



ESTRUTURAS CONDICIONAIS EM LINGUAGEM C

Disciplina: Algoritmos e Lógica de Programação

Subtítulo: Estruturas condicionais em linguagem C

Alex Barroso Paz – 2025

INSTITUIÇÃO: Ampli

DISCIPLINA: Algoritmos e Lógica de Programação

PROFESSOR: Paulo Sergio Ferro Junior

TÍTULO: Estruturas condicionais em linguagem C

ALUNO: Alex Barroso Paz

MATRÍCULA: 2023013724

GIT: <https://github.com/finntroll89/Estruturas-condicionais-em-linguagem-C.git>

DATA: 21 de Junho de 2025

INTRODUÇÃO

Este trabalho apresenta o desenvolvimento de um sistema de aprovação de empréstimos implementado em Linguagem C, utilizando estruturas condicionais para análise de crédito. O sistema foi desenvolvido como parte da disciplina de Algoritmos e Lógica de Programação, demonstrando a aplicação prática de conceitos fundamentais da programação.

O programa simula um sistema bancário real, avaliando quatro critérios essenciais para a aprovação de empréstimos: relação entre renda e valor solicitado, histórico de crédito, estabilidade no emprego e valor da entrada oferecida pelo cliente.

2. OBJETIVOS

2.1 Objetivo Geral

Desenvolver um sistema computacional em Linguagem C que automatize o processo de análise e aprovação de empréstimos bancários utilizando estruturas condicionais.

2.2 Objetivos Específicos

- Implementar estruturas condicionais (`if, else if, else`) para análise de critérios
- Criar um sistema de validação de dados de entrada
- Desenvolver lógica de negócio para análise de crédito
- Aplicar conceitos de programação estruturada
- Produzir relatórios detalhados sobre a decisão de aprovação

3. METODOLOGIA

O desenvolvimento seguiu as seguintes etapas:

1. **Análise de Requisitos:** Definição dos critérios de aprovação de empréstimos
2. **Planejamento:** Estruturação da lógica de programação
3. **Implementação:** Codificação em Linguagem C
4. **Testes:** Validação com diferentes cenários
5. **Documentação:** Explicação detalhada do código

Ferramentas Utilizadas:

- **Editor:** Visual Studio Code
- **Compilador:** GCC (GNU Compiler Collection)
- **Sistema Operacional:** Windows 11 pro
- **Terminal:** PowerShell

4. DESENVOLVIMENTO DO SISTEMA

4.1 Critérios de Aprovação Implementados

O sistema avalia quatro critérios fundamentais:

1. **Relação Renda x Empréstimo:** Valor solicitado não pode exceder 30% da renda mensal
2. **Histórico de Crédito:** Apenas clientes com histórico "Bom" são aprovados
3. **Estabilidade no Emprego:** Somente empregos "Estáveis" são aceitos
4. **Valor da Entrada:** Mínimo de 20% do valor do empréstimo

4.2 Fluxo de Funcionamento

1. **Entrada de Dados:** Sistema coleta informações do cliente
2. **Processamento:** Análise de cada critério individualmente
3. **Validação:** Verificação de dados e tratamento de erros
4. **Resultado:** Decisão final com justificativas detalhadas

5. CÓDIGO FONTE COMPLETO

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>

int main() {
    // Declaração de variáveis
    float rendaMensal, valorEmprestimo, valorEntrada;
    char historicoCredito, estabilidadeEmprego;
    float limiteEmprestimo;
    int aprovado = 1; // 1 = aprovado, 0 = rejeitado

    // Comando para limpar a tela
```

```

system("cls");

// Cabecalho do programa
printf(" ** Emprestimo Financeiro **\n");
printf("=====\\n\\n");

// Leitura das informacoes necessarias ao programa
printf("Digite sua renda mensal.....: ");
scanf("%f", &rendaMensal);

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

// Limpeza do buffer antes de ler caracteres
while(getchar() != '\n');

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

// Limpeza do buffer antes de ler caracteres
while(getchar() != '\n');

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

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

printf("\n=====\\n");
printf("ANALISE DE CREDITO\\n");
printf("=====\\n\\n");

// Calculo do limite de emprestimo (30% da renda mensal)
limiteEmprestimo = rendaMensal * 0.30;

// Criterio 1: Relacao entre renda e valor do emprestimo
if (valorEmprestimo > limiteEmprestimo) {
    printf("[REJEITADO] CRITERIO 1: O valor do emprestimo (R$ %.2f) excede 30%% da

```

```

sua renda mensal (R$ %.2f)\n",
            valorEmprestimo, limiteEmprestimo);
    aprovado = 0;
} else {
    printf("[APROVADO] CRITERIO 1: Valor do emprestimo aprovado (%.2f%% da renda
mensal)\n",
           (valorEmprestimo/rendaMensal)*100);
}

// Criterio 2: Historico de credito
if (historicoCredito == 'R') {
    printf("[REJEITADO] CRITERIO 2: Historico de credito ruim - emprestimo
rejeitado\n");
    aprovado = 0;
} else if (historicoCredito == 'B') {
    printf("[APROVADO] CRITERIO 2: Historico de credito bom - criterio atendido\n");
} else {
    printf("[REJEITADO] CRITERIO 2: Historico de credito invalido - digite B ou
R\n");
    aprovado = 0;
}

// Criterio 3: Estabilidade no emprego
if (estabilidadeEmprego == 'I') {
    printf("[REJEITADO] CRITERIO 3: Emprego instavel - risco aumentado\n");
    aprovado = 0;
} else if (estabilidadeEmprego == 'E') {
    printf("[APROVADO] CRITERIO 3: Emprego estavel - criterio atendido\n");
} else {
    printf("[REJEITADO] CRITERIO 3: Estabilidade no emprego invalida - digite E ou
I\n");
    aprovado = 0;
}

// Criterio 4: Valor da entrada (minimo 20% do valor do emprestimo)
float entradaMinima = valorEmprestimo * 0.20;
if (valorEntrada < entradaMinima) {
    printf("[REJEITADO] CRITERIO 4: Valor da entrada insuficiente (minimo R$
%.2f)\n", entradaMinima);
    aprovado = 0;
}

```

```

} else {
    printf("[APROVADO] CRITERIO 4: Valor da entrada adequado (%.2f%% do
emprestimo)\n",
           (valorEntrada/valorEmprestimo)*100);
}

printf("\n=====\\n");
printf("RESULTADO FINAL\\n");
printf("=====\\n\\n");

// Resultado final da analise
if (aprovado) {
    printf("*** PARABENS! Seu emprestimo foi APROVADO! ***\\n\\n");
    printf("Detalhes da aprovação:\\n");
    printf("- Valor do emprestimo: R$ %.2f\\n", valorEmprestimo);
    printf("- Valor da entrada: R$ %.2f\\n", valorEntrada);
    printf("- Valor a financiar: R$ %.2f\\n", valorEmprestimo - valorEntrada);
    printf("- Comprometimento da renda: %.2f%%\\n",
           (valorEmprestimo/rendaMensal)*100);
} else {
    printf("*** EMPRESTIMO REJEITADO! ***\\n\\n");
    printf("Motivos da rejeição:\\n");
    printf("- Verifique os critérios marcados com [REJEITADO] acima\\n");
    printf("- Ajuste as condições e solicite novamente\\n");
    printf("- Procure orientação financeira se necessário\\n");
}

printf("\n=====\\n");
printf("Fim do Programa!\\n");
printf("=====\\n");

return 0;
}

```

6. EXPLICAÇÃO DETALHADA DO CÓDIGO

6.1 Inclusão de Bibliotecas

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
```

Explicação:

- **stdio.h**: Biblioteca padrão de entrada e saída, fornece funções como `printf()` e `scanf()`
- **stdlib.h**: Biblioteca padrão que inclui funções como `system()` para comandos do sistema
- **ctype.h**: Biblioteca para manipulação de caracteres, fornece a função `toupper()`

6.2 Declaração de Variáveis

```
float rendaMensal, valorEmprestimo, valorEntrada;
char historicoCredito, estabilidadeEmprego;
float limiteEmprestimo;
int aprovado = 1;
```

Explicação:

- **float**: Tipo de dados para números decimais (valores monetários)
- **char**: Tipo de dados para caracteres únicos (B/R, E/I)
- **int aprovado**: Variável de controle (1 = aprovado, 0 = rejeitado)
- **limiteEmprestimo**: Armazena o cálculo de 30% da renda mensal

6.3 Entrada de Dados

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

Explicação:

- **printf()**: Exibe mensagens na tela
- **scanf()**: Lê dados digitados pelo usuário
- **%f**: Especificador de formato para números float
- **&**: Operador de endereço, fornece o endereço da variável

6.4 Limpeza do Buffer

```
while(getchar() != '\n');
```

Explicação:

- **getchar()**: Lê um caractere do buffer
- **while loop**: Consome todos os caracteres até encontrar '\n' (nova linha)
- **Necessário**: Evita problemas na leitura de caracteres após números

6.5 Conversão de Caracteres

```
historicoCredito = toupper(historicoCredito);
```

Explicação:

- **toupper()**: Converte caracteres minúsculos para maiúsculos
- **Vantagem**: Permite que o usuário digite 'b' ou 'B' sem erro

6.6 Estruturas Condicionais

```
if (valorEmprestimo > limiteEmprestimo) {
    printf("[REJEITADO] CRITERIO 1: O valor do emprestimo (R$ %.2f) excede 30% da sua
renda mensal (R$ %.2f)\n",
           valorEmprestimo, limiteEmprestimo);
    aprovado = 0;
} else {
    printf("[APROVADO] CRITERIO 1: Valor do emprestimo aprovado (%.2f%% da renda
mensal)\n",
           (valorEmprestimo/rendaMensal)*100);
}
```

Explicação:

- **if-else**: Estrutura condicional básica
- **%.2f**: Formata números float com 2 casas decimais
- **%%**: Exibe o símbolo % (escape necessário)

- **aprovado = 0:** Marca como rejeitado se critério não for atendido

6.7 Estruturas Condicionais Aninhadas

```
if (historicoCredito == 'R') {
    printf("[REJEITADO] CRITERIO 2: Historico de credito ruim - emprestimo
rejeitado\n");
    aprovado = 0;
} else if (historicoCredito == 'B') {
    printf("[APROVADO] CRITERIO 2: Historico de credito bom - criterio atendido\n");
} else {
    printf("[REJEITADO] CRITERIO 2: Historico de credito invalido - digite B ou R\n");
    aprovado = 0;
}
```

Explicação:

- **if-else if-else:** Estrutura para múltiplas condições
- **== 'R':** Comparação de caracteres
- **Validação:** Trata casos de entrada inválida
- **else:** Captura qualquer valor diferente de 'R' ou 'B'

6.8 Cálculos Matemáticos

```
float entradaMinima = valorEmprestimo * 0.20;
if (valorEntrada < entradaMinima) {
    printf("[REJEITADO] CRITERIO 4: Valor da entrada insuficiente (minimo R$ %.2f)\n",
    entradaMinima);
    aprovado = 0;
} else {
    printf("[APROVADO] CRITERIO 4: Valor da entrada adequado (%.2f%% do emprestimo)\n",
        (valorEntrada/valorEmprestimo)*100);
}
```

Explicação:

- **entradaMinima:** Calcula 20% do valor do empréstimo
- **Comparação:** Verifica se entrada oferecida é suficiente

- **Cálculo percentual:** $(\text{valorEntrada}/\text{valorEmprestimo}) * 100$

7. Resultado Final

```
if (aprovado) {
    printf("*** PARABENS! Seu emprestimo foi APROVADO! ***\n\n");
    // Detalhes da aprovação
} else {
    printf("*** EMPRESTIMO REJEITADO! ***\n\n");
    // Motivos da rejeição
}
```

Explicação:

- **if (aprovado):** Verifica se variável aprovado é verdadeira (1)
- **Condicional final:** Decide o resultado baseado em todos os critérios
- **Feedback detalhado:** Fornece informações específicas para cada caso

8. ANÁLISE DOS CRITÉRIOS

8.1 Critério 1: Relação Renda x Empréstimo

Regra Implementada:

- Valor do empréstimo $\leq 30\%$ da renda mensal
- Cálculo: `limiteEmprestimo = rendaMensal * 0.30`

Justificativa:

- Garante capacidade de pagamento do cliente
- Reduz risco de inadimplência
- Padrão utilizado no mercado financeiro

Implementação:

```
if (valorEmprestimo > limiteEmprestimo) {
    // Rejeita o empréstimo
    aprovado = 0;
```

}

8.2 Critério 2: Histórico de Crédito

Regra Implementada:

- Apenas clientes com histórico "Bom" (B) são aprovados
- Clientes com histórico "Ruim" (R) são rejeitados

Justificativa:

- Histórico de crédito é indicador de comportamento futuro
- Reduz significativamente o risco de calote
- Critério eliminatório essencial

Implementação:

```
if (historicoCredito == 'R') {  
    aprovado = 0;  
} else if (historicoCredito == 'B') {  
    // Critério atendido  
}
```

8.3 Critério 3: Estabilidade no Emprego

Regra Implementada:

- Apenas empregos "Estáveis" (E) são aceitos
- Empregos "Instáveis" (I) resultam em rejeição

Justificativa:

- Estabilidade garante continuidade da renda
- Reduz risco de desemprego durante pagamento
- Fator crucial para empréstimos de longo prazo

Implementação:

```
if (estabilidadeEmprego == 'I') {  
    aprovado = 0;  
} else if (estabilidadeEmprego == 'E') {  
    // Critério atendido  
}
```

8.4 Critério 4: Valor da Entrada

Regra Implementada:

- Entrada mínima de 20% do valor do empréstimo
- Cálculo: `entradaMinima = valorEmprestimo * 0.20`

Justificativa:

- Entrada reduz o valor a ser financiado
- Demonstra comprometimento do cliente
- Diminui risco da operação para a instituição

Implementação:

```
if (valorEntrada < entradaMinima) {  
    aprovado = 0;  
} else {  
    // Critério atendido  
}
```

9. COMO COMPILAR E EXECUTAR

9.1 Pré-requisitos

- **Editor de código:** Visual Studio Code ou similar
- **Compilador:** GCC instalado
- **Sistema:** Windows, Linux ou macOS

9.2 Passos para Execução

1. **Criar arquivo:** Salve o código como `emprestimo.c`

2. **Abrir terminal:** Use PowerShell ou Command Prompt
3. **Compilar:** Execute `gcc emprestimo.c -o emprestimo`
4. **Executar:** Execute `.\emprestimo.exe` (Windows) ou `./emprestimo` (Linux/Mac)

9.3 Exemplo de Uso

```
** Emprestimo Financeiro **  
=====
```

Digite sua renda mensal.....: 5000
Digite o valor do emprestimo.....: 1200
Digite seu historico de credito (B=Bom, R=Ruim).....: B
Digite a sua estabilidade no emprego (E=Estavel, I=Instavel.): E
Digite o valor da entrada....: 300

9.4 Prints dos testes

9.5 Empréstimo Reprovado

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure with a file named "emprestimo.c".
- Terminal:** Displays the execution of the program. The user inputs the following:

 - Digite sua renda mensal.....: 8000
 - Digite o valor do emprestimo.....: 3000
 - Digite seu historico de credito (B=Bom, R=Ruim).....: b
 - Digite a sua estabilidade no emprego (E=Estavel, I=Instavel.): e
 - Digite o valor da entrada....: 600

- Output:** The program's logic and results:
 - ANALISE DE CREDITO
 - [REJEITADO] CRITERIO 1: O valor do emprestimo (R\$ 3000.00) excede 30% da sua renda mensal (R\$ 2400.00)
 - [APROVADO] CRITERIO 2: Historico de credito bom - criterio atendido
 - [APROVADO] CRITERIO 3: Emprego estavel - criterio atendido
 - [APROVADO] CRITERIO 4: Valor da entrada adequado (20.00% do emprestimo)
- Result:** The final result is displayed as:
 - RESULTADO FINAL
 - *** EMPRESTIMO REJEITADO! ***
 - Motivos da rejeicao:
 - Verifique os criterios marcados com [REJEITADO] acima
 - Ajuste as condicoes e solicite novamente
 - Procure orientacao financeira se necessario
 - Fim do Programa!

9.6 Empréstimo Aprovado

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure with a file named "emprestimo.c".
- Terminal:** Displays the execution of the program. The user inputs the following:

 - Digite sua renda mensal.....: 8000
 - Digite o valor do emprestimo.....: 3000
 - Digite seu historico de credito (B=Bom, R=Ruim).....: b
 - Digite a sua estabilidade no emprego (E=Estavel, I=Instavel.): e
 - Digite o valor da entrada....: 600

- Output:** The program's logic and results:
 - ANALISE DE CREDITO
 - [APROVADO] CRITERIO 1: Valor do emprestimo aprovado (20.00% da renda mensal)
 - [APROVADO] CRITERIO 2: Historico de credito bom - criterio atendido
 - [APROVADO] CRITERIO 3: Emprego estavel - criterio atendido
 - [APROVADO] CRITERIO 4: Valor da entrada adequado (30.00% do emprestimo)
- Result:** The final result is displayed as:
 - RESULTADO FINAL
 - *** PARABENS! Seu emprestimo foi APROVADO! ***
 - Detalhes da aprovacao:
 - Valor do emprestimo: R\$ 2000.00
 - Valor da entrada: R\$ 600.00
 - Valor a financiar: R\$ 1400.00
 - Comprometimento da renda: 26.00%
 - Fim do Programa!

9.6 Empréstimo não tem os Critérios pra ser Aprovado

The screenshot shows a Microsoft Visual Studio Code interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** Search
- Explorer:** OPEN EDITORS (C emprestimo.c), AZURE IOT HUB
- Editor:** C emprestimo.c (C:\Users\walli...)
- Code:**

```
115     int main() {  
116         return 0;  
117     }  
118  
119     // ANALISE DE CREDITO  
120     //  
121     // [APROVADO] CRITERIO 1: Valor do emprestimo aprovado (20.00% da renda mensal)  
122     // [REJEITADO] CRITERIO 2: Historico de credito ruim - emprestimo rejeitado  
123     // [REJEITADO] CRITERIO 3: Emprego instavel - risco aumentado  
124     // [APROVADO] CRITERIO 4: Valor da entrada adequado (30.00% do emprestimo)  
125  
126     // RESULTADO FINAL  
127     //  
128     *** EMPRESTIMO REJEITADO! ***  
129  
130     // Motivos da rejeicao:  
131     // - Verifique os criterios marcados com [REJEITADO] acima  
132     // - Ajuste as condicoes e solicite novamente  
133     // ● - Procure orientacao financeira se necessario  
134  
135     // Fim do Programa!
```
- Terminal:** powershell + ...
- Status Bar:** Spaces: 4, UTF-8, C, Win32, Q

CONCLUSÃO

O sistema de aprovação de empréstimos foi desenvolvido com sucesso, demonstrando a aplicação prática das estruturas condicionais em Linguagem C. O programa implementa uma lógica robusta de análise de crédito que considera múltiplos fatores de risco.

10.1 Objetivos Alcançados

- Estruturas Condicionais:** Implementação completa de `if`, `else if` e `else`
- Validação de Dados:** Tratamento adequado de entradas inválidas
- Lógica de Negócio:** Critérios realistas baseados no mercado financeiro
- Interface Amigável:** Mensagens claras e feedback detalhado
- Código Documentado:** Comentários e explicações completas

10.2 Características Técnicas

- **Linguagem:** C ANSI padrão
- **Paradigma:** Programação estruturada
- **Estruturas:** Condicionais, entrada/saída, validação
- **Robustez:** Tratamento de erros e validação de dados
- **Portabilidade:** Funciona em diferentes sistemas operacionais

10.3 Aplicações Práticas

O sistema desenvolvido pode ser utilizado como:

- **Ferramenta educacional** para ensino de programação
- **Base para sistemas** mais complexos de análise de crédito
- **Exemplo prático** de implementação de regras de negócio
- **Demonstração** de boas práticas de programação

10.4 Melhorias Futuras

- Implementação de banco de dados para armazenar históricos
- Interface gráfica mais amigável
- Critérios adicionais de análise (idade, profissão, etc.)

- Sistema de pontuação mais sofisticado
- Integração com APIs de bureau de crédito

O projeto demonstra com sucesso a aplicação dos conceitos fundamentais de algoritmos e lógica de programação, cumprindo todos os requisitos propostos para a atividade.

REFERÊNCIAS

KERNIGHAN, Brian W.; RITCHIE, Dennis M. *C: A Linguagem de Programação*. Rio de Janeiro: Elsevier, 1989.

SCHILDT, Herbert. *C Completo e Total*. 3^a ed. São Paulo: Pearson Makron Books, 1997.

DEITEL, Harvey M.; DEITEL, Paul J. *C: Como Programar*. 6^a ed. São Paulo: Pearson Prentice Hall, 2011.

FORBELLONE, André Luiz Villar; EBERSPÄCHER, Henri Frederico. *Lógica de Programação: A Construção de Algoritmos e Estruturas de Dados*. 3^a ed. São Paulo: Prentice Hall, 2005.

VISUAL STUDIO CODE. *Visual Studio Code Documentation*. Disponível em:

<https://code.visualstudio.com/docs>. Acesso em: 16 jul. 2025.

Este documento foi elaborado como parte da atividade prática da disciplina Algoritmos e Lógica de Programação, demonstrando a aplicação das estruturas condicionais através do desenvolvimento de um sistema funcional de análise de crédito.