ANO
2025

UNINTER

ATIVIDADE PRÁTICA

LINGUAGEM DE PROGRAMAÇÃO

Prof. Winston Sen Lun Fung, Me.

INTRODUÇÃO

Olá a todos.

Sejam todos muito bem-vindos a esta nova jornada de estudos!

Esta avaliação foi especialmente elaborada para as disciplinas de Linguagem de Programação do Centro Universitário Internacional UNINTER, visando consolidar os conhecimentos teóricos adquiridos ao longo da rota de estudos.

Neste roteiro, você encontrará:

- ✓ As orientações gerais para a realização da atividade,
- ✓ Os critérios de correção,
- ✓ A lista de exercícios que deverão ser resolvidos,
- ✓ E todas as informações necessárias para que você possa demonstrar, na prática, o que aprendeu.

Desejamos a todos um excelente desenvolvimento dos estudos e da atividade prática.
Que seja um período produtivo e repleto de aprendizados!

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

Desenvolver um programa em linguagem C que calcule a média ponderada de um aluno com base em suas notas e determine sua situação acadêmica.

1. Estrutura de Dados:
    a. Crie uma struct chamada `Notas` para armazenar as seguintes informações:
        i. Nota da APOL1 (valor inteiro entre 0 e 100)
        ii. Nota da APOL2 (valor inteiro entre 0 e 100)
        iii. Nota da Prova Objetiva (valor inteiro entre 0 e 100)
        iv. Nota da Atividade Prática (valor inteiro entre 0 e 100)
2. Entrada de Dados:
    a. Solicite ao usuário que digite cada uma das notas, garantindo que os valores estejam dentro do intervalo de 0 a 100.

    Exemplo de mensagem:

```
Digite a nota da APOL1 (0-100):
```

3. Cálculo da Média:
    a. Utilize os seguintes pesos para calcular a média ponderada:
        i. APOL1: 15%
        ii. APOL2: 15%
        iii. Prova Objetiva: 30%
        iv. Atividade Prática: 40%
    b. Fórmula:

*Média = (APOL1 * 0.15) + (APOL2 * 0.15) + (Prova Objetiva * 0.30) + (Atividade Prática * 0.40)*

4. Determinação da Situação:
    a. Classifique o aluno com base na média obtida:
        i. **Aprovado**: Média ≥ 70
        ii. **Exame**: 30 ≤ Média < 70
        iii. **Reprovado**: Média < 30
5. Saída do Programa:
    a. Exiba a média calculada e a situação do aluno.

Exemplo:

```
Média: 68.5
Situação: Exame
```

6. Para demonstrar o funcionamento:
    a. Use os dois primeiros dígitos do seu RU como nota da APOL1.
    b. Use os dois últimos dígitos do seu RU como nota da APOL2.
    c. Escolha valores aleatórios entre 0 e 100 para a Prova Objetiva e Atividade Prática.
    d. Faça as capturas de tela, com as entradas de dados e os resultados para o seu caderno de respostas.

PRÁTICA 02

Desenvolver um programa em linguagem C que analise um vetor de caracteres contendo o RU e o nome completo do usuário e determine a quantidade de vogais, consoantes e espaços em branco utilizando um ponteiro.

1. Estrutura de Dados:
    a. Crie um vetor de caracteres com 120 posições para armazenar o RU e o nome completo do usuário.
2. Entrada de Dados:
    a. Solicite ao usuário que digite seu RU e seu nome completo, garantindo que o texto digitado caiba no vetor.

    Exemplo de mensagem:

```
Digite seu RU e nome completo:
```

3. Processamento:
    a. Utilize um ponteiro para percorrer o vetor.
    b. Verifique e conte:
        i. Vogais: Considere as letras 'a', 'e', 'i', 'o', 'u' (maiúsculas e minúsculas).
        ii. Consoantes: Considere todas as letras do alfabeto que não são vogais (também levando em conta letras maiúsculas e minúsculas).
        iii. Espaços em branco: Conte os caracteres de espaço (' ').
4. Saída do Programa:
    a. Exiba a quantidade de vogais, consoantes e espaços encontrados.

    Exemplo de saída:

```
Vogais: 12
Consoantes: 18
Espaços: 3
```

5. Demonstração do Funcionamento:
    a. Execute o programa utilizando seu RU e seu nome completo.
    b. Faça as capturas de tela do terminal que mostrem as entradas de dados e os resultados obtidos para anexar ao seu caderno de respostas.

PRÁTICA 03

Desenvolver um programa em linguagem C que utilize uma função recursiva para calcular a resistência equivalente de um conjunto de resistores conectados em paralelo. Para resistores em paralelo, a resistência equivalente é dada por:

$$R_{eq} = \frac{1}{\sum_{i=1}^{n} \frac{1}{n}}$$

1. Entrada de Dados:
    - Crie um vetor do com o tamanho igual a quantidade de dígitos do seu RU.
    - Para cada resistor, solicite o valor da resistência (em ohms).

Exemplo de mensagens:

```
Digite a quantidade de resistores:
Digite o valor da resistência do resistor 1 (ohms):
Digite o valor da resistência do resistor 2 (ohms):
```

2. Processamento**:**

- Armazene os valores das resistências em um vetor, o tamanho do vetor deve ser a quantidade de dígitos do seu RU.
- Implemente uma **função recursiva** que calcule a soma dos inversos das resistências:
    - **Protótipo da função:**
      `float somaInversos(float resistores[], int indice, int n);`
    - **Caso Base:** Se `indice == n`, retorne 0.
    - **Passo Recursivo:** Retorne `1/resistores[indice] + somaInversos(resistores, indice + 1, n)`.
- Após calcular a soma dos inversos, determine a resistência equivalente:

$$R_{eq} = \frac{1}{soma\ dos\ inversos}$$

3. Saída do Programa:

- Exiba a resistência equivalente do circuito paralelo.

Exemplo de saída:

```
Resistência equivalente do circuito paralelo: 5.23 ohms
```

4. Demonstração do Funcionamento:

- Execute o programa onde cada resistor é um dígito do seu RU, caso o dígito seja zero, utilizar o valor do dígito anterior.
- Faça capturas de tela do terminal mostrando as entradas e os resultados obtidos para anexar ao seu caderno de respostas.

PRÁTICA 04

Desenvolver um programa em linguagem C que calcule a quantidade mínima de lâmpadas necessárias para iluminar um cômodo, utilizando a norma NBR 5413. O programa deverá solicitar apenas a largura, o comprimento e o tipo do cômodo.

1. Entrada de Dados:

- ✓ Solicite ao usuário que informe:
  - Tipo do cômodo:

    Selecione uma das opções abaixo:

    1 - Quarto

    2 - Escritório

    3 – Cozinha
- ✓ Comprimento do cômodo (em metros)
- ✓ Largura do cômodo (em metros)

- ✓ Comprimento do cômodo (em metros)

Exemplo de mensagens:

```
Selecione o tipo de cômodo:
1 - Quarto
2 - Escritório
3 - Cozinha
Digite a opção desejada:

Digite o comprimento do cômodo (m):
Digite a largura do cômodo (m):
```

2. Processamento:

- ✓ Determinar a Iluminância Recomendada:

Utilize a seguinte tabela, baseada na NBR 5413:

Tipo de Cômodo	Iluminância Recomendada (lux)
1 - Quarto	150
2 - Escritório	300
3 - Cozinha	300

- ✓ Cálculo da Área do Cômodo:

$$Área = largura \times comprimento$$

- ✓ Parâmetros Fixos do Sistema:

  Para simplificar, considere os seguintes valores:

- Fluxo luminoso de cada lâmpada: 800 lúmens
- Fator de manutenção: 0.8

✓ Cálculo do Fluxo Luminoso Total Necessário:

Para atingir a iluminância recomendada, o fluxo total necessário (em lúmens) é dado por:

$$Fluxo\ Total = \frac{Área\ \times Iluminância\ Recomendada}{Fator\ de\ Manutenção}$$

✓ Determinar a Quantidade Mínima de Lâmpadas:

Divida o fluxo total necessário pelo fluxo luminoso de cada lâmpada:

$$Número\ de\ Lãmapadas = \frac{Fluxo\ Total}{800}$$

Caso o resultado não seja um número inteiro, arredonde-o para cima (utilize a função `ceil()` da biblioteca `<math.h>`).

✓ Saída do Programa:

- Exiba ao usuário:
  - O tipo de cômodo selecionado e a iluminância recomendada correspondente.
  - A área calculada do cômodo.
  - O fluxo luminoso total necessário.
  - A quantidade mínima de lâmpadas recomendadas.

Exemplo de saída:

```
Tipo de cômodo: Escritório (Iluminância recomendada: 300 lux)
Área do cômodo: 20.00 m²
Fluxo luminoso total necessário: 7500.00 lúmens
Quantidade mínima de lâmpadas recomendadas: 10
```

4. Demonstração do Funcionamento:

✓ Execute o programa informando como largura a soma dos dois primeiros dígitos do seu RU e para o comprimento os dois últimos dígitos do seu RU. **Atenção: caso a soma resulte em ZERO utilize o valor 12.3** .

✓ Faça capturas de tela do terminal exibindo as entradas e os resultados obtidos, e anexe ao seu caderno de respostas.

Dicas para Implementação:

✓ Utilize as bibliotecas `<stdio.h>` para entrada e saída de dados e `<math.h>` para a função `ceil()`.

✓ Valide a opção selecionada para o tipo de cômodo, garantindo que seja 1, 2 ou 3.

- Certifique-se de realizar os cálculos com precisão (use o tipo float ou double conforme necessário).