Roteiro
Aula Prática

ALGORITMOS E LÓGICA DE PROGRAMAÇÃO

Público

ROTEIRO DE AULA PRÁTICA

NOME DA DISCIPLINA: ALGORITMOS E LÓGICA DE PROGRAMAÇÃO

Unidade: U2 _ELEMENTOS DE ALGORITMOS

Aula: A2_ EXECUÇÃO SEQUENCIAL E ESTRUTURAS DE DECISÃO

Tempo previsto de execução de aula prática: 2h (CAMPO OBRIGATÓRIO – NÃO APARECER EM NENHUM RAP)

OBJETIVOS (campo obrigatório – exibição para todos)

Definição dos objetivos da aula prática:

- Interpretação dos requisitos e desenvolvimento correto do algoritmo abrangendo todas as possibilidades distintas na execução do programa.
- Aplicar uma estrutura de decisão: (“SE” ou “”ESCOLHA CASO”) no desenvolvimento de algoritmos computacionais com a pseudolinguagem Portugol.

INFRAESTRUTURA (OBRIGATÓRIO SE HOUVER – EXIBIÇÃO DOCENTE/TUTOR)

Instalações – Materiais de consumo – Equipamentos:

Laboratório de Informática

Materiais de consumo:

NSA

Equipamentos:

Computador com acesso à internet, e com o mínimo de 4 GB de Memória RAM.

SOLUÇÃO DIGITAL (OBRIGATÓRIO SE HOUVER - APARECER PARA TODOS)

Portugol WebStudio:

O Portugol WebStudio é uma ferramenta online de ensino e de aprendizagem de algoritmos baseada no Portugol (Português Estruturado), que é uma pseudolinguagem de programação para fins acadêmicos.

Tipo de Licença: GPL (GNU General Public License)

PROCEDIMENTO PARA INSTALAÇÃO: A ferramenta Portugol Web Studio não necessita de instalação, basta somente acessar por meio de uma conexão web o endereço (URL) a seguir:
LINK: **https://portugol.dev/**

EQUIPAMENTO DE PROTEÇÃO INDIVIDUAL (EPI) (CAMPO OBRIGATÓRIO – APARECER PARA TODOS)

NSA

PROCEDIMENTOS PRÁTICOS (OBRIGATÓRIO – TODOS)

Procedimento/Atividade nº 1 (Virtual)

Atividade proposta:

Você é um explorador corajoso que se aventura em uma misteriosa **Floresta Sussurrante** em busca de tesouros lendários e segredos perdidos. Ao adentrar na floresta, você se depara com três caminhos diferentes, cada um levando a uma parte desconhecida e perigosa da floresta. Sua missão é escolher sabiamente o caminho a seguir, enfrentando desafios e tomando decisões que impactarão sua jornada e sua pontuação final.

Prepare-se para esta aventura cheia de mistérios!

Caminho 1: O Caminho das Sombras

Descrição: Este caminho é cercado por árvores antigas e sombrias, com raios de lua penetrando entre os galhos. Parece ser o caminho mais misterioso e perigoso da floresta.

Obstáculo: O jogador encontra uma criatura mágica guardiã do caminho, que exige um enigma para deixá-lo passar. O enigma é: "Quem sou eu? Tenho olhos, mas não vejo. Tenho boca, mas não falo. O que sou?" (Resposta: uma caveira).

Premiação: Se o jogador responder corretamente ao enigma, ele encontra um baú escondido contendo uma gema preciosa que vale 100 pontos.

Caminho 2: O Caminho da Luz

Descrição: Este caminho é iluminado por raios de sol que filtram entre as copas das árvores. Parece ser o caminho mais seguro e reconfortante da floresta.

Obstáculo: O jogador encontra uma ponte quebrada sobre um rio turbulento. Ele deve decidir se tentará atravessar a ponte quebrada ou procurará um desvio seguro.

Premiação: Se o jogador decidir atravessar a ponte quebrada com sucesso, ele encontra uma fonte mágica que restaura sua saúde e adiciona 50 pontos à sua pontuação.

Caminho 3: O Caminho das Criaturas

Descrição: Este caminho é repleto de sons estranhos e pegadas misteriosas no chão. Parece ser o caminho mais imprevisível e enigmático da floresta.

Obstáculo: O jogador se depara com uma criatura mágica adormecida bloqueando o caminho. Ele deve decidir se tentará contornar a criatura com cuidado ou acordá-la para passar.

Premiação: Se o jogador decidir contornar a criatura com sucesso, ele encontra uma árvore encantada que concede a ele uma habilidade especial de camuflagem, adicionando 75 pontos à sua pontuação.

Com esses três caminhos, o jogador terá que tomar decisões estratégicas para superar os obstáculos e acumular o máximo de pontos possível em sua aventura pela Floresta Sussurrante.

Procedimentos para a realização da atividade:

Aplicação dos fundamentos da lógica de programação e algoritmos por meio de instruções de entrada e saída, criação de variáveis e constantes, sobretudo a utilização da estrutura de decisão (“SE” ou “ESCOLHA CASO”) em Portugol para resolução da atividade.

1. **Desenvolvimento do Programa:**

- Utilizar a linguagem Portugol no Portugol Web Studio.

  O estudante deverá utilizar a linguagem de programação “Portugol” por meio da ferramenta Portugol WebStudio acessando-o na url: https://portugol.dev/ . Na ferramenta o estudante chegará nesta tela e deverá clicar no botão “Novo Arquivo”:

- Implementar os três desvios condicionais: simples, composto e ou encadeado, criando um menu de opções para o jogador, por exemplo:

```
funcao inicio()
{
    inteiro opcao
    escreva("Você é um explorador corajoso que se aventura em uma misteriosa Floresta Sussurrante em busca de tesouros lendários e segredos perdidos.\n")
    escreva("Ao adentrar na floresta, você se depara com três caminhos diferentes:\n")
    escreva("1. O Caminho das Sombras\n")
    escreva("2. O Caminho da Luz\n")
    escreva("3. O Caminho das Criaturas\n")
    escreva("Escolha um caminho (1, 2 ou 3): ")
    leia(opcao)
```

  Observe que deverá ser criado um menu que apareça no momento da execução do programa para capturar a opção escolhida pelo usuário: 1, 2 ou 3.

- De acordo com a opção escolhida, o programa deve desviar a execução do código para o trecho que irá realizar as instruções de acordo com os caminhos do cenário: “**floresta sussurrante**”. Observe a seguir:

```
escolha(caminho)
{
    caso 1:
        caminho_das_sombras()
        pare

    caso 2:
```

O estudante pode utilizar a estrutura de seleção “ESCOLHA CASO” e executar as instruções para o “caminho das sombras” caso o usuário escolha a opção 1. Dessa forma, o algoritmo tem que executar todas as instruções que estão no “**Caminho 1: O Caminho das Sombras**”. Isso inclui: a apresentação da descrição, obstáculo e premiação, sendo esta de acordo com a resposta do usuário.

O mesmo princípio deve ser seguido quando o usuário escolher as opções “2” ou “3”. Ainda se o usuário escolher uma opção diferente, o algoritmo deve exibir uma mensagem:

```
caso contrario:
    escreva("Escolha inválida. Por favor, selecione um caminho válido.\n")
    leia(caminho)
    escolher_caminho(caminho)
```

- Criar uma narrativa envolvente que guie o usuário ao longo da aventura.

```
funcao caminho_das_criaturas()
{
    escreva("Você escolheu o Caminho das Criaturas.\n")
    escreva("Este caminho é repleto de sons estranhos e pegadas misteriosas no chão. Parece ser o caminho mais imprevisível e enigmático da floresta.\n")
    escreva("Você se depara com uma criatura mágica adormecida bloqueando o caminho. Você deve decidir se tentará contornar a criatura com cuidado ou acordá-la para passar.\n")
    escreva("Digite 'contornar' para tentar contornar a criatura ou 'acordar' para acordá-la: ")
    cadeia decisao
    leia(decisao)

    se (decisao == "contornar")
```

Utilizar o próprio texto fornecido na “atividade proposta” ficando a critério do estudante a customização e a implementação de melhorias no texto, pensando em envolver o usuário nesta aventura. Observar que poderão ser utilizadas as duas estruturas de seleção durante o desenvolvimento. Nesta imagem tem um exemplo da estrutura “SE” sendo implementada e para execução do menu de opções está sendo utilizado a estrutura “ESCOLHA CASO”.

2. Testes e Correções:

- Após toda a construção do programa, é hora de executar os testes.
- Testar o programa para garantir que todas as opções de escolha funcionem corretamente.
- Executando o código, de início deverá aparecer um texto explicando o programa e em seguida o menu de opções.

```
Você é um explorador corajoso que se aventura em uma misteriosa Floresta Sussurrante em busca de tesouros lendários e segredos perdidos.
Ao adentrar na floresta, você se depara com três caminhos diferentes:
1. O Caminho das Sombras
2. O Caminho da Luz
3. O Caminho das Criaturas
Escolha um caminho (1, 2 ou 3):
```

Neste ponto o usuário vai entrar com a opção escolhida e em seguida o algoritmo deve executar as instruções de acordo com a opção escolhida.

Verificar se o mesmo está acontecendo para as três opções do programa, inclusive quando o usuário escolhe incorretamente, ou seja, um número diferente de: 1, 2 ou 3.

```
caso contrario:
    escreva("Escolha inválida. Por favor, selecione um caminho válido.\n")
    leia(caminho)
    escolher_caminho(caminho)
```

3. **Documentação e Comentários:**

- Documentar o código fonte, explicando a lógica por trás de cada parte do programa.

Optar sempre por documentar o código fonte, principalmente quando não se está habituado com o comando, exemplo:

```
escolha(caminho)
{
    //quando a variável caminho for 1, o programa executa a função: caminho_das_sombras()
    caso 1:
        caminho_das_sombras()
        pare

    caso 2:
```

Colocando "//" antes da linha o programa ignora a execução da linha. Portanto, as anotações poderão ser realizadas em todo o código-fonte.

4. **Apresentação e Avaliação:**

- Apresentar o programa em sala de aula, demonstrando como funciona a aventura na Floresta Encantada.

```
Você é um explorador corajoso que se aventura em uma misteriosa Floresta
Ao adentrar na floresta, você se depara com três caminhos diferentes:
1. O Caminho das Sombras
2. O Caminho da Luz
3. O Caminho das Criaturas
Escolha um caminho (1, 2 ou 3): 2
```

- Após a construção do programa, testes e correções, o programa deverá ser executado passando por todas as opções sem que apresente erros ou falhas.
- Avaliar a pontuação dos estudantes com base nas escolhas que fizeram durante a construção do algoritmo, discutindo estratégias para maximizar a performance no desenvolvimento de soluções como a proposta por esta atividade.

Checklist:

- Analisar o cenário para criação do código-fonte do programa;
- Fazer a estruturação dos 3 caminhos, seus obstáculos e premiações previstas no percurso;
- Desenvolver o algoritmo em "Portugol" na ferramenta "Portugol WebStudio";

- Executar testes e possíveis correções;
- Apresentar o programa e verificar seu correto funcionamento.

RESULTADOS (obrigatório – aparecer para todos)

Resultados de Aprendizagem:

O objetivo da atividade é a correta compreensão e desenvolvimento do programa com os requisitos exigidos contendo a codificação completa para o Portugol WebStudio.

ESTUDANTE, VOCÊ DEVERÁ ENTREGAR (não obrigatório – aparecer para todos)

Descrição orientativa sobre a entregada da comprovação da aula prática:

O estudante deve entregar um arquivo em PDF contendo toda a codificação do exercício documentada, ou seja, para cada trecho do código-fonte, o estudante deve anexar um texto explicando o que acontece quando o trecho do código é executado.

REFERÊNCIAS BIBLIOGRÁFICAS (não obrigatório – aparecer para todos)

Descrição (em abnt) das referências utilizadas

CORMEN, Thomas. Algoritmos – Teoria e Prática. 3.ed. Rio de Janeiro: LTC, 2022.

MENÉNDEZ, Andrés. Simplificando algoritmos. 1. ed. - Rio de Janeiro: LTC, 2023.

SILVA, Flávio Soares Corrêa D.; FINGER, Marcelo; MELO, Ana Cristina Vieira. Lógica para computação. 2. ed. São Paulo: Cengage Learning, 2017.

NOME DA DISCIPLINA: ALGORITMOS E LÓGICA DE PROGRAMAÇÃO

Unidade: U2 _ELEMENTOS DE ALGORITMOS

Aula: A3_ ESTRUTURAS DE REPETIÇÃO

Tempo previsto de execução de aula prática: 2h (CAMPO OBRIGATÓRIO – NÃO APARECER EM NENHUM RAP)

OBJETIVOS (campo obrigatório – exibição para todos)

Definição dos objetivos da aula prática:

- Interpretação dos requisitos e desenvolvimento correto do algoritmo abrangendo todas as possibilidades distintas na execução do programa.

- Aplicar na prática a estrutura de repetição: (“ENQUANTO FAÇA”) no desenvolvimento de algoritmos computacionais com a pseudolinguagem Portugol.

INFRAESTRUTURA (OBRIGATÓRIO SE HOUVER – EXIBIÇÃO DOCENTE/TUTOR)

Instalações – Materiais de consumo – Equipamentos:

Laboratório de Informática

Materiais de consumo:

NSA

Equipamentos:

Computador com acesso à internet, e com o mínimo de 4 GB de Memória RAM.

SOLUÇÃO DIGITAL (OBRIGATÓRIO SE HOUVER - APARECER PARA TODOS)

Portugol WebStudio:

O Portugol WebStudio é uma ferramenta online de ensino e de aprendizagem de algoritmos baseada no Portugol (Português Estruturado), que é uma pseudolinguagem de programação para fins acadêmicos.

Tipo de Licença: GPL (GNU General Public License)

PROCEDIMENTO PARA INSTALAÇÃO: A ferramenta Portugol Web Studio não necessita de instalação, basta somente acessar por meio de uma conexão web o endereço (URL) a seguir:
LINK: **https://portugol.dev/**

EQUIPAMENTO DE PROTEÇÃO INDIVIDUAL (EPI) (CAMPO OBRIGATÓRIO – APARECER PARA TODOS)

NSA

PROCEDIMENTOS PRÁTICOS (OBRIGATÓRIO – TODOS)

Procedimento/Atividade nº 1 (Virtual)

Atividade proposta:

Suponha que você está em uma missão para encontrar um tesouro escondido no meio de um labirinto muito perigoso. O labirinto é composto por uma série de corredores estreitos, bifurcações e salas ocultas. Sua missão é encontrar o caminho para o tesouro, evitando armadilhas e obstáculos ao longo do caminho.

Descrição do Labirinto:

- Existem caminhos que não têm saída, fazendo com que você precise voltar atrás e tentar outro caminho.

- Algumas portas podem levar de volta ao início do labirinto, obrigando-o a recomeçar sua busca.
- O tesouro está escondido em um local diferente a cada vez que o programa iniciar, tornando a exploração do labirinto desafiadora e imprevisível.

Boa sorte em sua busca pelo tesouro!

Com essa descrição, têm-se um cenário envolvente e desafiador para criação de um algoritmo usando o comando "Enquanto-Faça" em Portugol para guiar o explorador pelo labirinto até encontrar o tesouro.

Procedimentos para a realização da atividade:

Aplicação dos fundamentos da lógica de programação e algoritmos, criação de variáveis e constantes, além de comandos da estrutura de repetição “Enquanto-Faça” em Portugol para resolução da atividade.

1. **Configuração da Ferramenta:**

- Utilizar a linguagem Portugol no Portugol Web Studio.

  O estudante deverá utilizar a linguagem de programação “Portugol” por meio da ferramenta Portugol WebStudio acessando-o na url: https://portugol.dev/ . Na ferramenta o estudante chegará nesta tela e deverá clicar no botão “Novo Arquivo”:

- Descrever o cenário do labirinto e a missão do “explorador” em encontrar o tesouro.

  Explicar o objetivo da atividade: utilizar o comando "Enquanto-Faça" para guiar o explorador pelo labirinto até encontrar o tesouro.

2. **Desenvolvimento do Programa:**

- Utilizar a linguagem Portugol no Portugol Web Studio.
- Criar variáveis para representar a posição do explorador no labirinto e outras informações relevantes.
    a. Exemplo: definir o labirinto como uma matriz de 5 x 5 contendo = 25 espaços.
    b. Definir o local do tesouro em um desses espaços. Labirinto[4][4] = tesouro.

- Implementar um loop "Enquanto-Faça" para controlar a movimentação do explorador até encontrar o tesouro.

- Definir uma posição iniciar para o explorador no início do caminho, ou seja, da matriz:
    a. posX = 0
    b. posY = 0

```
// Enquanto não encontrar o tesouro
enquanto (labirinto[posX][posY] != tesouro)
{
    escreva("Explorador está na posição: (", posX, ", ", posY, ") ")
```

Definir uma posição iniciar

- Criar condições para determinar os movimentos possíveis do explorador (para frente, para trás, para a esquerda, para a direita) e verificar se ele encontrou o tesouro.
    a. Baseado na posição da matriz 5 x 5 (labirinto), definir a próxima posição do jogador:

	0	1	2	3	4
0	Explor.				
1					
2					
3					
4					Tesouro

    b. Como sugestão você pode definir o tesouro em um local específico da (matriz / labirinto), e fazer o explorador ir caminhando para encontrar o tesouro.
    c. Como sugestão, os valores de cada endereço do labirinto podem ser:
        - 2 = tesouro
        - 1 = parede normal
        - 0 = caminho livre
        - Outro valor para punir o usuário e fazer ele voltar ao início do labirinto.
        - O importante é criar obstáculos ao explorador

3. Geração Aleatória do Labirinto:

- Implementar um algoritmo para gerar aleatoriamente o layout do labirinto ou gerá-lo da mesma forma iniciando o jogador na posição [0][0].
- Garantir que o tesouro esteja escondido em um local diferente a cada vez que o programa iniciar.

4. Testes e Correções:

- Testar o programa com diferentes configurações de labirinto para garantir que o explorador encontre o tesouro corretamente.
- Corrigir quaisquer bugs ou erros de lógica que possam surgir durante os testes.

5. Documentação e Comentários:

- Documentar o código fonte, explicando a lógica por trás do movimento do explorador e as condições para encontrar o tesouro.

- Incluir comentários explicativos para facilitar a compreensão e manutenção do código.

  Exemplo de comentários para a função que inicia o labirinto: // significa linha de comentário

```
funcao inicializarLabirinto()
{
    inteiro i, j

    // Inicializa o labirinto com paredes (1) e caminhos livres (0)
    para (i = 0; i < 5; i++)
    {
        para (j = 0; j < 5; j++)
        {
            labirinto[i][j] = parede
        }
    }

    // Define alguns caminhos livres
    labirinto[0][0] = caminhoLivre
    labirinto[0][1] = caminhoLivre
    labirinto[1][1] = caminhoLivre
```

6. **Execução do Programa:**

- Apresentar o programa, demonstrando como funciona a exploração do labirinto.

```
Explorador está na posição: (0, 0) Explorador está na posição: (0, 1) Explorador está na posição: (1, 1) Explorador está na posição: (2, 1) Explorador está na posição: (2, 2)
Explorador está na posição: (3, 2) Explorador está na posição: (4, 2) Explorador está na posição: (4, 3) Tesouro encontrado na posição: (4, 4)
Programa finalizado. Tempo de execução: 51 milissegundos
```

Checklist:

- Analisar o cenário para criação do código-fonte do programa;
- Gerar aleatoriamente o layout do labirinto em cada execução do programa.
- Garantir que o tesouro esteja escondido em um local diferente a cada vez que o programa iniciar.
- Desenvolver o algoritmo em “Portugol” na ferramenta “Portugol WebStudio”;
- Executar testes e possíveis correções;
- Apresentar o programa e verificar seu correto funcionamento.

RESULTADOS (obrigatório – aparecer para todos)

Resultados de Aprendizagem:

O objetivo da atividade é a aplicação do comando “Enquanto-Faça” para guiar o explorador pelo labirinto até encontrar o tesouro, evitando armadilhas e obstáculos ao longo do caminho.

A atividade proporciona uma oportunidade valiosa para o desenvolvimento de habilidades de resolução de problemas, pensamento crítico e raciocínio lógico.

ESTUDANTE, VOCÊ DEVERÁ ENTREGAR (não obrigatório – aparecer para todos)

Descrição orientativa sobre a entregada da comprovação da aula prática:
O estudante deve entregar um arquivo em PDF contendo toda a codificação do exercício documentada, ou seja, para cada trecho do código-fonte, o estudante deve anexar um texto explicando o que acontece quando o trecho do código é executado.

REFERÊNCIAS BIBLIOGRÁFICAS (não obrigatório – aparecer para todos)

Descrição (em abnt) das referências utilizadas

CORMEN, Thomas. Algoritmos – Teoria e Prática. 3.ed. Rio de Janeiro: LTC, 2022.

MENÉNDEZ, Andrés. Simplificando algoritmos. 1. ed. - Rio de Janeiro: LTC, 2023.

SILVA, Flávio Soares Corrêa D.; FINGER, Marcelo; MELO, Ana Cristina Vieira. Lógica para computação. 2. ed. São Paulo: Cengage Learning, 2017.

NOME DA DISCIPLINA: ALGORITMOS E LÓGICA DE PROGRAMAÇÃO

Unidade: **U3**_CONCEITOS DE PROGRAMAÇÃO

Aula: **A3**_ESTRUTURAS CONDICIONAIS EM LINGUAGEM C

Tempo previsto de execução de aula prática: 2h (CAMPO OBRIGATÓRIO – NÃO APARECER EM NENHUM RAP)

OBJETIVOS (campo obrigatório – exibição para todos)

Definição dos objetivos da aula prática:

- Interpretação dos requisitos e desenvolvimento correto do algoritmo abrangendo todas as possibilidades distintas na execução do programa.
- Aplicar na prática a estrutura condicional: (“IF”) no desenvolvimento de algoritmos computacionais na Linguagem de Programação “C”.

INFRAESTRUTURA (OBRIGATÓRIO SE HOUVER – EXIBIÇÃO DOCENTE/TUTOR)

Instalações – Materiais de consumo – Equipamentos:

Laboratório de Informática

Materiais de consumo:

NSA

Equipamentos:

- **Computador com acesso à internet, e com o mínimo de 4 GB de Memória RAM.**
- **Instalação do Visual Studio Code + extensões:**
  - C/C++ for Visual Studio Code;
  - C/C++ Compile Run extension.

SOLUÇÃO DIGITAL (OBRIGATÓRIO SE HOUVER - APARECER PARA TODOS)

Visual Studio Code (VS Code):

O VS Code é um editor de código-fonte desenvolvido pela Microsoft para Windows, Linux e macOS. Conta com a instalação de extensões para programação em diversas linguagens de programação.

Tipo de licença: Código-fonte: Licença MIT; Binários: Freeware

Link: https://code.visualstudio.com/download

EQUIPAMENTO DE PROTEÇÃO INDIVIDUAL (EPI) (CAMPO OBRIGATÓRIO – APARECER PARA TODOS)

NSA

PROCEDIMENTOS PRÁTICOS (OBRIGATÓRIO – TODOS)

Procedimento/Atividade nº 1 (Virtual)

Atividade proposta:

Você é um desenvolvedor de software em uma instituição financeira responsável por criar um sistema de aprovação de empréstimos. O seu objetivo é desenvolver um programa que analise as informações financeiras fornecidas pelos clientes e determine se o empréstimo pode ser aprovado com base em critérios pré-estabelecidos.

Descrição do Problema:

O sistema de aprovação de empréstimos precisa considerar diversos critérios para determinar se um cliente é elegível para receber um empréstimo. Os critérios são:

- **Relação entre renda e valor do empréstimo:** O valor do empréstimo não pode exceder 30% da renda mensal do cliente.
- **Histórico de crédito**: Clientes com um histórico de crédito ruim podem ter sua solicitação de empréstimo rejeitada.
- **Outros fatores**: Outros fatores, como a estabilidade no emprego do cliente e o valor da entrada, também podem influenciar na decisão de aprovação do empréstimo.

Procedimentos para a realização da atividade:

Faça uma análise detalhada de todas as ações que o programa deve executar, bem como, as estruturas algorítmicas necessárias para o desenvolvimento do código. Aplique os fundamentos da lógica de programação e algoritmos na criação de uma solução para o cenário apresentado utilizando a Linguagem de Programação "C" tendo como apoio o editor de código "**VS Code**".

A seguir, os procedimentos que devem ser realizados:

- **Desenvolver o programa:** Utilizando a linguagem de Programação C no VS Code, você precisa criar um programa que solicite as informações financeiras do cliente, como renda mensal e valor do empréstimo desejado e outras informações que julgue necessário dependendo do raciocínio lógico.

  Solicite do usuário os dados da renda mensal, valor do empréstimo, histórico de crédito, estabilidade no emprego e o valor da entrada. Lembre-se que todas essas informações fazem parte dos critérios para liberação do crédito.

  Abaixo uma sugestão, note que algumas variáveis são de tipos diferentes. Isso também deve ser tratado na construção do algoritmo.

```
printf("Digite sua renda mensal.....................................: ");
scanf("%f", &rendaMensal);

printf("Digite o valor do empréstimo................................: ");
scanf("%f", &valorEmprestimo);

printf("Digite seu histórico de crédito (B=Bom, R=Ruim).............: ");
scanf(" %c", &historicoCredito);

printf("Digite a sua estabilidade no emprego (E=Estável, I=Instável.: )");
scanf(" %c", &estabilidadeEmprego);

printf("Digite o valor da entrada...................................: ");
scanf("%d", &valorEntrada);
```

- **Implementar a lógica de aprovação:** Você deve implementar a lógica necessária para analisar as informações fornecidas pelo cliente e determinar se o empréstimo pode ser aprovado com base nos critérios estabelecidos. Critérios 1, 2 e 3 no item: **Atividade Proposta**.

  Utilize uma estrutura de seleção para construir essa parte do programa. Você pode utilizar os comandos "IF" e ou o comando "switch case", dependendo da elaboração do algoritmo.

  Exemplo de um dos critérios:

  If (Valor do Emprestimo <= 30% da renda mensal) – critério para conceder o empréstimo.

- **Exibir o resultado:** Após a análise, o programa deve informar ao cliente se o empréstimo foi aprovado ou rejeitado, juntamente com detalhes adicionais, se necessário.

Neste ponto você pode exibir uma mensagem quando todos os critérios são cumpridos:

```
Parabéns! Seu empréstimo foi aprovado.
Fim do Programa!
```

E uma outra mensagem explicando o porque do crédito ter sido negado.

- **Testar e depurar:** Teste o programa com diferentes conjuntos de dados para garantir que a análise de aprovação de empréstimos esteja correta. Corrija eventuais falhas ou erros de lógica que possam surgir durante os testes. A seguir um exemplo de teste quando todos os critérios são cumpridos e o crédito será liberado ao cliente.

```
 ** Empréstimo Financeiro **
Digite sua renda mensal.....................................: 7000
Digite o valor do empréstimo................................: 2000
Digite seu histórico de crédito (B=Bom, R=Ruim).............: B
Digite a sua estabilidade no emprego (E=Estável, I=Instável.: )E
Digite o valor da entrada...................................: 600
Parabéns! Seu empréstimo foi aprovado.
Fim do Programa!
```

- **Documentar o código:** Documente o código fonte do programa, explicando a lógica por trás da análise de aprovação de empréstimos. Inclua comentários explicativos para facilitar a compreensão e manutenção do código. Pense sempre que em uma empresa, outros profissionais poderão dar manutenção na codificação realizada anteriormente por você.

  As linhas 11 e 15 possuem comentários do código-fonte

```
11      //comando para limpar a tela
12      system("clear");
13      printf(" ** Empréstimo Financeiro **\n");
14
15      //leitura das informações necessárias ao programa
16      printf("Digite sua renda mensal.....................................: ");
17      scanf("%f", &rendaMensal);
18
```

- **Execução do Programa:** Apresentar o programa, demonstrando como funciona a logística de análise de crédito.

  Exemplo do programa sendo executado:

```
PROBLEMAS 1   SAÍDA   CONSOLE DE DEPURAÇÃO   TERMINAL   PORTAS

 ** Empréstimo Financeiro **
Digite sua renda mensal.....................................: 3500
Digite o valor do empréstimo................................: 950
Digite seu histórico de crédito (B=Bom, R=Ruim).............:
```

Munido de todas essas informações, coloque suas habilidades de programação em prática e crie um programa que ajude a instituição financeira a tomar decisões assertivas na concessão de crédito.

Checklist:

- Analisar o cenário para criação do código-fonte do programa;
- Instalar o VS Code em seu dispositivo. Baixe a IDE escolhendo o sistema operacional por meio do link: https://code.visualstudio.com/download
- Desenvolver o algoritmo na Linguagem de Programação C;
- Considerar todos os critérios solicitados para construção do algoritmo.
- Executar testes e possíveis correções;
- Apresentar o programa e verificar seu correto funcionamento.

RESULTADOS (obrigatório – aparecer para todos)

Resultados de Aprendizagem:

O objetivo da atividade é aplicar os conceitos de comandos de seleção "IF", podendo se necessário, aplicar comandos de "laço de repetições" para analisar as informações financeiras fornecidas pelo usuário, além de determinar se o empréstimo pode ou não ser aprovado. Diante do contexto, ser capaz de aplicar os conceitos de lógica de programação e raciocínio lógico de forma prática em um contexto do mundo real.

ESTUDANTE, VOCÊ DEVERÁ ENTREGAR (não obrigatório – aparecer para todos)

Descrição orientativa sobre a entregada da comprovação da aula prática:

O estudante deve entregar um arquivo em PDF contendo toda a codificação do exercício documentada, ou seja, para cada trecho do código-fonte, o estudante deve anexar um texto explicando o que acontece quando o trecho do código é executado.

REFERÊNCIAS BIBLIOGRÁFICAS (não obrigatório – aparecer para todos)

Descrição (em abnt) das referências utilizadas

CORMEN, Thomas. Algoritmos – Teoria e Prática. 3.ed. Rio de Janeiro: LTC, 2022.

MENÉNDEZ, Andrés. Simplificando algoritmos. 1. ed. - Rio de Janeiro: LTC, 2023.

SILVA, Flávio Soares Corrêa D.; FINGER, Marcelo; MELO, Ana Cristina Vieira. Lógica para computação. 2. ed. São Paulo: Cengage Learning, 2017.

NOME DA DISCIPLINA: ALGORITMOS E LÓGICA DE PROGRAMAÇÃO

Unidade: **U4**_APLICAÇÕES DE PROGRAMAÇÃO

Aula: **A4**_REGISTROS E ARQUIVOS

Tempo previsto de execução de aula prática: 2h (CAMPO OBRIGATÓRIO – NÃO APARECER EM NENHUM RAP)

OBJETIVOS (campo obrigatório – exibição para todos)

Definição dos objetivos da aula prática:

- Interpretação dos requisitos e desenvolvimento correto do algoritmo abrangendo todas as possibilidades distintas na execução do programa.
- Aplicar conceitos de manipulação de “arquivos”, além da criação e acesso a registros “Structs” na Linguagem de Programação “C”.

INFRAESTRUTURA (OBRIGATÓRIO SE HOUVER – EXIBIÇÃO DOCENTE/TUTOR)

Instalações – Materiais de consumo – Equipamentos:

Laboratório de Informática

Materiais de consumo:

NSA

Equipamentos:

- **Computador com acesso à internet, e com o mínimo de 4 GB de Memória RAM.**
- **Instalação do Visual Studio Code + extensões:**
  - C/C++ for Visual Studio Code;
  - C/C++ Compile Run extension.

SOLUÇÃO DIGITAL (OBRIGATÓRIO SE HOUVER - APARECER PARA TODOS)

Visual Studio Code (VS Code):

O VS Code é um editor de código-fonte desenvolvido pela Microsoft para Windows, Linux e macOS. Conta com a instalação de extensões para programação em diversas linguagens de programação.

Tipo de licença: Código-fonte: Licença MIT; Binários: Freeware

Link: https://code.visualstudio.com/download

EQUIPAMENTO DE PROTEÇÃO INDIVIDUAL (EPI) (CAMPO OBRIGATÓRIO – APARECER PARA TODOS)

NSA

PROCEDIMENTOS PRÁTICOS (OBRIGATÓRIO – TODOS)

Procedimento/Atividade nº 1 (Virtual)

Atividade proposta:

Você foi designado pela sua instituição de ensino para desenvolvedor um programa de classificação dos alunos. O programa deverá solicitar alguns dados, inclusive as médias das disciplinas de cada aluno para que ele possa fazer o cálculo da média do aluno. Deverá solicitar também o nome do aluno.

Descrição do Problema:

Para realização da atividade você deve utilizar os recursos de Structs e Arquivos. A Struct armazenará os dados do aluno como: Nome, média da primeira disciplina, média da segunda disciplina, média geral por aluno e sua classificação.

Um arquivo do tipo ".txt" deve ser utilizado para armazenar as informações no final da execução do programa. O layout do arquivo deve conter:

- **Nome do Aluno:**
- **Média Disciplina 1:**
- **Média Disciplina 2:**
- **Média Geral:**
- **Classificação:**

Procedimentos para a realização da atividade:

Faça uma análise detalhada de todas as ações que o programa deve executar, bem como, as estruturas algorítmicas necessárias para o desenvolvimento do código. Aplique os fundamentos da lógica de programação e algoritmos na criação de uma solução para o cenário apresentado utilizando a Linguagem de Programação "C" tendo como apoio o editor de código "**VS Code**".

A seguir, os procedimentos que devem ser realizados:

1. **Passo 1: Definição das estruturas de dados:**
    a. Criar a estrutura(Struct) "Aluno" com os campos necessários (nome, médias das disciplinas, média geral, classificação).
    Você pode optar pelo nome da Struct ser "Aluno", ou outro nome qualquer desde que você respeite as regras para nomenclatura de variáveis: evitando acentuação, sem

utilização de espaços em branco, não podendo utilizar palavras reservadas à linguagem, exemplo: “**void**” e não começar com caractere numérico.

Aqui optei pelo nome “**Aluno**”.

```
typedef struct {
    char nome[MAX_NOME];
    float media_disciplina1;
    float media_disciplina2;
    float media_geral;
    int classificacao;
} Aluno;
```

2. **Passo 2: Implementação das funções de cálculo:**
    a. Implementar uma função para calcular a média geral de um aluno com base nas médias das disciplinas.

    Um exemplo da função para calcular a média de um aluno. Passa-se por parâmetro as médias das disciplinas e depois retorna o cálculo da média.

```
float calcularMediaGeral(float media_disciplina1, float media_disciplina2) {
    return (media_disciplina1 + media_disciplina2) / 2;
}
```

    b. Implementar uma função de comparação para ser utilizada na ordenação da turma.

    Essa função (**compararMedias**) é chamada dentro de outra função que salva o arquivo .txt que tem o objetivo de ordenar o aluno dentro do arquivo de acordo com sua classificação por nota. Exemplo:

```
int compararMedias(const void *a, const void *b) {
    const Aluno *alunoA = (const Aluno *)a;
    const Aluno *alunoB = (const Aluno *)b;
    if (alunoA->media_geral < alunoB->media_geral) {…
    } else if (alunoA->media_geral > alunoB->media_geral) {
        return -1;
    } else {
        return 0;
    }
}
```

3. **Passo 3: Entrada de dados:**
    a. Pedir ao usuário que informe quantos alunos serão registrados.

    Isso é feito dentro da função main(). Uma sugestão de como resolver está logo a seguir:

```
69  int main() {
70      system("clear");
71      int num_alunos;
72      printf("Quantos alunos deseja registrar? (maximo %d): ", MAX_ALUNOS);
73      scanf("%d", &num_alunos);
```

b. Usar um loop para solicitar o nome, médias das disciplinas 1 e 2 de cada aluno e armazenar esses dados na estrutura Aluno.

Dentro de um laço qualquer na linguagem de programação C, solicite do usuário esses dados.

4. **Passo 4: Cálculo das médias e classificação:**
   a. Calcular a média geral de cada aluno utilizando uma função.

      A função que faz esse cálculo é a **calcularMediaGeral()** já exibida no item 2.

   b. Ordenar a turma em ordem decrescente de acordo com as médias dos alunos.

      Você deve comparar as médias gerais de cada aluno e colocar a classificação de cada um dentro do arquivo .txt ao lado dos dados do aluno.

   c. Atribuir as classificações aos alunos com base na ordem da turma.

      Inserir dentro do arquivo um campo para a classificação do aluno. Utilizar o campo classificação da Struct.

```c
typedef struct {
    char nome[MAX_NOME];
    float media_disciplina1;
    float media_disciplina2;
    float media_geral;
    int classificacao;
} Aluno;
```

5. **Passo 5: Saída de dados:**
   a. Para cada aluno, escrever no arquivo seu nome, médias das disciplinas, média geral e classificação.

   b. Calcular e escrever no arquivo a média geral da turma.

      Uma sugestão para a saída do arquivo texto logo a seguir:

notas_alunos.txt

```
Nome: C
Media Disciplina 1: 70.00
Media Disciplina 2: 80.00
Media Geral: 75.00
Ordem: 1

Nome: B
Media Disciplina 1: 60.00
Media Disciplina 2: 70.00
Media Geral: 65.00
Ordem: 2
```

6. **Passo 6: Finalização:**
   a. Fechar o arquivo.

      Utilizar a classe “FILE” da linguagem C para manipular dados no arquivo.

7. **Execução do Programa:**

   Apresentar o programa, demonstrando o seu funcionamento.

   Exemplo do programa sendo executado:

```
PROBLEMAS 1   SAÍDA   CONSOLE DE DEPURAÇÃO   TERMINAL

Quantos alunos deseja registrar? (maximo 50): 2

Aluno 1:
Nome: Anderson
Media Disciplina 1: 100
Media Disciplina 2: 90

Aluno 2:
Nome: Renata
Media Disciplina 1: 85
Media Disciplina 2: 48

Dados dos alunos salvos no arquivo 'notas_alunos.txt'.
```

Na sequência, o arquivo .txt gerado pelo programa.

```
notas_alunos.txt
1    Nome: Anderson
2    Media Disciplina 1: 100.00
3    Media Disciplina 2: 90.00
4    Media Geral: 95.00
5    Ordem: 1
6
7    Nome: Renata
8    Media Disciplina 1: 85.00
9    Media Disciplina 2: 48.00
10   Media Geral: 66.50
11   Ordem: 2
12
13
14   Media geral da turma: 161.50
```

Munido de todas essas informações, coloque suas habilidades de programação em prática e crie um programa que atenda a todos os requisitos solicitados pela sua instituição de ensino.

Checklist:

- Analisar o cenário para criação do código-fonte do programa;
- Instalar o VS Code em seu dispositivo. Baixe a IDE escolhendo o sistema operacional por meio do link: https://code.visualstudio.com/download
- Desenvolver o algoritmo na Linguagem de Programação C;
- Considerar todos os critérios solicitados para construção do algoritmo.
- Executar testes e possíveis correções;
- Apresentar o programa e verificar seu correto funcionamento.

RESULTADOS (obrigatório – aparecer para todos)

Resultados de Aprendizagem:

O objetivo da atividade é aplicar conceitos das “estruturas de seleção”, “laços de repetições”, “funções”, “Structs” e “arquivos”, para coletar os dados, realizar os cálculos e fazer a classificação dos alunos. Diante do cenário apresentado, ser capaz de aplicar os conceitos de lógica de programação desenvolvendo o raciocínio lógico de forma prática em um contexto do mundo real.

ESTUDANTE, VOCÊ DEVERÁ ENTREGAR (não obrigatório – aparecer para todos)

Descrição orientativa sobre a entregada da comprovação da aula prática:

O estudante deve entregar um arquivo em PDF contendo toda a codificação do exercício documentada, ou seja, para cada trecho do código-fonte, o estudante deve anexar um texto explicando o que acontece quando o trecho do código é executado.

REFERÊNCIAS BIBLIOGRÁFICAS (não obrigatório – aparecer para todos)

Descrição (em abnt) das referências utilizadas

CORMEN, Thomas. Algoritmos – Teoria e Prática. 3.ed. Rio de Janeiro: LTC, 2022.

MENÉNDEZ, Andrés. Simplificando algoritmos. 1. ed. - Rio de Janeiro: LTC, 2023.

SILVA, Flávio Soares Corrêa D.; FINGER, Marcelo; MELO, Ana Cristina Vieira. Lógica para computação. 2. ed. São Paulo: Cengage Learning, 2017.