MAUÁ ESCOLA DE ENGENHARIA

1

Introdução à Linguagem C

Professores

Professores da disciplina ECM404

- Alexandre Harayashiki Moreira
- Murilo Zanini de Carvalho
- Roberto Scalco (responsável pela disciplina)

Conteúdo do curso

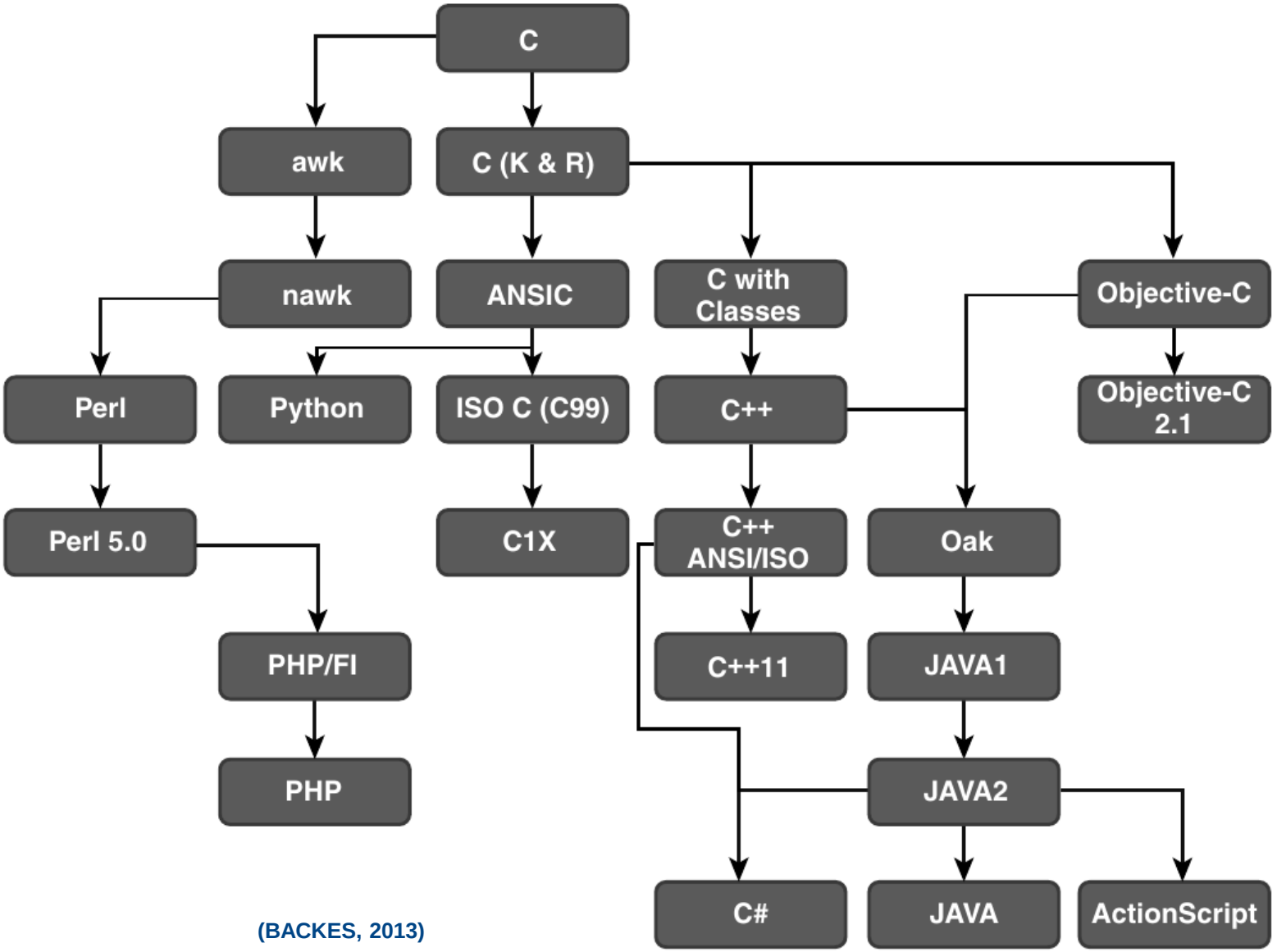
Tópicos a serem abordados

- **Linguagem C**: principais conceitos de programação com a linguagem C;
- **Técnicas de projeto de programas**: abstração funcional e de dados, *top-down*, *bottom-up*, modularização, testes … ;
- **Tipos abstratos de dados, seus algoritmos e aplicações**: sequências, pilhas, filas, grafos, dígrafos, listas ligadas e árvores;
- **Introdução aos bancos de dados relacionais**: SQL e programação C com SQL.

História

- A **linguagem BCPL** e, depois, a **linguagem B**, projetada por **Ken Thompson** em 1970 nos **Laboratórios Bell**, para o primeiro sistema UNIX em um computador PDP-7 DEC, **influenciaram diretamente C**;
- Em **1972**, **Dennis Ritchie** nos **Laboratórios Bell** projeta **C** e, em **1978**, a **publicação** de *The C Programming Language* por **Kernighan** & **Ritchie** causou uma **revolução** no **mundo** da **computação**.
- Em **1983**, o **American National Standards Institute** (**ANSI**) criou o padrão ANSI, ou "ANSI C", foi concluída no final de 1988;
- Última revisão da linguagem: **C11** - ISO/IEC 9899:2011.

Linguagens influenciadas por C



(BACKES, 2013)

Características

- **Linguagem genérica procedural** – tal como Python;
- **Sintaxe enxuta**: a descrição da sintaxe completa de C ocupa apenas 49 páginas (KERNIGHAN; RITCHIE, 1988);
- **Portável**: o código fonte do compilador pode ser compilado facilmente para muitos processadores diferentes;
- **Eficiente**;
- **Próximo da máquina**: permite o controle total dos recursos do computador – o programador pode fazer virtualmente tudo... ☠ ☠ ☠ ☠ ☠

Porque aprender?

- **C** é uma linguagem muito utilizada nas **Engenharias** de **Computação**, **Elétrica** e **Controle e Automação**;
  - Programação em rede;
  - Jogos;
  - Programação de sistemas embarcados;
  - Infraestrutura;
  - Redes neurais artificiais;
  - Sistemas de controle discreto;
  - Gerenciadores de dispositivos;
  - Aplicações em tempo real;
  - …

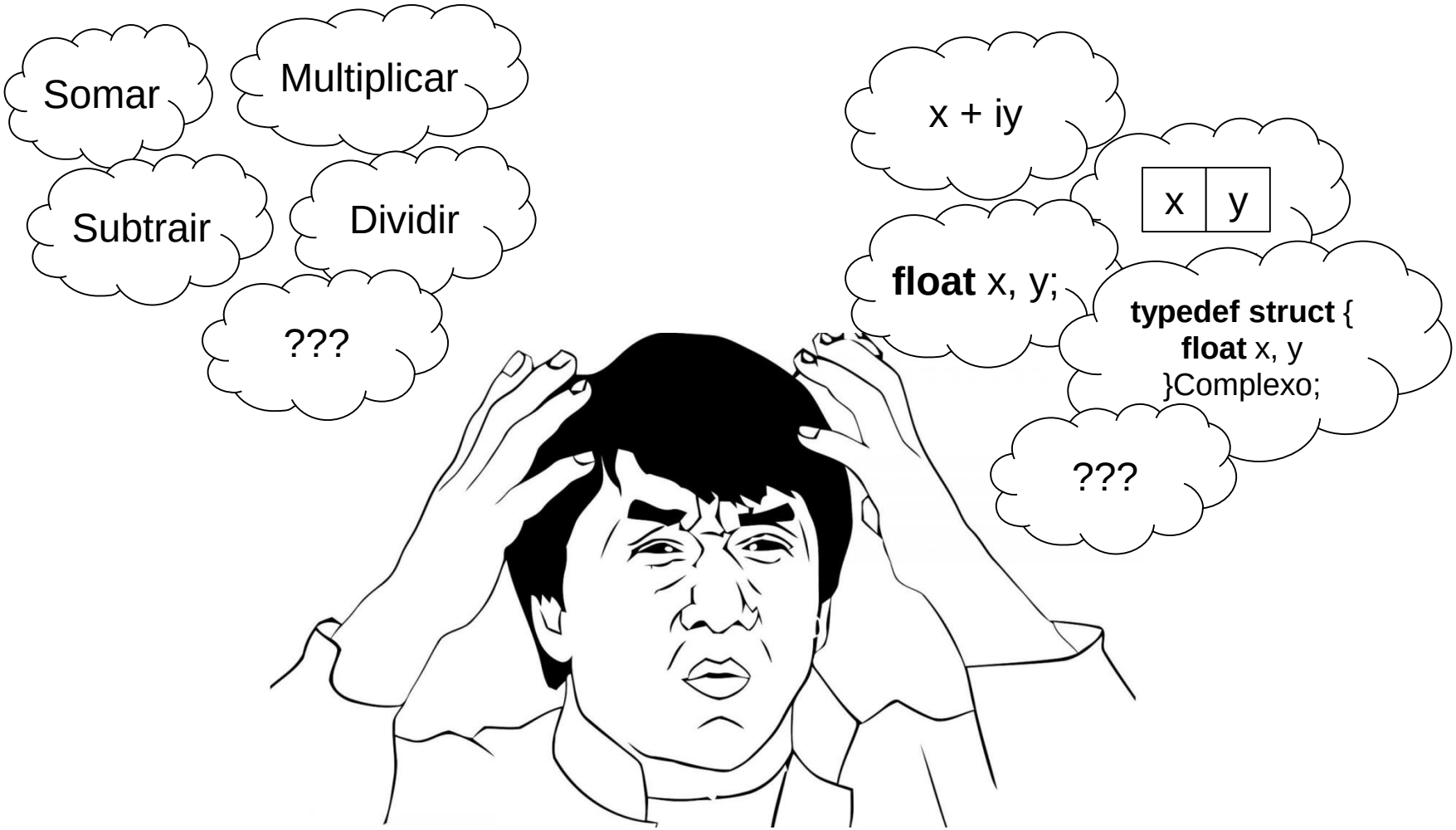
Técnicas de projeto de programas

Abstração

- Projetar **sistemas computadorizados** significa saber **traduzir**...
  - **Requisitos** do **cliente** em um **projeto** de **software**;
  - **Projeto** de ***software*** em um **sistema funcional**.
- As **traduções** envolvem **abstração**:
  - **Funcional**: como dividir as **funcionalidades esperadas** do **sistema** em **módulos** (ou **funções**), de modo a **facilitar** seu **entendimento**, **manutenção** e **evolução?**
  - **Dados**: que **estruturas** de **dados** são mais convenientes para serem utilizadas na **solução** do **problema** em questão e **manipuladas** pelas **funções** anteriormente elaboradas?
- Para **melhorar** as **capacidades** de **abstração**: **treino**, **treino**, **treino** …

Abstração

- **Exemplo** – aritmética de números complexos



Técnicas de projeto de programas

Estratégias para lidar com sistemas

- É sabido que sistemas **complexos** possuem algum tipo de **hierarquia** (SIMON, 1962);
- Em linguagens procedurais, existem **duas estratégias básicas** de **projeto** para se hierarquizar um sistema:
  - **Estratégia *top-down***
    - É um processo de **refinamento passo a passo** (WIRTH, 1971) – parte-se de um problema maior, cuja solução não se conhece imediatamente e gradualmente decompõe-se em partes menores até que se consiga determinar uma solução para essas partes.
    - Daí a solução final é a hierarquia de soluções encontradas
  - **Estratégia *bottom-up***:
    - Parte-se da solução particular de problemas simples, que vão então compondo a solução de problemas maiores.
  - **Na prática**: intercalam-se estas duas técnicas.

Técnicas de projeto de programas

Modularização

- Sistemas computadorizados de tamanho arbitrário devem ser organizados internamente em **módulos**;
- Um **módulo** é um bloco de código (conjunto de comandos contíguos) que
  - Possui um nome;
  - Um conjunto próprio de variáveis;
  - Pode ser invocado por outros módulos.
- **Módulos** podem representar **procedimentos** e **funções**;
- **Vantagens**: **evitar** a **redundância** de código e **promover** a **reutilização** de **código**;
- **Objetivo**: módulos **coesos** e com **baixo acoplamento**.

Tipo Abstrato de Dados (TAD)

O que é?

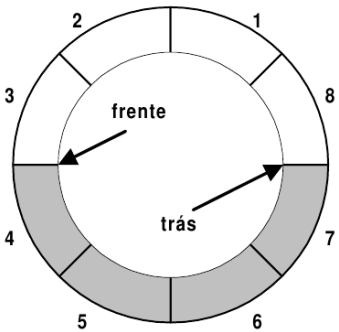
- Representa alguma **estruturação** de **dados** e as **operações** (**algoritmos**) que a **manipulam**;
- O **projeto** da estrutura de dados é **iniciado** por uma **análise *top-down*** do problema, de onde se isolam as suas **especificações** que, depois, são **utilizadas** em seu **projeto**;
- O **TAD** então é **implementado** a partir desta **especificação**, independente dos problemas que o utilizarão e então reutilizado – emprega-se então um processo ***bottom-up*** de solução;
- Existem algumas **estruturas** de **dados** consagradas e que são **utilizadas** em uma **gama variada** de **problemas**: **filas**, **pilhas**, **listas**, **grafos**…

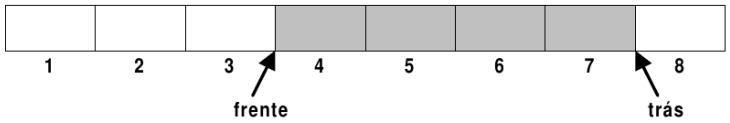
Exemplo – TAD Fila

Simulação de sistema de atendimento bancário

Armazenamento de caracteres conforme são digitados

etc …





Inserir (no final)
Remover (da frente)
Verificar se cheia
Verificar se vazia

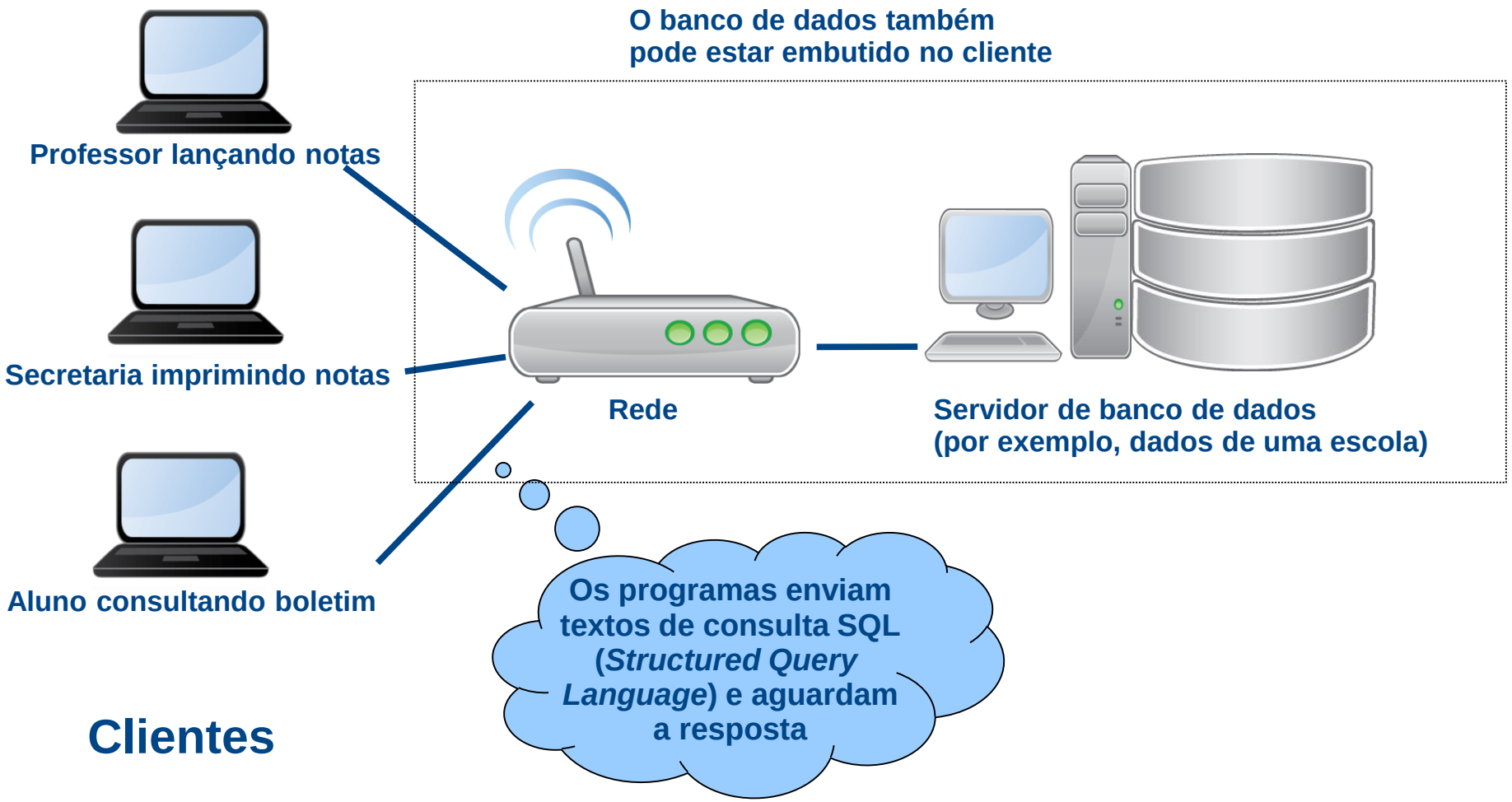
etc …

```c
#ifndef _FILA_H_
#define _FILA_H_

#define TAM_MAX_FILA 10
typedef int Elemento; /*Tipo do elemento da fila*/
typedef struct {
    int contagem; /* total de elementos na fila */
    int frente; /* índice da frente da fila */
    int tras; /* índice da traseira da fila */
    int  elementos[TAM_MAX_FILA]; /* armazenar elementos da fila */
} Fila;

/* Inicia
void Fila_
/* Insere
int Fila_I
/* Remove
int Fila_R
/* Retorna
int Fila_C
/* Retorna
int Fila_V
/* Retorna
int Fila_C

#endif
```

```c
#include "fila.h"

/* Inicializar a fila - fila vazia - contagem zero */
void Fila_Inicializar(Fila* fila){
        fila->contagem = 0;
        fila->frente = 0;
        fila->tras = 0;
}

/* Insere x na fila - o retorno indica se conseguiu ou não */
int Fila_Inserir(Fila* fila, Elemento x){
        if(!Fila_Cheia(fila)) {
                fila->elementos[fila->tras] = x;
                fila->tras = (fila->tras + 1) % TAM_MAX_FILA;
                fila->contagem++;
                return 1;
        }
        else
                return 0;
}

/* Remove um elemento da fila em x - o retorno indica se conseguiu ou não */
int Fila_Remover(Fila* fila, Elemento *x){
        if(!Fila_Vazia(fila)) {
                *x = fila->elementos[fila->frente];
                fila->frente = (fila->frente + 1) % TAM_MAX_FILA;
                fila->contagem--;
                return 1;
        }
        else
                return 0;
}

/* Retorna o tamanho atual da fila */
int Fila_ObterTamanho(Fila* fila){
        return fila->contagem;
}

/* Retorna se a fila está vazia ou não */
int Fila_Vazia(Fila* fila){
        return fila->contagem == 0;
}

/* Retorna se a fila está cheia ou não */
int Fila_Cheia(Fila* fila){
        return fila->contagem == TAM_MAX_FILA;
}
```

Necessidade | **Abstração** | **Implementação**

Programas em C com banco de dados relacional



Variáveis, entrada e saída e expressões

Python

```
"""
        FOGUETE.PY
        Calcula a máxima altura, o tempo para
        atingir a máxima altura, a máxima distância
        horizontal e o tempo de retorno à terra a
        partir do lançamento de um foguete com
        uma inclinação de teta graus e velocidade
        de v m/s.
"""

from math import pi, sin, cos
g = 10.0

teta = float(input('Digite o ângulo de lancamento: '))
teta = teta*pi/180;
v = float(input("Digite a velocidade: "))
tmax = v*sin(teta)/g;
print('Tempo para altura máxima: %8.2f s' % tmax)
hmax = (v*sin(teta))**2/(2*g)
print("Altura máxima: %8.2f m" % hmax)
tcheg = 2*v*sin(teta)/g
print('Tempo de chegada: %8.2f s' % tcheg)
dmax = 2*v**2*cos(teta)*sin(teta)/g
print("Máxima distância horizontal: %8.2f m" % dmax)
```

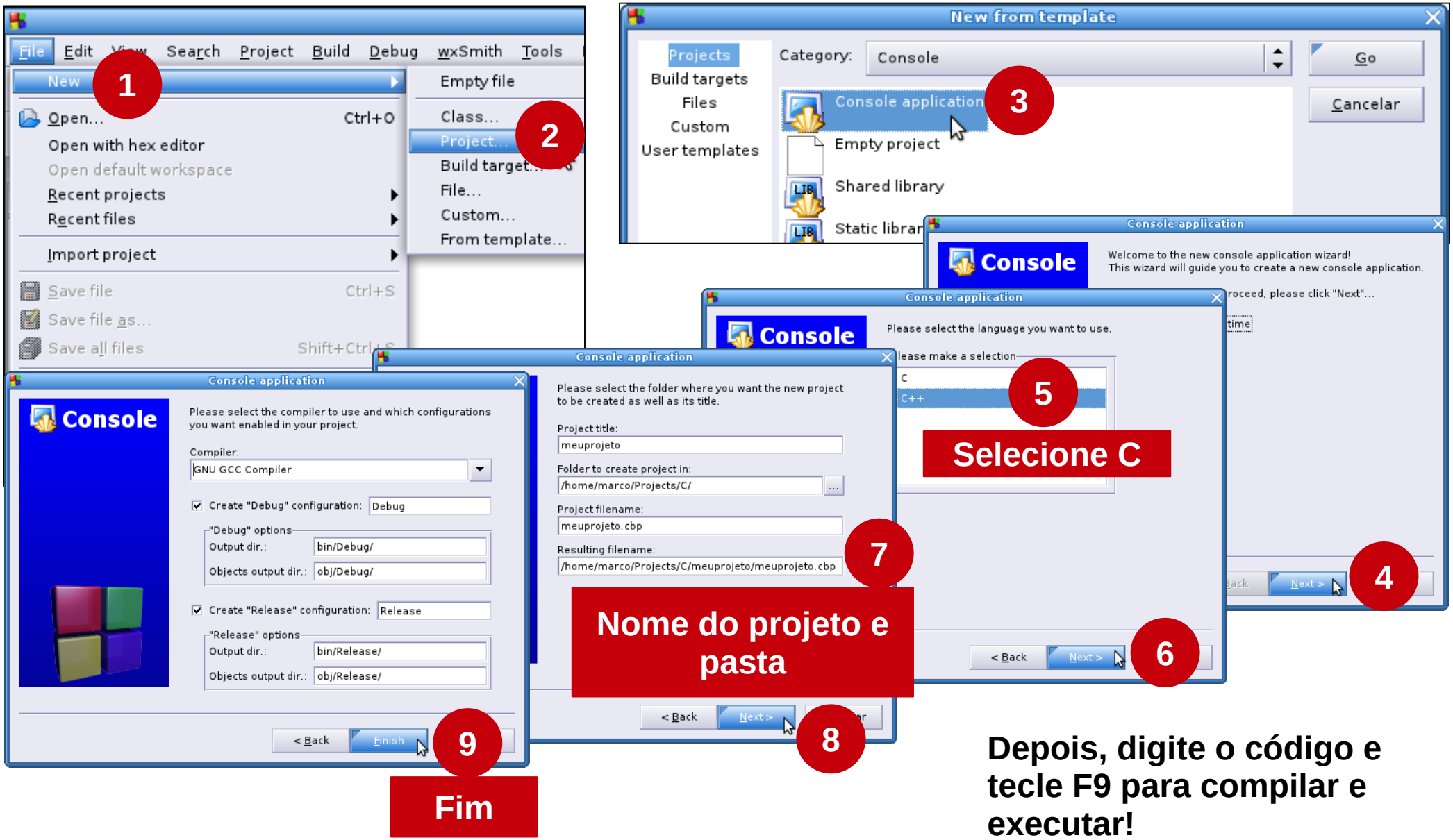
C

```
/*
        FOGUETE.C
        Calcula a máxima altura, o tempo para
        atingir a máxima altura, a máxima distância
        horizontal e o tempo de retorno à terra a
        partir do lançamento de um foguete com
        uma inclinação de teta graus e velocidade
        de v m/s.
*/
#include<stdio.h>
#include<math.h>
#define g 10.0
int main(void) {
    float teta, v, hmax, tmax, tcheg, dmax;
    printf("Digite o angulo de lancamento: ");
    scanf("%f", &teta);
    teta = teta*M_PI/180;
    printf("Digite a velocidade: ");
    scanf("%f", &v);
    tmax = v*sin(teta)/g;
    printf("Tempo para altura maxima: %8.2f s\n", tmax);
    hmax = pow(v*sin(teta),2)/(2*g);
    printf("Altura maxima: %8.2f m\n", hmax);
    tcheg = 2*v*sin(teta)/g;
    printf("Tempo de chegada: %8.2f s\n", tcheg);
    dmax = 2*pow(v, 2)*cos(teta)*sin(teta)/g;
    printf("Maxima distancia horizontal: %8.2f m\n",
                                                dmax);
    return 0;
}
```

Atividade

- Para o programa apresentado e para os próximos:
  - **Identifique** que **diferenças** e **similaridades existem** nos **dois programas** apresentados anteriormente (~10 min.);
  - **Tente inferir** no **programa** em **C** o **significado** dos **operadores**, **símbolos** e **funções** em relação ao que foi apresentado no programa em Python.
  - **Discussão** …
  - **Teste** o **programa** no **Code::Blocks**!

Usando o CodeBlocks



Depois, digite o código e tecle F9 para compilar e executar!

Estruturas condicionais

Python

```
"""
    RAIZES.PY
    Calcula as raízes da equação de
    segundo grau de coeficientes
    a, b, c.
"""

from math import sqrt


a = float(input('Digite o valor de a: '))
b = float(input('Digite o valor de b: '))
c = float(input('Digite o valor de c: '))
delta = (b**2 - 4*a*c)



if delta < 0:
    print("Não existem raízes reais")
else:
    r1 = (-b + sqrt(delta))/(2*a)
    r2 = (-b - sqrt(delta))/(2*a)
    if delta == 0:
        print("Existe uma única raiz: %4.2f" % r1)
    else:
        print("As raízes são: %4.2f e %4.2f" % (r1, r2))
```

C

```
/*
    RAIZES.C
    Calcula as raízes da equação de
    segundo grau de coeficientes
    a, b, c.
*/
#include<stdio.h>
#include<math.h>
int main(void) {
    float a, b, c, delta, r1, r2;
    printf("Digite o valor de a: ");
    scanf("%f", &a);
    printf("Digite o valor de b: ");
    scanf("%f", &b);
    printf("Digite o valor de c: ");
    scanf("%f", &c);
    delta = (pow(b,2) - 4*a*c);
    if( delta < 0 )
        printf("Nao existem raizes reais \n");
    else {
        r1 = (-b + sqrt(delta))/(2*a);
        r2 = (-b - sqrt(delta))/(2*a);
        if( delta == 0 )
            printf("Existe uma unica raiz: %4.2f\n",
                    r1);
        else
            printf("As raizes são: %4.2f e %4.2f\n",
                    r1, r2);
    }
    return 0;
}
```

Estruturas condicionais – também funciona!

Python

```
"""
    RAIZES.PY
    Calcula as raízes da equação de
    segundo grau de coeficientes
    a, b, c.
"""

from math import sqrt


a = float(input('Digite o valor de a: '))
b = float(input('Digite o valor de b: '))
c = float(input('Digite o valor de c: '))
delta = (b**2 - 4*a*c)



if delta < 0:
    print("Não existem raízes reais")
else:
    r1 = (-b + sqrt(delta))/(2*a)
    r2 = (-b - sqrt(delta))/(2*a)
    if delta == 0:
        print("Existe uma única raiz: %4.2f" % r1)
    else:
        print("As raízes são: %4.2f e %4.2f" % (r1, r2))
```

C

```
/*
    RAIZES.C
    Calcula as raízes da equação de
    segundo grau de coeficientes
    a, b, c.
*/
#include<stdio.h>
#include<math.h>
int main(void) {
float a, b, c, delta, r1, r2;
printf("Digite o valor de a: ");
scanf("%f", &a);
printf("Digite o valor de b: ");
scanf("%f", &b);
printf("Digite o valor de c: ");
scanf("%f", &c);
delta = (pow(b,2) - 4*a*c);
if( delta < 0 )
printf("Nao existem raizes reais \n");
else {
r1 = (-b + sqrt(delta))/(2*a);
r2 = (-b - sqrt(delta))/(2*a);
if( delta == 0 )
printf("Existe uma unica raiz: %4.2f\n", r1);
else
printf("As raizes são: %4.2f e %4.2f\n", r1, r2);
}
return 0;
}
```

Estruturas de repetição

Python

```
"""
    MDC.PY
    Calcula o máximo divisor comum entre
    dois números inteiros.
"""



a = int(input("Digite o valor de a: "))
b = int(input("Digite o valor de b: "))
r1 = a
r2 = b


while r1 != r2:
    if r1 > r2:
        r1 = r1 - r2
    else:
        r2 = r2 - r1
print('O máximo divisor comum de %i e %i é %i' \
        % (a, b, r1))
```

C

```
/*
    MDC.C
    Calcula o máximo divisor comum entre
    dois números inteiros.
*/
#include<stdio.h>
int main(void)
{
    int a, b, r1, r2;
    printf("Digite o valor de a: ");
    scanf("%d", &a);
    printf("Digite o valor de b: ");
    scanf("%d", &b);
    r1 = a;
    r2 = b;
    while( r1 != r2 )
        if(r1 > r2)
            r1 = r1 - r2;
        else
            r2 = r2 - r1;
    printf("O maximo divisor comum de %d e %d e %d\n",
            a, b, r1);
    return 0;
}
```

Variáveis indexadas

Python

```
"""
    ESTAT.PY
    Calcula a média e o desvio padrão
    de valores presentes em uma lista.
"""

from math import sqrt

N = 10



soma = 0
valores = []
for i in range(N):
    valor = float(input('Digite o valor %i: ' \
            % i))
    valores.append(valor)
    soma = soma + valores[i]
media = soma/N
soma = 0
for i in range(N):
    soma = soma + (valores[i] - media)**2
desviop = sqrt(soma/(N-1))
print('Média: %8.3f Desvio: %8.3f' \
          % (media, desviop))
```

C

```
/*
    ESTAT.C
    Calcula a média e o desvio padrão
    de valores presentes em um vetor.
*/
#include<stdio.h>
#include<math.h>
#define N 10
int main(void)
{
    float valores[N];
    float soma, desviop, media;
    int i;
    soma = 0;
    for( i=0; i<N; i++)
    {
        printf("Digite o valor %d: ", i + 1);
        scanf("%f", &valores[i]);
        soma = soma + valores[i];
    }
    media = soma/N;
    soma = 0;
    for( i=0; i<N; i++)
        soma = soma + pow(valores[i] - media, 2.0);
    desviop = sqrt(soma/(N-1));
    printf("Media: %8.3f  Desvio: %8.3f\n", media,
                desviop);
    return 0;
}
```

Cadeias de caracteres

Python

```
"""
    PALINDROMO.PY
    Verifica se uma cadeia de
    caracteres contém um palíndromo.
"""

frase = input('Digite uma frase: ')
igual = True
i = 0
j = len(frase)-1
while igual and (i < j):
    igual = frase[i] == frase[j]
    i += 1
    j -= 1
if igual:
    print(frase, 'é um palíndromo!' )
else:
    print(frase, 'NÃO é um palíndromo!' )
```

C

```
/*
    PALINDROMO.C
    Verifica se uma cadeia de
    caracteres contém um palíndromo.
*/
#include<stdio.h>
#include<string.h>
#define N 100
int main(void)
{
    char frase[N];
    int i, j, igual;
    printf("Digite uma frase: ");
    setbuf(stdin, 0);
    fgets(frase, N, stdin);
    if (frase[strlen(frase)-1] == '\n')
        frase[strlen(frase)-1] = '\0';
    igual = 1;
    i = 0;
    j = strlen(frase) - 1;
    while( igual && (i < j) )
    {
        igual = frase[i] == frase[j];
        i++;
        j--;
    }
    if( igual )
        printf( "%s e um palindromo!\n", frase );
    else
        printf( "%s NAO e um palindromo!\n",
                frase );
    return 0;
}
```

Cadeias de caracteres

Python

```
"""
    PALINDROMO.PY
    Verifica se uma cadeia de
    caracteres contém um palíndromo.
"""

frase = input('Digite uma frase: ')
igual = True
i = 0
j = len(frase)-1
while igual and (i < j):
    igual = frase[i] == frase[j]
    i += 1
    j -= 1
if igual:
    print(frase, 'é um palíndromo!' )
else:
    print(frase, 'NÃO é um palíndromo!' )

# ou ...
frase = input('Digite uma frase: ')
if frase == frase[::-1]:
    print(frase, 'é um palíndromo!' )
else:
    print(frase, 'NÃO é um palíndromo!' )
```

C

```
/*
    PALINDROMO.C
    Verifica se uma cadeia de
    caracteres contém um palíndromo.
*/
#include<stdio.h>
#include<string.h>
#define N 100
int main(void)
{
    char frase[N];
    int i, j, igual;
    printf("Digite uma frase: ");
    setbuf(stdin, 0);
    fgets(frase, N, stdin);
    if (frase[strlen(frase)-1] == '\n')
        frase[strlen(frase)-1] = '\0';
    igual = 1;
    i = 0;
    j = strlen(frase) - 1;
    while( igual && (i < j) )
    {
        igual = frase[i] == frase[j];
        i++;
        j--;
    }
    if( igual )
        printf( "%s e um palindromo!\n", frase );
    else
        printf( "%s NAO e um palindromo!\n",
                frase );
    return 0;
}
```

Cadeias de caracteres

Python

```
"""
    PALINDROMO.PY
    Verifica se uma cadeia de
    caracteres contém um palíndromo.
"""

frase = input('Digite uma frase: ')
igual = True
i = 0
j = len(frase)-1
while igual and (i < j):
    igual = frase[i] == frase[j]
    i += 1
    j -= 1
if igual:
    print(frase, 'é um palíndromo!' )
else:
    print(frase, 'NÃO é um palíndromo!' )

# ou ...
frase = input('Digite uma frase: ')
if frase == frase[::-1]:
    print(frase, 'é um palíndromo!' )
else:
    print(frase, 'NÃO é um palíndromo!' )

# ou ...
frase = input('Digite uma frase: ')
print('%s%s é um palíndromo!' % \
  (frase, \
   '' if frase == frase[::-1] else " NÃO"))
```

C

```
/*
    PALINDROMO.C
    Verifica se uma cadeia de
    caracteres contém um palíndromo.
*/
#include<stdio.h>
#include<string.h>
#define N 100
int main(void)
{
    char frase[N];
    int i, j, igual;
    printf("Digite uma frase: ");
    setbuf(stdin, 0);
    fgets(frase, N, stdin);
    if (frase[strlen(frase)-1] == '\n')
        frase[strlen(frase)-1] = '\0';
    igual = 1;
    i = 0;
    j = strlen(frase) - 1;
    while( igual && (i < j) )
    {
        igual = frase[i] == frase[j];
        i++;
        j--;
    }
    if( igual )
        printf( "%s e um palindromo!\n", frase );
    else
        printf( "%s NAO e um palindromo!\n",
                frase );
    return 0;
}
```

Funções

Python

```python
"""
    FUN.PY
    Função para calcula o máximo divisor comum
    entre dois números inteiros.
"""

def mdc(r1, r2):
    while r1 != r2:
        if r1 > r2:
            r1 = r1 - r2
        else:
            r2 = r2 - r1;
    return r1




a = int(input('Digite o valor de a: '))
b = int(input('Digite o valor de b: '))
print ('O máximo divisor comum de %i e %i é %i' \
         % (a, b, mdc(a,b)))
```

C

```c
/*
    FUN.C
    Função para calcula o máximo divisor comum
    entre dois números inteiros.
*/
#include<stdio.h>
int mdc( int r1, int r2)
{
    while( r1 != r2 )
        if( r1 > r2 )
            r1 = r1 - r2;
        else
            r2 = r2 - r1;
    return r1;
}
int main(void)
{
    int a, b;
    printf("Digite o valor de a: ");
    scanf("%d", &a);
    printf("Digite o valor de b: ");
    scanf("%d", &b);
    printf("O maximo divisor comum de %d e %d e %d\n",
            a, b, mdc(a,b) );
    return 0;
}
```

Procedimentos

Python

```python
"""
    PROC.PY
    Procedimento que lê uma lista de
    tamanho N e procedimento que
    exibe a lista
"""


def lerLista(L, num):
    for i in range(num):
        valor = float(input('Valor %i: ' % i))
        L.append(valor)




def exibirLista(L):
    for i in range(len(L)):
        print('v[%i]= %4.2f' % (i, L[i]))




N = int(input("Digite o número de elementos: "))
L = []
lerLista(L, N)
exibirLista(L)
```

C

```c
/*
    PROC.C
    Procedimento que lê um vetor de
    tamanho N e procedimento que
    exibe o vetor
*/
#include<stdio.h>

void lerVetor( float v[], int num )
{
    int i;
    for( i=0; i<num; i++)
    {
        printf("Valor %d: ", i);
        scanf("%f", &v[i]);
    }
}

void exibirVetor( float v[], int num )
{
    int i;
    for( i=0; i<num; i++)
        printf("v[%d]= %4.2f\n", i, v[i]);
}

int main(void)
{
    int N;
    printf("Digite o número de elementos: ");
    scanf("%i", &N);
    float v[N];
    lerVetor(v, N);
    exibirVetor(v, N);
    return 0;
}
```

Registros

Python

```
"""
    COMPLEXO.PY
    Representação de números
    complexos com dicionários.
"""



c = {'re' : 0.0, 'im' : 0.0}









c['re'] = float(input('Digite a parte real: '))
c['im'] = float(input('Digite a parte imaginária: '))
if c['im'] > 0:
    sinal = '+'
else:
    sinal = '-';
if c['im'] != 0:
    print('Seu número é: %4.2f%ci%4.2f' \
                % (c['re'], sinal, abs(c['im'])))
else:
    print('Seu número é: %4.2f' % c['re'])
```

C

```
/*
    COMPLEXO.C
    Representação de números
    complexos.
*/
#include<stdio.h>
#include<math.h>
struct NumeroComplexo
{
    float re,im;
};

int main(void)
{
    struct NumeroComplexo c;
    printf("Digite a parte real: ");
    scanf("%f", &c.re);
    printf("Digite a parte imaginaria: ");
    scanf("%f", &c.im);
    if( c.im != 0 )
        printf("Seu numero e: %4.2f%ci%4.2f\n",
          c.re,((c.im > 0)?'+':'-'), fabs(c.im));
    else
        printf("Seu numero e: %4.2f\n", c.re );
    return 0;
}
```

Registros

Python – outra solução

```
"""
    COMPLEXO.PY
    Representação de números
    complexos com dicionários.
"""



c = {'re' : 0.0, 'im' : 0.0}









c['re'] = float(input('Digite a parte real: '))
c['im'] = float(input('Digite a parte imaginária: '))

if c['im'] != 0:
    print('Seu número é: %4.2f%ci%4.2f' \
          % (c['re'], '+' if c['im'] >0 else '-', \
                                        abs(c['im'])))
else:
    print('Seu número é: %4.2f' % c['re'])
```

C

```
/*
    COMPLEXO.C
    Representação de números
    complexos.
*/
#include<stdio.h>
#include<math.h>
struct NumeroComplexo
{
    float re,im;
};

int main(void)
{
    struct NumeroComplexo c;
    printf("Digite a parte real: ");
    scanf("%f", &c.re);
    printf("Digite a parte imaginaria: ");
    scanf("%f", &c.im);
    if( c.im != 0 )
        printf("Seu numero e: %4.2f%ci%4.2f\n",
          c.re,((c.im > 0)?'+':'-'), fabs(c.im));
    else
        printf("Seu numero e: %4.2f\n", c.re );
    return 0;
}
```

Referências

BACKES, **A. Linguagem C - Completa e Descomplicada**. Rio de Janeiro (RJ): Elsevier, 2013.

KERNIGHAN, B. W.; RITCHIE, D. M. **The C Programming Language**. Englewood Cliffs, N.J.: Prentice Hall, 1988.

SEBESTA, R. W. **Concepts of Programming Languages**. 10. ed. Boston: Pearson, 2012.

SIMON, H. A. The architecture of complexity. **Proceedings of the American philosophical society**, v. 106, n. 6, p. 467–482, 1962.

WIRTH, N. Program development by stepwise refinement. **Communications of the ACM**, v. 14, n. 4, p. 221–227, 1 abr. 1971.