.
- ✓ O modelo do **Caderno de Respostas** (em Word) está disponível no AVA-Univirtus.
  - Baixe o modelo.
  - Preencha-o com suas soluções, seguindo a estrutura acima.
  - Exporte ou imprima em PDF para realizar o envio.
- ✓ Atenção: arquivos em formatos diferentes do PDF não serão corrigidos e a nota será zero.

CUIDADOS IMPORTANTES

1. **Evitar Plágio**
   a. Cada código deve ter a sua "digital": nomes de variáveis, estruturas de repetição, formatação, comentários, estilo de programação etc.
   b. Não serão aceitos dois códigos idênticos (entre alunos ou copiados da internet).
   c. Em caso de cópias, será atribuída nota zero por **plágio**.
2. **Validação do Código**
   a. Antes de inserir o código no Caderno de Respostas, teste-o no seu computador ou ambiente de programação para garantir que está compilando/executando sem erros.
   b. Corrija eventuais problemas de sintaxe ou lógica.
3. **Explicações e Comentários no Código**
   a. Use comentários para deixar claro o que cada parte faz.
   b. Comentar o código ajuda o professor/tutor a entender suas escolhas e facilita a correção.
   c. Comentários são fundamentais para a nota final, pois demonstram que você entende o que está fazendo.
4. **Captura de Tela**
   a. Mostre o prompt/terminal rodando e exibindo a saída do seu programa para cada questão.
   b. Se quiser colocar exemplos adicionais, é bem-vindo, mas não esqueça de mostrar o caso mínimo que o professor pede.

PASSO A PASSO PARA MONTAR SEU DOCUMENTO FINAL

1. **Abra o modelo de Caderno de Respostas (Word)** no AVA-Univirtus.
2. **Preencha a capa** com seu nome e RU.
3. Para cada exercício:
   a. Copie e cole **todo** o seu código-fonte.
      i. Lembre-se de adicionar comentários dentro do código.
   b. Insira a **captura de tela** do terminal após o código.
   c. (Opcional) Acrescente uma breve explicação depois do print, se necessário.
4. Ao concluir todos os exercícios, revise o documento.
   a. Verifique se há possíveis erros de digitação e se todos os comentários estão claros.
   b. Confirme se cada exercício está bem-organizado (código → print da execução).
5. **Exporte ou Salve em PDF**.
   a. Vá em “Arquivo” → “Salvar como” ou “Exportar” → escolha “PDF”.
   b. Verifique se o arquivo ficou correto (capa, códigos, imagens).
6. Faça o **envio** no ícone de “Trabalhos” do AVA-Univirtus.
   a. Verifique novamente se você enviou o PDF certo e se o nome do arquivo está adequado.

DICAS FINAIS

- ✓ **Testes de Funcionamento:** sempre rode seu programa várias vezes, testando diferentes entradas (quando aplicável), para garantir que ele se comporta conforme exigido.
- ✓ **Padronização de Variáveis e Funções:** dê nomes de variáveis claros, por exemplo, `nomeAluno`, `idade`, `calculaMedia()`, para que a lógica seja fácil de entender.
- ✓ **Comentários Objetivos**: faça comentários curtos e diretos, por exemplo:

```
# Aqui, solicitamos ao usuário que digite a idade
printf("Digite o seu nome: ");
fgets(nome,60,stdin);
```

- ✓ **Use Exemplos Simples:** caso queira ilustrar algo extra, pode colocar ao final do exercício, mas não esqueça de cumprir o que é solicitado.
- ✓ **Prazo de Entrega:** fique atento(a) à data limite de envio no AVA. Entregas após o prazo podem não ser aceitas ou terão descontos na nota, conforme as regras da disciplina.

CRITÉRIOS DE AVALIAÇÃO

1. **Código Fonte Completo e Organizado (20%)**
    a. O código deve compilar ou executar corretamente, sem erros de sintaxe.
    b. Deve estar bem indentado, com variáveis e funções que facilitem a compreensão.
2. **Comentários e Explicações (20%)**
    a. Utilize comentários no corpo do código para explicar com suas próprias palavras o que está sendo feito.
    b. Comentários claros facilitam a correção e demonstram compreensão.
3. **Captura de Tela Mostrando a Execução (20%)**
    a. A imagem no PDF deve exibir o resultado do seu programa em execução.
    b. Tenha cuidado para a imagem ficar legível.
4. **Corretude das Saídas (20%)**
    a. O programa deve apresentar as saídas corretas e atender ao que o exercício pede.
    b. Valem aqui a lógica e o funcionamento final.
5. **Originalidade e Autoria (20%)**
    a. Cada código deve refletir o trabalho do próprio aluno.
    b. Códigos idênticos, plagiados ou sem originalidade não serão aceitos.

Atenção:

Imagine o RU: 1 2 3 4 5 6 7

1	2	3	4	5	6	**7**
Primeiro dígito						**Último dígito**

ATIVIDADE PRÁTICA

PRÁTICA 01

Cálculo de Momento de Inércia de uma Seção Retangular (Engenharia) e Análise Básica. Neste exercício, você irá:

a) Solicitar ao usuário as dimensões (largura e altura) de uma viga com seção transversal retangular.

b) Calcular dois valores importantes na área de Engenharia Estrutural:

   a. Área da seção transversal (b × h)

   b. Momento de inércia dessa seção em relação ao eixo neutro, usando a fórmula:

$$I = \frac{b \times h^3}{12}$$

c) Exibir os resultados e verificar se a área resultante atinge ou excede determinado valor de referência (por exemplo, 100 cm²), interpretando de forma simples se a viga poderia suportar uma “carga básica”.

Dessa forma, você exercitará tanto o cálculo de propriedades de uma seção retangular (conceito importante na Engenharia), quanto a implementação de entrada, processamento e saída de dados em Linguagem C.

1. Objetivo

- Ler a largura (b) e a altura (h) de uma viga retangular (em cm).
- Calcular:
  • A área da seção transversal: A = b × h
  • O momento de inércia em relação ao eixo neutro (seção retangular): I = (b × h³) / 12
- Verificar se a área ≥ 100 cm² (por exemplo), informando se está apta ou não para suportar uma “carga básica”.

2. Passos Sugeridos

a) Ler b e h do teclado (como float/double).

b) Calcular a área e o momento de inércia.

c) Exibir os valores calculados.

d) Exibir a mensagem se A >= 100: “A viga está apta…”; e a mensagem se A < 100: “A viga não está apta ...”.

3. Exemplo de Saída

```
Digite a largura (cm): 10
Digite a altura (cm): 15

Area = 150.00 cm^2
Momento de inercia = 2812.50 cm^4
A viga esta apta para carga basica (A >= 100 cm^2).
```

4. Demonstrando o funcionamento do programa

- Informe a soma dois primeiros dígitos do seu RU para a largura.

- Informe a soma dois últimos dígitos do seu RU para a altura.

* Caso a soma resulte em zero informe o valor 15.

PRÁTICA 02

Análise de Leituras de Sensores c/ Alocação Dinâmica e Ponteiros

1. Objetivo

- Ler N leituras (ex.: temperaturas), usando malloc/calloc.
- Calcular a média, valor mínimo e valor máximo das leituras.

2. Passos Sugeridos

a) Pedir ao usuário “Quantas leituras?” (N).

b) Alocar dinamicamente (float *leituras = malloc(N*sizeof(float));).

c) Ler os N valores.

d) Calcular:

- Média (soma / N).
- Min e Max percorrendo o vetor.

e) Exibir todos os resultados.

3. Exemplo de Saída

```
Quantas leituras? 5
Digite as leituras:
22.5
24.0
21.8
23.2
25.1

Resultado:
- Média: 23.32
- Mínimo: 21.80
- Máximo: 25.10
```

4. Demonstrando o funcionamento do programa

- Informe a para “quantas leituras” a quantidade de dígitos do seu RU.
- As medidas deverão ser informadas pela composição: [digito do RU].[posição], informando, na ordem, do primeiro ao último dígito do seu RU.

Exemplo: RU 9876543

1° medida: 9.1

2° medida: 8.2

3° medida: 7.3

...

PRÁTICA 03

Criptografia Simples usando Ponteiros e Manipulação de Strings

1. Objetivo

- Ler uma frase (máx. 100 caracteres).
- Substituir vogais (A, E, I, O, U) por números (2,3,4,5,6).
- Substituir dígitos (0-9) por vogais, conforme um mapeamento que **você** definir.
- Percorrer a string via ponteiro de char.

2. Exemplo de Saída

```
Frase original: "Eu tenho 2 exames e 3 provas"
Frase criptografada: "36 t3nh4 i 3x2m3s 3 4 pr5v2s"
```

(mapeamento ilustrativo; defina o seu)

3. Demonstrando o funcionamento do programa

- Para a frase original informe o seu nome completo e o seu RU.

PRÁTICA 04

Imagine o projeto de um circuito eletrônico e para controle na sua fabricação necessita-se de uma lista de componentes de uma placa. Desenvolva uma Lista Encadeada de Componentes Eletrônicos.

1. Objetivo

- Implementar lista encadeada que guarda:
  - nome[50]
  - valor (por exemplo, ohms, microF etc.)
  - tipo (resistor, capacitor etc.)
- Inserir quantos itens o usuário quiser (utilizar alocação dinâmica de memória).
- Ao final, salvar num arquivo CSV.

2. Passos

a) Definir struct nó (contendo nome, valor, tipo e ponteiro -> próximo).

b) Funções para inserir (no fim ou início) e exibir.

c) Um menu no main:

- Inserir componente
- Listar componentes
- Gerar arquivo CSV com a listagem de componentes (utilizador como separador o ponto e vírgula)
- Sair

3. Demonstrando o funcionamento do programa

- Cadastre 10 componentes, solicite a listagem em tela dos componentes, grave o arquivo CSV e mostre o seu conteúdo.

PRÁTICA 05

Crie um Agendador Simples de Tarefas com Prioridade.

1. Objetivo

- Criar um “mini-sistema” que gerencia tarefas:
  • idTarefa (int)
  • descricao[100]
  • prioridade (1=alta, 2=media, 3=baixa)
- Manter numa lista encadeada ou fila de prioridade.
- Inserir, remover tarefa de maior prioridade, listar e salvar em CSV, utilizador como separador o ponto e vírgula.

2. Passos

a) struct Tarefa {int id; char desc[100]; int prio; ...}

b) Incluir funções para inserir (no fim ou com base na prioridade) e remover (sempre a de maior prioridade).

c) Menu:

- 1) Inserir Tarefa
- 2) Remover Tarefa Prioridade Máxima
- 3) Editar uma tarefa
- 4) Listar
- 5) Sair (salvar CSV – utilizador como separador o ponto e vírgula).

d) Todos os dados devem ser salvos em um arquivo CSV quando sair do programa, ao iniciar o programa todos os dados salvos devem ser recarregados, o nome do arquivo CSV deve ser o seu RU.

3. Exemplo de Saída

```
Menu:
1) Inserir Tarefa
2) Remover Tarefa Prioridade Máxima
3) Editar uma tarefa
4) Listar
5) Sair (salvar CSV).

Inserir Tarefa
ID_Tarefa:
Descricao da Tarefa:
Prioridade [1-Alta, 2-Media, 3-baixa]:
```

4. Demonstrando o funcionamento do programa

- Cadastre 8 atividades diferentes, mostre as telas de cadastro, tela com a listagem de o conteúdo salvo no arquivo CSV.

O que é um arquivo CSV?

Um arquivo CSV (*Comma-Separated Values*, ou "valores separados por vírgula") é um formato de arquivo simples, amplamente utilizado para armazenar e compartilhar dados de forma tabular (linhas e colunas). Mesmo sendo chamado de "*Comma-Separated Values*" no nome original em inglês, é bastante comum usar outros separadores além da vírgula, como o **ponto e vírgula (;)** ou mesmo tabulações, dependendo do padrão adotado ou da localidade.

- Como o CSV funciona?

1. Cada linha do arquivo representa um registro (ou uma linha de uma planilha).
2. Dentro de cada linha, os campos (colunas) são separados por um caractere específico - o separador.
   - No nosso caso, será o **ponto e vírgula (;)**.
   - Em outros contextos, pode ser a vírgula (,) ou o tab (\t).
3. Exemplo de duas linhas de arquivo CSV usando `;`

```
Nome;Idade;Cidade
Ana;23;Curitiba
```

- Como gravar dados em CSV?

No contexto de um programa em C, por exemplo, para salvar os registros em CSV:

1. Abri (ou criar) um arquivo de texto com extensão ".csv"
   `fopen("meuArquivo.csv", "w")`
2. Para cada registro (linha) que deseja armazenar:
   - Montar uma string no formato:
     `campo1;campo2;campo3`
   - Escrever essa string seguida de uma quebra de linha (`\n`) no arquivo.
3. Fechar o arquivo ao final
   `fclose(ponteiroDoArquivo);`

Exemplo de trecho de código em C:

```c
    FILE *arq = fopen("dados.csv", "w");
    if (arq != NULL) {
        // Suponha que temos nome, cpf, telefone, email
        fprintf(arq, "Nome;CPF;Telefone;Email\n"); // Cabeçalho
        fprintf(arq, "%s;%s;%s;%s\n", nome, cpf, telefone, email);
        // Repita para cada registro
        fclose(arq);
}
```

- Como recuperar dados de um CSV?

Para ler os dados de volta no programa (em C, por exemplo):

1. Abrir o arquivo CSV em modo de leitura
   `fopen("dados.csv", "r")`
2. Ler linha por linha usando funções como `` `fgets()` ``.
3. Para separar cada campo, você pode:
   - Usar a função `` `strtok()` `` indicando o caractere `` `;` `` como delimitador,
   - Ou manualmente varrer a linha até encontrar o `` `;` `` e separar campos.
4. Guardar cada campo em variáveis adequadas (ex.: nome, cpf, etc.).
5. Ao final, fechar o arquivo
   `fclose(arq);`

Exemplo resumido:

```
    FILE *arq = fopen("dados.csv", "r");
    char linha[256];
    if (arq != NULL) {
        while (fgets(linha, sizeof(linha), arq)) {
        // Retirar newline e processar
        char *campo1 = strtok(linha, ";");
        char *campo2 = strtok(NULL, ";");
        char *campo3 = strtok(NULL, ";");
        ...
        // Agora campo1 tem valor do primeiro campo, campo2 do
segundo,                                                    etc.
    }
    fclose(arq);
}
```

Por que CSV?
- Simples: É apenas texto puro, fácil de criar e ler.
- Portável: Pode ser aberto em diversas ferramentas, como Excel, Google Sheets ou mesmo editores de texto simples.
- Flexível: Cada linha corresponde a um registro, cada separador (;) define colunas.