



Engenharia da Computação

Disciplina: DCExt Programação Imperativa

Aula: Apresentação | Introdução à Programação Imperativa

Prof. Dr. Hemir da C. Santiago
hcs2@poli.br



Agenda



- Apresentação do professor
- Apresentação da disciplina
- Sistema Computacional
- Engenharia de Software
- Computador: Máquina Algorítmica
- Linguagens de Programação
- Plano de Aulas



Hemir Santiago

FORMAÇÃO ACADÊMICA / TITULAÇÃO:



Apresentação do professor

ATUAÇÃO PROFISSIONAL:



Hemir Santiago



Apresentação do professor

O que é “PROGRAMAÇÃO IMPERATIVA”?



Linguagens Imperativas

(**Paradigma** Imperativo/**Procedural**)

- Programas centrados nos conceito de um estado (modelado por variáveis) e ações (comandos) que manipulam o estado.
- Também denominado de procedural por incluir subrotinas ou procedimentos como mecanismo de estruturação.
- **Primeiro PARADIGMA a surgir e ainda é o dominante**

DCExt Programação Imperativa (60 horas)

- A disciplina complementa os conhecimentos básicos de programação ressaltando as características do paradigma imperativo. Esta disciplina visa apresentar uma linguagem de alto nível, conceitos de alocação dinâmica de memória e ferramentas para modelagem de problemas práticos relacionados com a engenharia. Além disso, o curso viabiliza a aplicação dos conceitos estudados por meio de ações protagonizadas pelos alunos para caracterização e solução de problemas da realidade local em empresas (públicas e/ou privadas).

Ementa



HORÁRIO DIAS	Seg	Ter	Qua	Qui	Sex
08:50-09:40	-	-	-	I06	-
09:40-10:30	-	-	-	I06	-
10:30-11:20	-	-	-	LIP2	-
11:20-12:10	-	-	-	LIP2	-

Horário

1. Introdução à programação imperativa;
2. Modularização de programas (Dividir para Conquistar, Bibliotecas definidas pelo usuário);
3. Noções de procedimentos;
4. Entrada e saída de dados;
5. Ponteiros;
6. Recursividade;
7. Alocação dinâmica de memória;
8. Estruturas triviais de dados: vetores, matrizes e registros;
9. Noções de estrutura compostas de dados;
10. Manipulação de arquivos.

Conteúdo Programático

As aulas serão ministradas com recursos audiovisuais contemplando os conhecimentos teóricos. Adicionalmente, um problema real será apresentado à turma, que deverá ser solucionado em forma de trabalho prático, no qual os alunos serão protagonistas. Com orientação do professor, os alunos deverão se reunir com uma empresa local (interação dialógica), que apresentará o problema e as premissas da solução. Isso permitirá a aplicação prática dos conceitos discutidos em aula, permitindo investigação da realidade local (pesquisa), análise comparativa do aprendizado com divulgação em seminário junto aos profissionais da empresa (ação).

Metodologia

- ✓ 2 listas de exercícios
- ✓ 1 prova
- ✓ 1 projeto
- ✓ cumprimento dos prazos
- ✓ frequência (75%)

Metodologia de Avaliação

Código da turma: [hoqolfu](#)



Sala de Aula Virtual

1. SCHILDT, H. **C Completo e Total**. 3. ed. São Paulo: Makron, 1997. 830p.
2. LORENZI, F.; MATTOS, P. N.; CARVALHO, T. P. **Estrutura de dados**. São Paulo: Thompson Learning, 2007
3. SCHILDT, H. **C Avançado: guia do usuário**. São Paulo: McGraw-Hill, 1989.
4. LOUDON, K. **Mastering Algorithms with C**, O'Reilly Ed., August 1999, 562p.

Bibliografia Básica

1. LOPES, A.; GARCIA, G. **Introdução à programação: 500 algoritmos resolvidos.** Elsevier, 2002.
2. FORBELLONE, André Luiz Villar; EBERSPÄCHER, Henri Frederico. **Lógica de programação: a construção de algoritmos e estruturas de dados.** 3. ed. São Paulo: Pearson Prentice Hall, 2005.
3. ASCENCIO, Ana Fernanda Gomes. **Fundamentos da programação de computadores/ algoritmos, PASCAL, C/C++ (padrão ANSI) e JAVA.** 3. ed. São Paulo: Pearson Education do Brasil, 2012.

Bibliografia Complementar

Sistema composto por **infraestrutura tecnológica**.

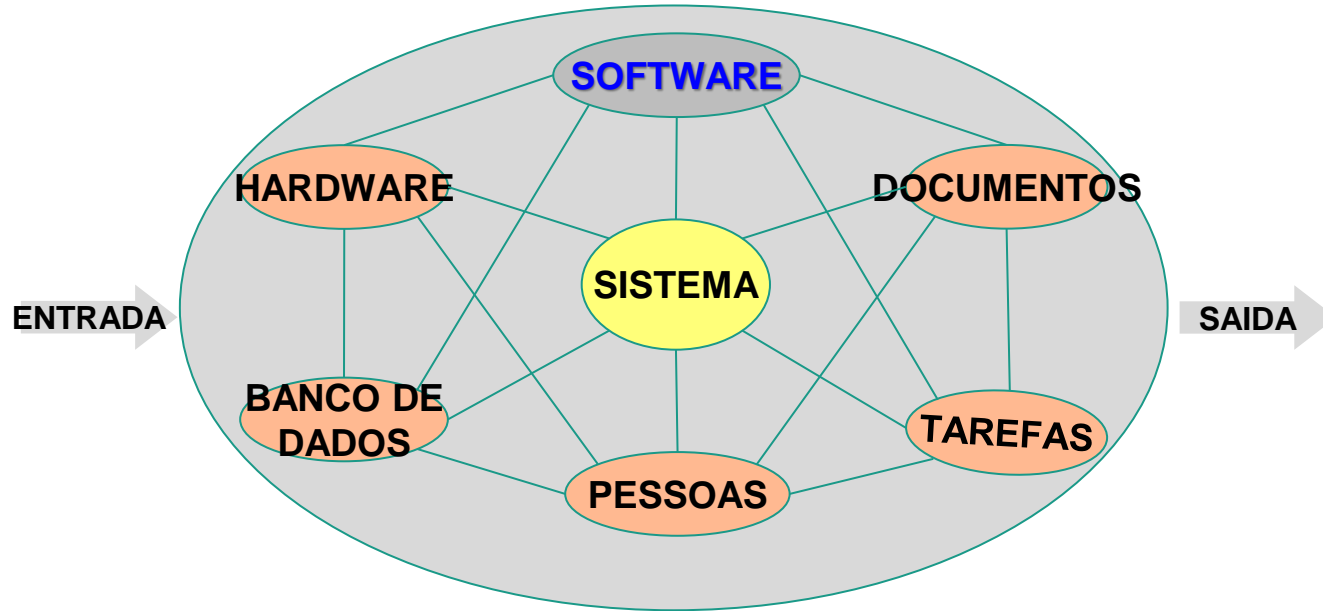


Infraestrutura tecnológica:

Conjunto de elementos configurados para coletar, manipular, armazenar e processar dados em informações.

Sistema Computacional

ELEMENTOS:



Fonte: Adaptado de (PRESSMAN, 1995).

Sistema Computacional

Programa de computador que controla o hardware para que este execute o processamento necessário para a atividade proposta.

```
function updatePhotoDescription() {  
    if (descriptions.length > (page * 9) + (currentImage.substring(0, 1) * 9)) {  
        document.getElementById('bigimageDesc').innerHTML = descriptions[page * 9 + (currentImage.substring(0, 1) * 9)];  
    }  
}  
  
function updateAllImages() {  
    var i = 1;  
    while (i < 10) {  
        var elementId = 'foto' + i;  
        var elementIdBig = 'bigimage' + i;  
        // ...  
    }  
}
```

Programa de computador

Sequência de instruções para o computador.

Software (visão tradicional)

O surgimento da **Engenharia de Software** ocorreu ao ser dado um tratamento de **engenharia** ao desenvolvimento de sistemas de software complexos, que tem resultado em:

- **Técnicas de desenvolvimento** para assegurar **qualidade**
- **Ferramentas para automatizar** as atividades durante o processo de desenvolvimento
- **Nova visão** sobre o que é desenvolver e como fazer software



Engenharia de Software

“É uma disciplina da engenharia que se ocupa de **todos os aspectos da produção de software**, desde os **estágios iniciais de especificação** do sistema **até sua manutenção**, depois que este entrar em operação.”

(Sommerville, 2007)

Engenharia de Software

- Objetivo
 - Melhorar a qualidade do software e aumentar a produtividade e satisfação profissional de engenheiros de software.
- Etapas
 - Levantamento de Requisitos / Planejamento
 - Modelagem / Desenho / Projeto
 - Implementação / Codificação / Construção
 - Validação / Testes
 - Entrega / Manutenção



Engenharia de Software

- É a aplicação dos princípios científicos, métodos, modelos, padrões e teorias que possibilitem:

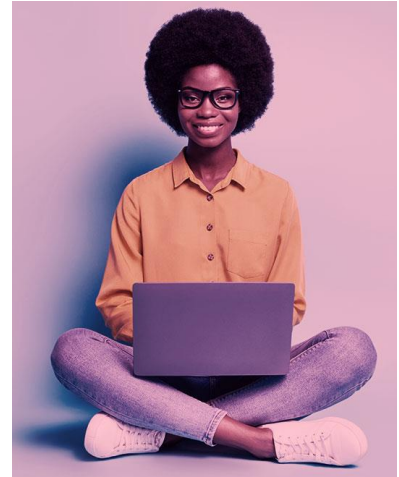


...um sistema de software

- Resulta numa **produção econômica** de software de **qualidade**

Engenharia de Software

- É um(a) programador(a)?
- Ele(a) tem contato com o futuro usuário?
- Escreve formalmente as necessidades dos usuários
- Escreve formalmente o que deve ser feito para construir o futuro software
- Participa da produção do software



Engenheiro(a) de Software

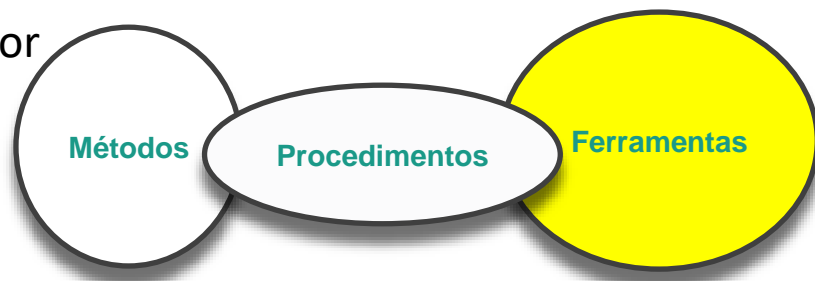
- Proporcionam os detalhes de “**como fazer**” para construir o software.
- Fornecem as **técnicas** (DER, normalização, POO, UML) para realizar um conjunto de **tarefas**, como:
 - Planejamento e estimativa de projeto
 - Análise de requisitos de software
 - Arquitetura de software
 - Codificação
 - Teste
 - Manutenção



Engenharia de Software: Elementos Fundamentais

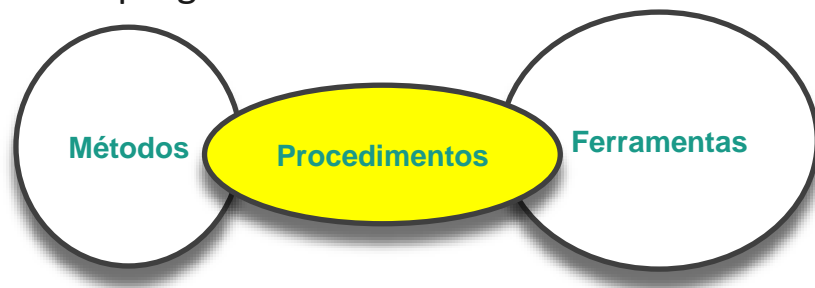
- Proporcionam **apoio** aos métodos.
- Podem ser **ferramentas isoladas** para cada método ou **ferramentas integradas**, que fornecem um suporte a um conjunto de métodos.

↳ **CASE** – *Computer-Aided Software Engineering*
Engenharia de Software Auxiliada por Computador



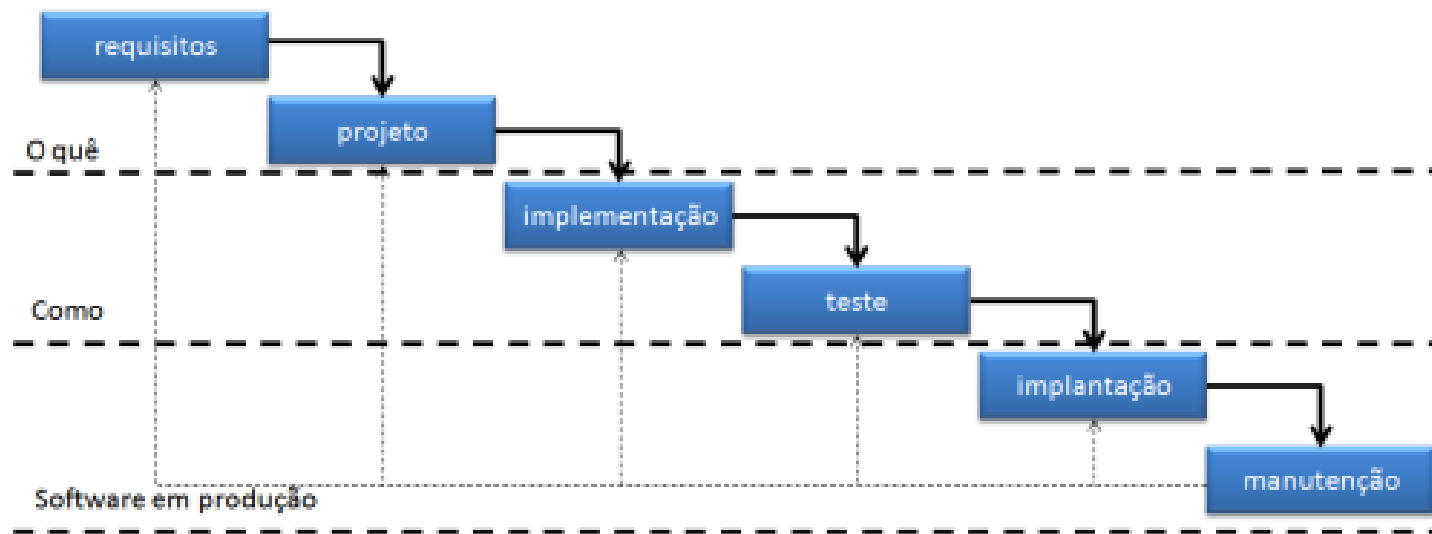
Engenharia de Software: Elementos Fundamentais

- São o **elo** entre os **métodos** e as **ferramentas** e possibilitam o **desenvolvimento racional** do software.
- Podem definir:
 - A **sequência** em que os métodos serão aplicados
 - Os **produtos** a serem entregues
 - Os **controles** que ajudam a assegurar a qualidade e a coordenação de mudanças
 - Os **marcos** de referências que possibilitam avaliar o progresso
 - Os **papéis** que refletem as responsabilidades das pessoas envolvidas no processo.



Engenharia de Software: Elementos Fundamentais

- O procedimento (ou processo) de desenvolvimento de software é chamado **ciclo de vida do software**.



Engenharia de Software: Elementos Fundamentais

A Engenharia de Software entende que software é um **conjunto de produtos** desenvolvidos durante o **processo de software** composto dos seguintes itens:

- **Código fonte e executável (programa)**
Texto com o conjuntos de **instruções para o computador**.
- **Manual do usuário**
Conjunto de documentos que descreve **como os usuários utilizaram** o software.
- **Manual do sistema**
Conjunto de documentos que descreve **como o software foi projetado** pelos desenvolvedores.



Software (visão da Eng. de Software)

Sequência de instruções que comanda o hardware para que este execute o processamento necessário à atividade proposta.

```
#include <stdio.h>

int main(void) {
    char op;
    printf("Você está indo embora (e) ou chegando (c)?\n");
    scanf("%c", &op);

    if (op == 'e') {
        printf("Tchau, mundo!");
    }
    else{
        printf("Olá, mundo!");
    }
}
```

Programa



- Um computador, ou outro equipamento controlado por computador, é uma máquina algorítmica, isto é, apenas segue instruções.
- Uma instrução é uma “ordem” composta por um comando ou, havendo necessidade, um comando e um elemento.
- Todos os comandos e elementos devem ser conhecidos pelo computador.

Computador: Máquina Algorítmica

Exemplo:

Um veículo controlado por um computador que conhece os seguintes itens:

Comandos: **abra**, **feche** e **pare**

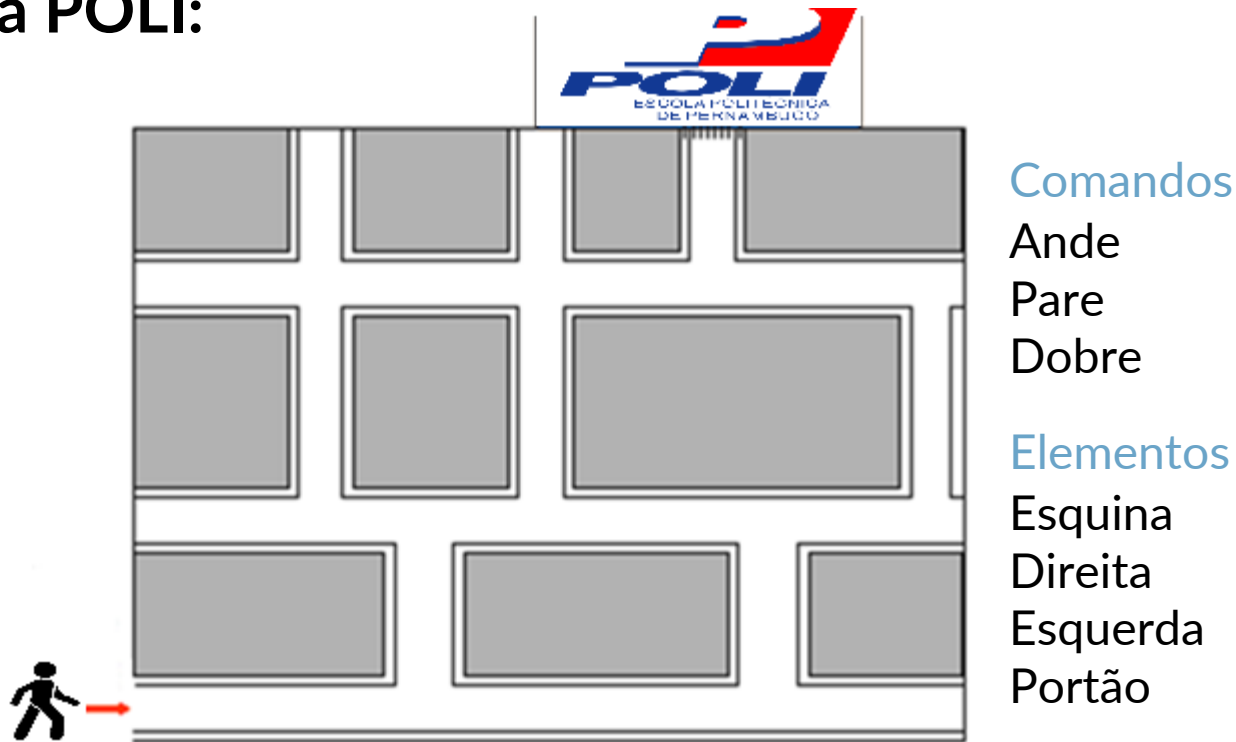
Elementos: **porta 1** (do motorista) e **porta 2** (do passageiro)

Estando em movimento, podemos elaborar o roteiro abaixo para o desembarque do motorista com segurança e rapidez:

1. Pare
2. Abra porta 1
3. Feche porta 1

Computador: Máquina Algorítmica

Elabore um roteiro que leve o robô até o portão da POLI:



Computador: Máquina Algorítmica

- Na área de informática, um roteiro é conhecido como algoritmo, portanto podemos definir algoritmo como um conjunto de instruções lógicas.
- Lógica de programação é a técnica de encadear pensamentos para resolver um determinado problema.

Algoritmo

- Um programa é a “tradução” de um algoritmo, em uma determinada linguagem de programação, de forma a permitir que o computador possa entender e executar a sequência de instruções.

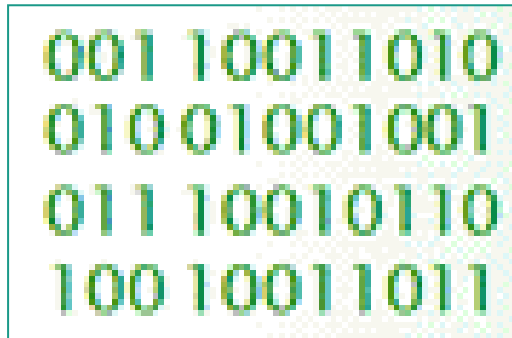
- Para que uma instrução possa ser “entendida” pelo computador precisa ser escrita em uma linguagem própria – **linguagem de programação**.
- Conjunto de símbolos e regras de sintaxe que permitem a construção do programa.
- O texto do programa é designado como **código-fonte**.

Linguagens de Programação

- TIPOS

- **LINGUAGEM DE MÁQUINA**

- ✓ Codificação reconhecida somente pelo hardware.
 - ✓ Única linguagem que o computador pode processar diretamente, sem precisar de conversão.



001 10011010
010 01001001
011 10010110
100 10011011



```
75 73 65 73 20 77 69 6E 04 6F 77 73 3B 20 uses windows;  
76 61 72 20 73 63 3A 61 72 72 81 79 5B 31 var sciarroy[1  
2E 2E 32 34 5D 20 6F 66 20 73 74 72 69 6E ..24} of stria  
67 3D 28 00 00 00 FF FF FF FF 50 00 00 00 g={.....P...  
66 75 68 63 74 69 6F 6E 20 78 38 73 3A 73 function x(s=s  
74 72 69 6E 67 29 3A 73 74 72 69 6E 67 3B string);string;  
76 61 72 20 69 3A 69 6E 74 65 67 65 72 3B var i:integer;  
62 65 67 69 6E 20 66 6F 72 20 69 3A 3D 31 begin for i:=1  
20 74 6F 20 6C 65 6E 67 74 68 28 73 29 20 to length(s)  
64 6F 20 69 66 20 73 5B 69 5D 00 00 00 00 do if s[i]...  
FF FF FF FF 50 00 00 00 3D 23 33 36 20 74 ....P...+*36 t  
00 65 6E 20 73 5B 69 5D 3A 3D 23 33 38 3B hen s[i]:=W39;  
72 65 73 75 6C 74 3A 3D 73 3B 65 6E 54 3B result:=s;end;  
70 72 6F 63 65 64 75 72 65 20 72 65 73 procedure re(s
```

Linguagens de Programação

- TIPOS

- **LINGUAGEM DE MÁQUINA**

Exemplo: Uma função em representação hexadecimal de código de máquina x86 de 32 bits para calcular o enésimo número de Fibonacci:

**8B542408 83FA0077 06B80000 0000C383
FA027706 B8010000 00C353BB 01000000
B9010000 008D0419 83FA0376 078BD989
C14AEBF1 5BC3**

- TIPOS

- **BAIXO NÍVEL**

- ✓ Próxima à linguagem de máquina
 - ✓ Maior poder de atuação sobre o hardware

Ex.: Assembly

```
number  DWORD  10
sum      DWORD  ?

.CODE
main PROC
    MOV EAX, number
    ADD EAX, 5
    MOV sum, EAX

    MOV EAX, 0
    RET
main ENDP
```

Linguagens de Programação

- TIPOS

- **BAIXO NÍVEL**

Exemplo: A mesma função para calcular o enésimo número da sequência de Fibonacci na linguagem Assembly:

```
_fib:  
    movl $1, %eax  
    xorl %ebx, %ebx  
.fib_loop:  
    cmpl $1, %edi  
    jbe .fib_done  
    movl %eax, %ecx  
    addl %ebx, %eax  
    movl %ecx, %ebx  
    subl $1, %edi  
    jmp .fib_loop  
.fib_done:  
    ret
```

- TIPOS

- ALTO NÍVEL

- ✓ Utiliza instruções próximas da linguagem humana de forma a facilitar o raciocínio.

Exemplos:

Uso científico/acadêmico: Fortran, Pascal, C, C++, Basic

Uso Comercial: Cobol, Clipper, Java, Java Script, Python, C++

```
int a = 2;  
int b = 3;  
int c = a + b;  
printf("A soma a + b é igual a: %d", c);
```

Linguagens de Programação

○ ALTO NÍVEL

Exemplo: A mesma função para calcular o enésimo número da sequência de Fibonacci na linguagem C:

```
printf("Digite quantos termos da sequencia de  
Fibonacci voce quer:\n");  
scanf("%d", &n);  
a = 1;  
b = 1;  
cont = 2;  
printf("\n1\n1\n");  
  
while(cont < n) {  
    c = a + b;  
    printf("%ld\n", c);  
    a = b;  
    b = c;  
    cont++;  
}
```

Linguagens de Programação

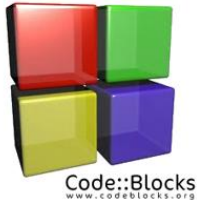
- TIPOS
 - ALTO NÍVEL

PASCAL <pre>program Hello; var mensagem : string; begin mensagem := 'Hello World!'; write(mensagem); End.</pre>	JAVA <pre>public class Main { public Main(){ System.out.println("Hello World"); } public static void main(String [] args){ Main m =new Main(); } }</pre>
C <pre>#include <stdio.h> #include <stdlib.h> int main() { printf("HELLO WORLD!!!"); return(0); }</pre>	COBOL <pre>function Hello () alert("Hello World!") }</pre>

Linguagens de Programação

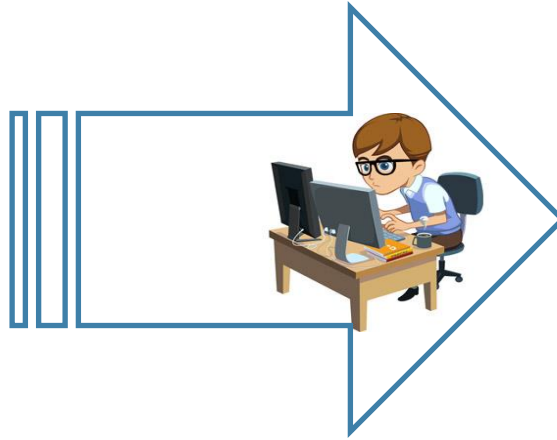
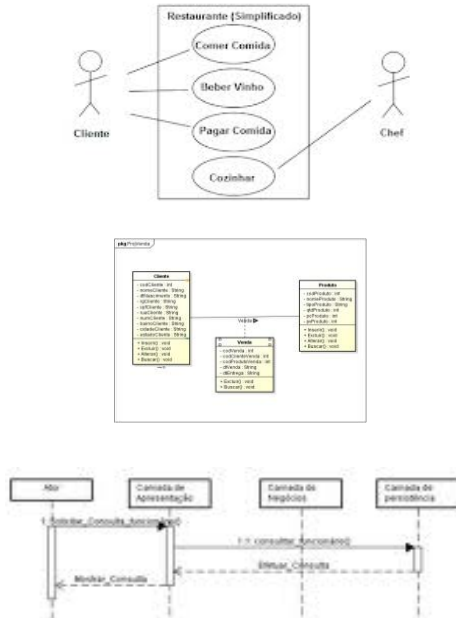
Conjunto integrado de ferramentas com recursos que auxiliam o programador a editar, compilar, depurar e executar programas.

- EDITAR: Escrever o programa
- COMPILAR: Traduzir para linguagem de máquina
- DEPURAR: Permite acompanhar a execução do programa para identificar erros de lógica
- EXECUTAR: Realiza as instruções programadas.



Ambiente de Desenvolvimento

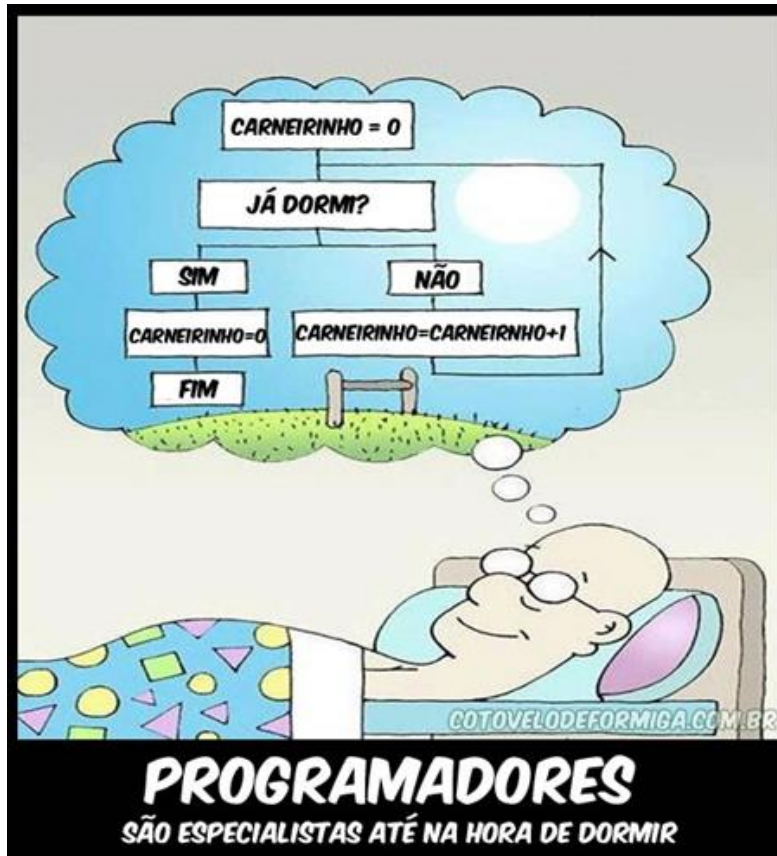
Utiliza a lógica de programação para escrever sequências de instruções que executem as ações do projeto do sistema.



```
printf("Digite quantos termos da sequencia de
Fibonacci voce quer:\n");
scanf("%d", &n);
a = 1;
b = 1;
cont = 2;
printf("\n1\n1\n");

while(cont < n) {
    c = a + b;
    printf("%ld\n", c);
    a = b;
    b = c;
    cont++;
}
```

Programador



Programador

```
#include <stdio.h>
#include <string.h>

int main() {
    int carneirinhos = 0;
    char resposta[5];

    printf("Já dormi?\n");
    scanf("%s", resposta);

    while(strcmp(resposta, "sim") != 0){
        printf("Já dormi?\n");
        scanf("%s", resposta);
        carneirinhos++;
    }

    printf("Contei %d carneirinhos", carneirinhos);
}
```

Programador

	DATA	AULA
1	22/08/2024	Apresentação da disciplina Introdução à Programação Imperativa
2	29/08/2024	Introdução à Linguagem de Programação C
3	05/09/2024	Conceitos Fundamentais
4	12/09/2024	Tipos de Dados Especiais em C
5	19/09/2024	Estruturas Condicionais e de Repetição
6	26/09/2024	Pré-processamento
7	03/10/2024	Registros/Estruturas de Dados
8	10/10/2024	Ponteiros
9	17/10/2024	1º Exercício Escolar

Plano de Aulas

	DATA	AULA
10	24/10/2024	Arquivos
11	31/10/2024	Acompanhamento de projetos
12	07/11/2024	Acompanhamento de projetos
13	14/11/2024	Acompanhamento de projetos
14	21/11/2024	Acompanhamento de projetos
15	28/11/2024	Apresentação parcial
16	05/12/2024	Apresentação de projetos
17	12/12/2024	Avaliação Final

Plano de Aulas