

Instruções de Entrega - Unidade 2

Disciplina: Introdução a Técnicas de Programação (ITP)

Período: 01/10/2025 a 03/11/2025

Data limite: 03/11/2025 às 23:59

Plataforma de entrega: SIGAA







Sumário

1. [Visão Geral](#)
 2. [Componentes da Avaliação](#)
 3. [Estrutura do Repositório](#)
 4. [Projeto Individual](#)
 5. [Listas de Exercícios](#)
 6. [Relatório Técnico](#)
 7. [Vídeo de Demonstração](#)
 8. [Critérios de Avaliação](#)
 9. [Entrega](#)
 10. [Perguntas Frequentes](#)
-

Visão Geral

A Unidade 2 consolida e expande os conhecimentos adquiridos na U1, introduzindo novos conceitos fundamentais de programação em C. Esta unidade foca em estruturas de dados mais complexas e gerenciamento de memória.

Conteúdos que devem estar presentes no projeto da U2:

-  Todos os tópicos da U1 (variáveis, tipos, operadores, vetores, condicionais, repetições, funções)
 -  **Strings** e manipulação de texto
 -  **Estruturas de repetição aninhadas** (loops dentro de loops)
 -  **Matrizes** (arrays bidimensionais)
 -  **Ponteiros** básicos
 -  **Alocação dinâmica de memória** básica (malloc, free)
-

Componentes da Avaliação

A nota da Unidade 2 será calculada da seguinte forma:

$$\text{Nota da U2} = (\text{Projeto} \times 0,8) + (\text{Média das Listas} \times 0,2)$$

Composição da nota:





- **Projeto Individual: 80%**
 - Código-fonte (40%)
 - Relatório técnico (20%)
 - Vídeo de demonstração (20%)
- **Listas de Exercícios: 20%**
 - Semana 7: Strings
 - Semana 8: Estruturas de Repetição Aninhadas
 - Semana 9: Matrizes
 - Semana 10: Ponteiros e Alocação Dinâmica 1

Estrutura do Repositório

Seu repositório deve seguir a estrutura abaixo:

```
nome-do-aluno-itp-2025-2/
├── projeto/
│   ├── src/           # Código-fonte (.c)
│   ├── include/       # Arquivos de cabeçalho (.h)
│   ├── Makefile        # Script de compilação (recomendado)
│   └── README.md       # Instruções de compilação e execução
├── listas/
│   ├── semana2-variaveis/ # Lista da U1
│   ├── semana3-condicionais/ # Lista da U1
│   ├── semana4-repeticoes-a/ # Lista da U1
│   ├── semana4-repeticoes-b/ # Lista da U1
│   ├── semana5-funcoes/     # Lista da U1
│   ├── semana6-vetores/     # Lista da U1
│   ├── semana7-strings/     # ✨ Nova - Lista da U2
│   ├── semana8-repeticoes-aninhadas/ # ✨ Nova - Lista da U2
│   ├── semana9-matrizes/     # ✨ Nova - Lista da U2
│   └── semana10-ponteiros-alocacao/ # ✨ Nova - Lista da U2
├── relatorios/
│   ├── relatorio-u1.pdf      # Relatório da U1
│   └── relatorio-u2.pdf      # ✨ Novo - Relatório da U2
├── videos/
│   ├── demonstracao-u1.md    # Link do vídeo da U1
│   └── demonstracao-u2.md    # ✨ Novo - Link do vídeo da U2
└── README.md                 # Descrição geral do repositório
```

Observações importantes:

-  **Mantenha as entregas da U1** no repositório
-  O projeto deve ser **evolutivo**: expanda o projeto da U1 ou desenvolva um novo
-  Cada pasta de lista deve conter os arquivos **.c** das soluções
-  Os arquivos **.md** na pasta **videos/** devem conter apenas o link do vídeo

Projeto Individual

Requisitos Obrigatórios:



1. Conteúdos da U1 (mínimo 70% dos tópicos):






- Variáveis com tipos bem-definidos
- Operações aritméticas, lógicas e relacionais
- Vetores (arrays unidimensionais)
- Comandos condicionais (if, else, switch)
- Estruturas de repetição (for, while, do-while)
- Funções (declaração, definição, chamada)

2. Novos conteúdos da U2 (mínimo 70% dos tópicos):

- ✨ **Strings:**
 - Declaração e inicialização
 - Funções da `string.h` (strlen, strcpy, strcmp, strcat, etc.)
 - Manipulação caractere a caractere
- ✨ **Estruturas de Repetição Aninhadas:**
 - Loops dentro de loops
 - Aplicação em problemas matriciais
 - Padrões e figuras com asteriscos
- ✨ **Matrizes:**
 - Declaração e inicialização
 - Operações matriciais (soma, multiplicação, transposição)
 - Percorrimento bidimensional
- ✨ **Ponteiros Básicos:**
 - Declaração e uso de ponteiros
 - Operador de endereço (&) e desreferenciação (*)
 - Passagem por referência em funções
 - Ponteiros e arrays
- ✨ **Alocação Dinâmica Básica:**
 - Uso de `malloc()` e `free()`
 - Alocação de vetores dinâmicos
 - Gerenciamento básico de memória
 - Verificação de alocação bem-sucedida

Características Técnicas:

-  Desenvolvido em **C puro** (padrão C99 ou superior)
-  Interface de **linha de comando (CLI)**

-  **Código original** (não copiado de repositórios públicos)
-  **Complexidade média ou alta**
-  **Código bem comentado** e indentado
-  **Tratamento de erros** adequado
-  **Sem memory leaks** (toda memória alocada deve ser liberada)

Sugestões de Evolução do Projeto da U1:

Se você optou por continuar o projeto da U1, aqui estão algumas ideias de como incorporar os novos conteúdos:

1. Sistema de Gerenciamento de Biblioteca:

- Adicione busca de livros por string (título/autor)
- Use matrizes para relatórios tabulares
- Implemente alocação dinâmica para lista de livros

2. Jogo da Velha:

- Expanda para um tabuleiro maior (4x4 ou 5x5)
- Adicione histórico de jogadas com strings
- Use alocação dinâmica para diferentes tamanhos de tabuleiro

3. Calculadora Científica:

- Adicione manipulação de expressões com strings
- Implemente matrizes para operações matriciais
- Use alocação dinâmica para histórico ilimitado

4. Gerenciador de Tarefas:

- Adicione categorização com strings
- Use matrizes para relatórios mensais
- Implemente lista dinâmica de tarefas

Novos Projetos Sugeridos para U2:

- Sistema de Criptografia de Textos
- Editor de Texto Simples em Memória
- Jogo de Palavras Cruzadas
- Sistema de Análise de Textos (contagem de palavras, frequência)
- Simulador de Planilha Eletrônica Simples
- Sistema de Processamento de Imagens em ASCII Art
- Jogo de Sudoku
- Gerenciador de Senhas com Criptografia Simples
- Sistema de Busca de Padrões em Textos



Listas de Exercícios

Listas da U2 (obrigatórias):

Semana 7: Strings

- **Pasta:** `listas/semana7-strings/`
- **Conteúdo:** Manipulação de strings, funções da biblioteca padrão

Semana 8: Estruturas de Repetição Aninhadas

- **Pasta:** `listas/semana8-repeticoes-aninhadas/`
- **Conteúdo:** Loops aninhados, padrões, figuras

Semana 9: Matrizes

- **Pasta:** `listas/semana9-matrizes/`
- **Conteúdo:** Operações com matrizes bidimensionais

Semana 10: Ponteiros e Alocação Dinâmica 1

- **Pasta:** `listas/semana10-ponteiros-alocacao/`
- **Conteúdo:** Ponteiros básicos, malloc, free

Formato de Entrega das Listas:

Cada problema deve estar em um arquivo separado:

```
semana7-strings/  
├─ problema1.c  
├─ problema2.c  
├─ problema3.c  
└─ README.md      # (opcional) Observações sobre as soluções
```

Critérios de Avaliação das Listas:

- ☒ Solução correta (60%)
- ☒ Qualidade do código (20%)
- ☒ Eficiência (10%)
- ☒ Comentários e documentação (10%)



Relatório Técnico

Especificações:

- **Formato:** PDF
- **Tamanho:** 3 a 5 páginas
- **Localização:** `relatorios/relatorio-u2.pdf`
- **Fonte:** Times New Roman ou Arial, tamanho 12
- **Espaçamento:** 1,5 linhas
- **Margens:** 2,5 cm

Estrutura Obrigatória:

1. Introdução (0,5 página)

- Contexto do projeto
- Objetivos da U2
- Breve descrição das funcionalidades implementadas

2. Metodologia (0,5-1 página)

- Ferramentas utilizadas (compilador, IDE, etc.)
- Abordagem de desenvolvimento
- Organização do código

3. Análise do Código (1,5-2 páginas)

Foco nos novos conteúdos da U2:

- **Strings:**
 - Como foram utilizadas no projeto?
 - Quais funções de manipulação foram implementadas?
 - Exemplos de uso
- **Estruturas de Repetição Aninhadas:**
 - Onde foram aplicadas?
 - Qual a complexidade dos algoritmos?
 - Casos de uso
- **Matrizes:**
 - Como foram implementadas?
 - Quais operações são suportadas?
 - Estratégias de percorrimento
- **Ponteiros:**
 - Como foram utilizados?
 - Passagem por referência vs. valor
 - Exemplos práticos
- **Alocação Dinâmica:**
 - Onde foi necessária?
 - Como é gerenciada a memória?
 - Tratamento de falhas de alocação
 - Estratégia para evitar memory leaks

4. Dificuldades e Soluções (0,5-1 página)

- Principais desafios técnicos
- Como foram superados
- Aprendizados importantes
- **Especialmente:** Dificuldades com ponteiros e gerenciamento de memória

5. Conclusão (0,5 página)

- Síntese dos aprendizados
- Reflexão sobre a evolução desde a U1
- Possíveis melhorias futuras
- Próximos passos para a U3

Perguntas Orientadoras (devem ser respondidas):

Gerais:

1. Quais conceitos da U2 foram aplicados no projeto?
2. Como a organização do código facilita a manutenção?
3. Quais foram os principais desafios técnicos enfrentados?

Específicas da U2: 4. Como foram implementadas as estruturas de dados complexas (matrizes)? 5. Qual a estratégia para gerenciamento de memória? 6. Como você garante que não há vazamentos de memória? 7. Quais vantagens a alocação dinâmica trouxe para seu projeto? 8. Como os ponteiros foram utilizados para melhorar a eficiência?

Critérios de Avaliação do Relatório:

- Clareza e coerência (10%)
- Profundidade técnica (10%)
- Resposta às perguntas orientadoras (10%)
- Formatação e organização (5%)
- Análise crítica e reflexão (5%)

Vídeo de Demonstração

Especificações:

- **Duração:** 5 a 8 minutos
- **Formato:** Link para YouTube, Google Drive ou similar
- **Localização:** [videos/demonstracao-u2.md](#)

Conteúdo do Arquivo [demonstracao-u2.md](#):

Vídeo de Demonstração – Unidade 2

****Aluno:**** Seu Nome Completo

****Matrícula:**** 123456789

****Projeto:**** Nome do Projeto

****Link do Vídeo:**** [URL_DO_VIDEO]

****Duração:**** X minutos e Y segundos

Conteúdos Demonstrados:

- [] Funcionalidades envolvendo strings
- [] Uso de estruturas de repetição aninhadas
- [] Operações com matrizes
- [] Aplicação de ponteiros
- [] Gerenciamento de memória dinâmica
- [] Outros: _____

O que deve ser mostrado no vídeo:

1. Introdução (1 minuto)

- Apresentação pessoal
- Nome e objetivo do projeto
- Novidades implementadas na U2

2. Demonstração das Funcionalidades (3-4 minutos)

Priorize demonstrar os novos conteúdos da U2:

- ✨ **Funcionalidades com strings:**
 - Busca, comparação, concatenação
 - Validações de entrada
- ✨ **Uso de matrizes:**
 - Visualização de dados bidimensionais
 - Operações matriciais
- ✨ **Demonstração de ponteiros:**
 - Como afetam o comportamento do programa
 - Passagem por referência
- ✨ **Alocação dinâmica em ação:**
 - Criação dinâmica de estruturas
 - Liberação de memória
 - Mostrar que não há leaks (usar Valgrind, se possível)

3. Análise do Código (2-3 minutos)

Mostre trechos relevantes do código explicando:

- Implementação de funções que usam strings
- Estruturas de repetição aninhadas
- Declaração e uso de matrizes
- Uso de ponteiros e alocação dinâmica
- Tratamento de erros de alocação
- Liberação de memória

4. Conclusão (0,5-1 minuto)

- Principais aprendizados da U2
- Próximos passos

Dicas para Gravação:

- Ensaie antes de gravar para respeitar o tempo
- Edite cortes e erros
- Demonstre casos de erro e validação
- Tenha certeza que o código está legível no vídeo (se necessário aumente o tamanho da fonte do editor)
- ⚠ **Mostre o código compilando sem warnings**

Critérios de Avaliação do Vídeo:

- Demonstração completa do projeto (15%)
- Qualidade da explicação técnica (10%)
- Domínio do conteúdo (10%)
- Clareza e objetividade (5%)
- Respeito ao tempo (5%)

Critérios de Avaliação

Projeto (40% da nota total)

Qualidade e Organização (10%):

- Estrutura de pastas adequada
- Nomenclatura consistente e clara
- Indentação e formatação
- Comentários relevantes
- Uso de constantes e boas práticas

Funcionalidade (15%):

- Programa executa conforme esperado
- Resolve o problema proposto
- Interface de usuário funcional
- Tratamento de entradas inválidas

Aplicação dos Conceitos da U2 (10%):

- Uso adequado de strings
- Implementação de estruturas aninhadas
- Operações com matrizes
- Uso correto de ponteiros
- Gerenciamento adequado de memória
- **Ausência de memory leaks**

Histórico de Commits (5%):

- Commits frequentes (mínimo 10 commits novos na U2)
- Mensagens descritivas
- Evolução gradual do código
- Commits ao longo do período (não tudo no último dia)

Relatório (20% da nota total)

- Clareza e coerência (10%)
- Profundidade técnica (10%)
- Resposta às perguntas orientadoras (10%)
- Formatação adequada (5%)

Vídeo (20% da nota total)

- Demonstração completa (15%)
- Qualidade da explicação (10%)
- Domínio do conteúdo (10%)
- Respeito ao tempo (5%)

Listas de Exercícios (20% da nota total)

- Média aritmética das 4 listas da U2
- Cada lista vale 0 a 10

Entrega

Como Entregar:

1. **Organize seu repositório** conforme a estrutura especificada
2. **Faça commit** de todas as alterações
3. **Acesse o SIGAA** e localize a atividade "Entrega U2"
4. **Envie o link do repositório**
5. **Confirme** que todos os arquivos estão visíveis no GitHub

Checklist de Entrega:

- ☐ Projeto atualizado na pasta **projeto/**
- ☐ Listas da semana 7, 8, 9 e 10 completas

- ☐ Relatório **relatorio-u2.pdf** na pasta **relatorios/**
- ☐ Arquivo **demonstracao-u2.md** com link do vídeo
- ☐ Commits frequentes e descritivos
- ☐ README.md atualizado com instruções
- ☐ Código compila sem erros
- ☐ Código testado e funcionando
- ☐ Memória sendo liberada corretamente
- ☐ Link do repositório enviado no SIGAA

⚠ Atenção:

- **Prazo final:** 03/11/2025 às 23:59
- 📧 **Dúvidas:** Entre em contato comigo o quanto antes, não deixe pra última hora
- 🚫 **Plágio:** Resultará em nota zero
- 🔒 **Repositório:** Deve estar público ou com acesso compartilhado

? Perguntas Frequentes

1. Preciso criar um projeto novo ou posso evoluir o da U1?

Você pode escolher:

- **Opção A:** Evoluir o projeto da U1 adicionando os novos conceitos
- **Opção B:** Criar um projeto completamente novo

Ambas as opções são válidas, desde que os requisitos da U2 sejam atendidos.

2. Como sei se estou usando pelo menos 70% dos conteúdos?

Para a U2, você deve usar pelo menos:

- 3 de 4 novos conteúdos (strings, estruturas aninhadas, matrizes, ponteiros/alocação)
- E continuar usando os conceitos da U1

3. Como posso verificar memory leaks?

Use o Valgrind no Linux:

```
valgrind --leak-check=full ./seu_programa
```

Ou ferramentas similares no Windows/Mac.

4. Posso usar bibliotecas externas?

Apenas as bibliotecas padrão do C:

- **stdio.h, stdlib.h, string.h, math.h, time.h, ctype.h**

Bibliotecas externas (não padrão) **são permitidas** desde que a(o) aluna(o) justifique o motivo.

5. Quantos commits devo fazer?

Recomenda-se:

- Mínimo: 10 commits novos durante a U2
- Ideal: 20-25 commits
- Distribuídos ao longo das 4 semanas

6. O vídeo pode ter mais de 8 minutos?

O ideal é que o tempo seja respeitado e caso ultrapasse, que seja por pouco tempo.

7. Como devem ser os comentários no código?

```
// Comentários de linha única para explicações breves

/*
 * Comentários de bloco para explicações mais longas,
 * como descrições de funções complexas
 */

/**
 * Documentação de funções (estilo Doxygen)
 * @param x Descrição do parâmetro
 * @return Descrição do retorno
 */
```

8. Posso trabalhar em dupla?

Não. O projeto é **estritamente individual**.

9. E se eu não conseguir implementar todos os conteúdos da U2?

O mínimo é 70% dos conteúdos novos (3 de 4). Menos que isso resultará em perda de pontos na avaliação.