

EEL270 – Computação II

2021/1

Prof. Marcelo Luiz Drumond Lanza

Objetivos

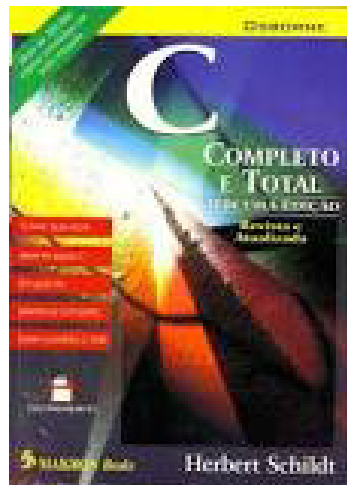
- Apresentar a linguagem C procedural
 - Exemplos
 - Boas práticas de programação
- Programas portáveis
 - Uso de diferentes compiladores/linkeditores
 - clang x gcc
 - Multiplataformas
 - Programas que possam ser executados em máquinas com diferentes sistemas operacionais
 - FreeBSD (*loghost02*)
 - Linux (*loghost03/loghost04*)
 - MAC-OS
 - Windows (*usando o Cygwin e no caso do Windows 10, o Linux Bash Shell*).
- Programação de CGIs (Common Gateway Interface)

Ementa

- Linguagem C procedural
- Boas práticas de programação
- Arquivos de dependências
- Ferramentas de controle de versão
- Programação portátil
 - Uso de diferentes compiladores/linkeditores (clang, gcc, ...)
 - Uso de diferentes versões de compiladores/linkeditores
 - Uso de diferentes dialetos da linguagem C (ANSI/C90, C89, C99 e C11)
 - Multiplataforma
- Programação de CGIs (*Common Gateway Interfaces*)
- Uso da biblioteca *Ncurses* (a confirmar)

Bibliografia

- Schildt, Herbert, “C Completo e Total”, Makron Books, 3ª Edição, 1997.
- Gay, Warren W., “Programação para Linux - Aprenda em 24 horas”, Editora Campus, 1ª Edição, 1999.



Dúvidas

- Dúvidas deverão ser enviadas para o grupo do *Whatsapp*.
 - Link para inscrição no grupo:
<https://chat.whatsapp.com/Hnp2xIJmSSBHlpKFbWrTk8>
- Horários de Atendimento
 - Segunda-feira – 13:00-17:00
 - Terça-feira – 10:00-12:00
 - Quarta-feira – 08:00-12:00 e 13:00-17:00
 - Quinta-feira – 08:00-12:00 e 13:00-17:00
 - Sexta-feira - 08:00-12:00 e 13:00-17:00
- ❖ Dúvidas relacionadas ao conteúdo da disciplina enviadas diretamente para o meu número não serão respondidas.

Monitoria

- Monitor:
 - Pedro Lídio da Silva
- Horários de Monitoria
 - Segunda-feira – 14:00-17:00
 - Terça-feira – 13:00-15:00
 - Quarta-feira – 15:00-17:00
 - Quinta-feira – 13:00-15:00
 - Sexta-feira - 10:00-11:00 e 15:00-17:00
- Plataforma
 - Meet

Aulas Teóricas e Práticas

- As aulas teóricas serão síncronas através da sala de aulas do Google (via POLI).
- As aulas teóricas serão gravadas, mas não serão disponibilizadas.
- Todos o código-fonte gerado durante as aulas teóricas ficará disponível no portal DEL (na área correspondente à disciplina/turma) e no diretório:

~marcelo.lanza/public/EEL270/2021-1/Aulas-Teoricas

- Durante as aulas práticas, a sala de aulas do Google permanecerá aberta (pra uso se necessário).
- Dúvidas deverão ser enviadas para o grupo do *Whatsapp*.

Dúvidas relacionadas com o conteúdo da disciplina enviadas diretamente para o meu número não serão respondidas.

Critérios de Avaliação

- Atividades referentes às aulas práticas.
 - 10 aulas práticas com pesos variando de 1 a 3
 - P1 a P3 – peso 1
 - P4 a P7 – peso 2
 - P8 a P10 – peso 3
- Avaliação online durante as aulas práticas. Alunos escolhidos aleatoriamente. Presença obrigatória.
- Se Média $\geq 5,0$ – Aprovado
Se Média $< 5,0$ - Reprovado

Correção das Aulas Práticas

- Correção em 2 etapas usando scripts.
 - 1ª Etapa
 - Verificação da entrega dos arquivos solicitados.
 - Compilação e *linkedição* dos itens solicitados, utilizando os arquivos de dependências que deverão ser escritos e mantidos pela(o) aluna(o).
 - Nos casos em que os arquivos de dependência não incluam as informações solicitadas nos roteiros (exatamente como definidas nestes roteiros) e não for possível completar com sucesso os procedimentos de compilação, linkedição e/ou execução de um item, a nota correspondente a este item será **ZERO**.
 - 2ª Etapa
 - Somente para os itens que foram compilados/linkeditados com sucesso.
 - Verificação do uso das boas práticas de programação apresentadas durante as aulas teóricas.
 - Verificação da execução dos binários (executáveis) solicitados nos roteiros.
- Entrega no prazo definido para cada item.
 - Itens entregues fora do prazo, terão um fator de redução correspondente ao número de dias em atraso.

Correção das Aulas Práticas

- Caso sejam detectadas 2 ou mais cópias de um determinado código-fonte, as notas deste item, para todos os alunos envolvidos, serão iguais a zero, independente de quem foi o autor e de quem copiou.

Portal DEL - Área da Disciplina

The screenshot displays the 'Portal do Departamento de Engenharia Eletrônica e de Computação' (DEL) website. The browser address bar shows the URL: https://www.del.ufrj.br/introducao/view?set_language=pt-br. The page features a green header with the department's logo and name. Below the header is a navigation menu with links such as 'página inicial', 'equipe', 'atividades', 'localização', 'contatos', 'área de membros', 'pastas das disciplinas', 'tutoriais', 'novidades', and 'eventos'. A search bar is located on the right side of the header.

The main content area is titled 'Introdução' and includes a 'Histórico' section. The text describes the course offered by the Department of Electronic and Computer Engineering (DEL/POLI/UFRJ) and mentions the participation of students in various engineering courses. The page also features a sidebar on the left with a 'navegação' section and a 'alterações recentes' section. The right sidebar contains a 'lista de revisão' and a 'noticias' section.

Introdução
por admin última modificação 12/07/2008 01:03 [Histórico](#)

O Departamento de Engenharia Eletrônica e de Computação da [Escola Politécnica](#) da [Universidade Federal do Rio de Janeiro](#) (DEL/POLI/UFRJ) oferece o curso de graduação em Engenharia Eletrônica e de Computação, além de participar ativamente nos cursos de Engenharia de Computação e Informação e Engenharia de Controle e Automação.

O DEL tem, sob a responsabilidade de seus professores, disciplinas em cursos de pós-graduação da COPPE e em cursos de pós-graduação (*lato-sensu*) da Escola Politécnica.

O Departamento atende a cerca de 800 alunos matriculados nos cursos de Engenharia Eletrônica e de Computação, Engenharia de Computação e Informação e Engenharia de Controle e Automação.

alterações recentes

- Aulas Práticas 24/08/2020
- Aulas Teóricas 24/08/2020
- Horários de Monitoria 24/08/2020
- Critérios de Avaliação 24/08/2020
- Bibliografia 24/08/2020

lista de revisão

- mario.machado 24/06/2009
- Página Pessoal 22/02/2012
- joarez.monteiro 17/09/2013
- alessandro.peixoto 21/08/2014
- Lista completa para revisão...

noticias

- Horário 2017/2 31/07/2017
- Acesso às redes cabeada e sem fio 01/02/2014
- Ex-aluna do DEL recruta desenvolvedor Ruby on Rails para empresa selecionada pelo Startup Brasil.

Portal DEL - Área da Disciplina



Portal do Departamento de Engenharia
Eletrônica e de Computação

mapa do site | acessibilidade | contato | configuração do site

[página inicial](#) | [equipe](#) | [atividades](#) | [localização](#) | [contatos](#) | [área de membros](#) | [pastas das disciplinas](#) | [tutoriais](#) | [novidades](#) | [eventos](#)

[Marcelo Luiz Drumond Lanza](#) | [minha pasta](#) | [downloads](#) | [preferências](#) | [desfazer](#) | [sair](#)

você está aqui: [página inicial](#) → [pastas das disciplinas](#) → [eel270 - computação ii](#) → [turma 2020-1](#)

navegação

- [Página Inicial](#)
- [Equipe](#)
- [Atividades](#)
- [Localização](#)
- [Contatos](#)
- [Área de Membros](#)
- [Pastas das Disciplinas](#)
- [EEL170 - Computação I](#)
- [EEL270 - Computação II](#)
- [EEL670 - Linguagens de Programação](#)
- [EEL875 - Internet e Arquitetura TCP/IP](#)
- [EEL878 - Redes de Computadores I](#)
- [EEL879 - Redes de Computadores II](#)

conteúdo | visão | edição | propriedades | compartilhamento

traduzir para | ações | exibição | adicionar item | estado: publicado

Turma 2020-1

Ir um nível acima

Contém informações úteis para os alunos inscritos em Computação II - turma 2020/1 (incluindo objetivos, ementa, bibliografia, critérios de avaliação, calendário de provas, horários de monitoria, frequência das aulas teóricas, frequência das aulas práticas, planilha de notas, roteiros das aulas práticas e roteiro do trabalho final).

Objetivos — por [Marcelo Luiz Drumond Lanza](#) — última modificação 30/11/2020 19:58
Apresenta de forma sucinta os objetivos da disciplina Computação II - turma 2020/1.

Ementa — por [Marcelo Luiz Drumond Lanza](#) — última modificação 30/11/2020 19:59
Define a ementa da disciplina Computação II - turma 2020/1.

Bibliografia — por [Marcelo Luiz Drumond Lanza](#) — última modificação 30/11/2020 20:12
Sugestões de bibliografia para a disciplina Computação II - turma 2020/1.

Críticos de Avaliação — por [Marcelo Luiz Drumond Lanza](#) — última modificação 30/11/2020 20:27
Define os critérios de avaliação da disciplina Computação II - turma 2020/1.

Horários de Monitoria — por [Marcelo Luiz Drumond Lanza](#) — última modificação 30/11/2020 20:33
Horários e locais de monitoria da disciplina Computação II - turma 2020/1.

Aulas Teóricas — por [Marcelo Luiz Drumond Lanza](#) — última modificação 30/11/2020 20:35
Contém os exemplos utilizados nas aulas teóricas e a planilha com a frequência correspondente.

Aulas Práticas — por [Marcelo Luiz Drumond Lanza](#) — última modificação 30/11/2020 20:35
Contém os roteiros referentes às aulas práticas e a planilha com a frequência correspondente.

Dezembro 2020

| Do | Se | Te | Qu | Qu | Se | Sa |
|----|----|----|----|----|----|----|
| | | 1 | 2 | 3 | 4 | 5 |
| 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 20 | 21 | 22 | 23 | 24 | 25 | 26 |
| 27 | 28 | 29 | 30 | 31 | | |

lista de revisão

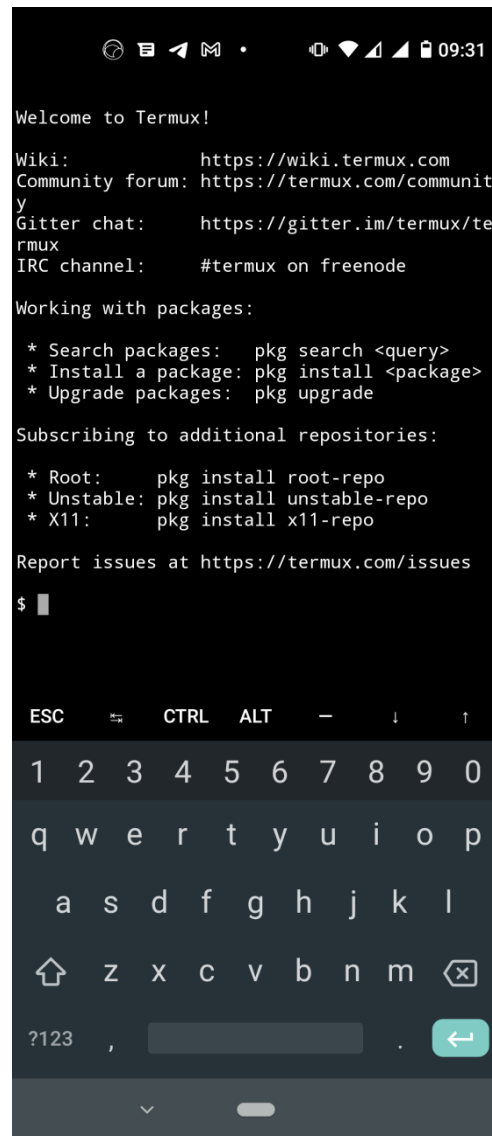
- [mario.machado](#)
24/06/2009
- [Página Pessoal](#)
22/02/2012
- [joarez.monteiro](#)
17/09/2013
- [alessandro.peixoto](#)
21/08/2014

Lista completa para revisão...

notícias

- [Horário 2017/2](#)
31/07/2017
- [Acesso às redes](#)
aberto a com fa

Termux



```
Welcome to Termux!

Wiki:      https://wiki.termux.com
Community forum: https://termux.com/community
Gitter chat:  https://gitter.im/termux/termux
IRC channel:  #termux on freenode

Working with packages:

* Search packages:  pkg search <query>
* Install a package: pkg install <package>
* Upgrade packages: pkg upgrade

Subscribing to additional repositories:

* Root:    pkg install root-repo
* Unstable: pkg install unstable-repo
* X11:     pkg install x11-repo

Report issues at https://termux.com/issues

$ █
```

ESC FN CTRL ALT - ↓ ↑

1 2 3 4 5 6 7 8 9 0

q w e r t y u i o p

a s d f g h j k l

⏮ z x c v b n m ⏭

?123 , . ←

Comandos Unix

comando [-opções_curtas [argumento]] [--opções_longas [argumentos]] [argumentos]

- `man`
 - exibe a página de manual do comando/função
- `clear`
 - limpa o terminal
- `pwd`
 - exibe o caminho absoluto da diretório atual
- `hostname`
 - exibe o nome da máquina
- `uname`
 - exibe informações sobre o sistema operacional
- `cd`
 - troca o diretório de trabalho
- `mkdir`
 - cria um ou mais diretórios.
- `rmdir`
 - Remove um ou mais diretórios (se vazios)
- `ls`
 - Lista o conteúdo de um ou mais diretórios
- `cp`
 - copia um ou mais itens do sistema de arquivos
- `rm`
 - Remove um ou mais itens do sistema de arquivos
- `mv`
 - Move e/ou renomeia um ou mais itens do sistema de arquivos
- `chmod`
 - Altera as permissões de um ou mais itens do sistema de arquivos (r – leitura, w – escrita e x – execução)

Comandos Screen

- **Ctrl + a c**
 - Cria um novo terminal virtual
- **Ctrl + a A**
 - Permite editar o nome do terminal
- **Ctrl + a “**
 - Lista os terminais existentes
- **Ctrl + a p**
 - Muda para o terminal anterior
- **Ctrl + a n**
 - Muda para o próximo terminal
- **Ctrl + a Dígito**
 - Muda para o terminal cujo número é dado por dígito
- **Ctrl + a d**
 - Desconecta o screen.
- **screen -r**
 - Reconecta o screen
- **screen -r -d**
 - Reconecta o screen

Comandos vim

- ESC i ou ESC INSERT
 - Muda para modo edição.
- ESC :w
 - Salva o arquivo.
- ESC :q
 - Encerra a edição (somente se o arquivo não tiver alterações pendentes para salvar).
- ESC :q!
 - Encerra a edição descartando possíveis alterações.
- ESC :wq ou ESC:x
 - Salva o arquivo e encerra a edição.
- ESC /STRING
 - Procura por STRING no conteúdo do arquivo.
- ESC dd
 - Apaga a linha atual (sob o cursor)
- ESC yy
 - Copia a linha atual.
- ESC p
 - Cola o que foi copiado a partir da linha abaixo.
- ESC P
 - Cola o que foi copiado a partir da linha acima.
- ESC r
 - Substituir o caractere atual.
- Os comandos acima podem usar um número representando repetição, por exemplo:
- ESC 10 dd
 - Apaga 10 linhas a partir da linha atual.

Linguagem C - Tipos

- Tipos básicos
 - void
 - char
 - int
 - float
 - double
- Modificadores de sinal
 - signed
 - unsigned
- Modificadores de largura
 - short
 - long
 - long long
- Tipos básicos podem ser combinados com modificadores de sinal e/ou com modificadores de largura.
- Modificadores de sinal, modificadores de largura e as combinações entre eles sem um tipo básico correspondem às combinações com o tipo **int**.
- Ordem de declaração não importa.
- unsigned int == int unsigned == unsigned
- unsigned long long == int long long unsigned
- sizeof (tipo) ou sizeof (variável) corresponde ao tamanho em bytes do tipo/variável em questão.

Linguagem C - Funções

- Cabeçalho da Função
Corpo da Função
- Cabeçalho da Função:
TIPO_DA_FUNÇÃO NOME_DA_FUNÇÃO (ARGUMENTOS_DA_FUNÇÃO)
- Corpo da Função:

```
{  
    DECLARAÇÕES;  
}
```
- Protótipo da Função versus Cabeçalho da Função:
 - Protótipo **não precisa** incluir os identificadores dos argumentos da função.
 - Cabeçalho **precisa** incluir os identificadores dos argumentos da função.
 - Protótipo **precisa** incluir ponto e vírgula no final da declaração.

Linguagem C - Funções

- ```
int /* int */
main (int argc, char *argv []) /* main (int argc, char **argv) */
{
 declarações;
}
```
- Tipo da função: **int**
  - Função deve retornar um valor do tipo int.
  - Valor retornado para quem executou a função (sistema operacional, servidor web, etc.)
  - 0 (ZERO) representa sucesso e diferente de ZERO representa erro.
- Nome da função: **main**
- Tipos dos argumentos da função: **int** e **char \* [ ]**.
  - char \* - tipo definido pelo usuário, lê-se ponteiro para char e é uma dentre as duas possíveis maneiras de se definir uma **string**.
  - tipo [ ] – vetor/matriz unidimensional
  - Se char \* corresponde a uma **string**, **char \* [ ]** corresponde a **string [ ]**, ou seja, é um vetor unidimensional de strings.
- Nomes dos argumentos da função: **argc** (contador de argumentos) e **argv** (vetor de argumentos).

# Identificadores – Estilos

- *Snake Case*
  - letras maiúsculas, dígitos e o caractere sublinhado
  - macros e constantes
  - *COMPRIMENTO\_MAXIMO\_NOME*
- *Camel Case*
  - letras minúsculas, com exceção da primeira letra de cada palavra (a partir da segunda palavra) que compõe o identificador
  - tipos, membros de tipos enumerados e variáveis
  - *comprimentoNomeCompleto*
- *Pascal Case*
  - letras minúsculas, com exceção da primeira letra de cada palavra que compõe o identificador
  - Funções
  - *CalcularFatorial*
- Todos os identificadores podem conter dígito, mas nenhum identificador pode começar com dígito.
- Todos os identificadores devem ser significativos e sem abreviações.
- O identificador de uma função deve começar com um verbo no infinitivo.

# Sistemas de Controle de Versão

- Primeira Geração
  - SCCS (*Source Code Control System*)
  - RCS (*Revision Control System*)
- Segunda Geração
  - CVS (*Concurrent Versions System*)
  - SVN (*Apache Subversion*)
  - Perforce Helix Core
- Terceira Geração
  - Git
  - Mercurial
  - BitKeeper
  - Darcs (*Darcs Advanced Revision Control System*)
  - Monotone
  - Bazaar
  - Fossil
  - Pijul

# RCS (*Revision Control System*)

- Criar o diretório que será usado como repositório
  - mkdir RCS
    - Nome padrão é RCS (letras maiúsculas)
- Submeter arquivos ao sistema de controle de versão
  - ci arquivo
    - Será criada uma cópia do arquivo no repositório com a extensão “.v”
    - Arquivos do repositório não devem ser manipulados sem o uso da ferramenta.
  - Ci -u arquivo
    - Será criando uma cópia do arquivo no repositório com a extensão “.v” e será mantida uma cópia somente de leitura no diretório atual. Corresponde à execução seguida dos comandos ci e co.
- Obter uma cópia do arquivo submetido ao RCS
  - co [release] arquivo
    - Cópia somente de leitura
  - co -l [release] arquivo
    - Cópia de escrita: -l (lock)

# Arquivos de Dependências

- Macros

CC = gcc

CC = /opt/intel/bin/icc

LD = clang

CFLAGS = -Wall -ansi

LFLAGS = -Wall

EXEMPLO01OBS = exemplo\_001.o

EXECS = exemplo\_001

LIBS = libmatematica.a

ALL = \$(EXECS) \$(LIBS)

# Arquivos de Dependências

- Objetivos

Objetivo: Dependências  
**TAB**comandos

- Regra Implícita

.c.o:

`$(CC) $(CFLAGS) -c $<`

`exemplo_001: exemplo_001.o`

`gcc -Wall -ansi -o exemplo_001 exemplo_001.o`

`exemplo_001: exemplo_001.o`

`gcc -Wall -ansi exemplo_001.o -o exemplo_001`

`exemplo_001: $(EXEMPLO01OBJS)`

`$(LD) $(LFLAGS) -o $@ $(EXEMPLO01OBJS)`



# Arquivos de Dependências

- CentOS
  - GNU Make
  - GNUMakefile
- FreeBSD
  - comando nativo
  - BSDmakefile
  - o comando *gmake* poderia ser utilizado

# Arquivos de Dependências

- Prioridade
  - GNUmakefile/BSDmakefile
  - makefile
  - Makefile
  - OutroNome
    - make -f OutroNome

# Arquivos de Dependências

- Rótulos Genéricos
  - all
  - clean
  - clean-all
  - clean-objs
  - clean-centos
  - clean-freebsd
  - clean-ansi
  - clean-c89[
  - clean-c99
  - clean-c11
  - clean-gcc
  - clean-clang

# Arquivos de Dependências

- Rótulos Especias (Built-In)

- .PHONY

- .PHONY: clean

- clean:

- rm -f \*.o \$(ALL)

# Arquivos de Dependências

- Passagem de argumentos para o comando make
  - CentOS
    - make cc=CLANG
  - FreeBSD
    - make -DCLANG
    - make -DCLANG=clang
    - make CLANG=clang
    - make CLANG=1

# Arquivos de Dependências

- Verificação de valores de argumentos

- CentOS

- ifeq (\$(cc), CLANG)
    - CC = clang
    - else
    - CC = /opt/intel/bin/icc
    - endif

- FreeBSD

- .ifdef CLANG
    - CC = clang
    - .else
    - CC = icc
    - .endif

# Arquivos de Dependências

- Obtendo o nome do sistema operacional
  - CentOS  
OS = `$(shell uname -s)`
  - FreeBSD  
OS = ``uname -s``

# Palavras-Chaves

main

void

int

char

float

double

unsigned

signed

short

long

return

include

define

if

for

do

while

switch

case

break

default

continue

sizeof



- Código de Retorno
  - 0 = Sucesso
  - Dif. de 0 = Erro
  - (Computação II – sempre que possível  $> 0$ )
- Valores booleanos
  - 0 = Falso
  - Dif. de 0 = Verdadeiro

# Estruturas Condicionais

- **if** (expressão)                      if ();  
    {  
        declarações;  
    }  
    else  
    {  
        declarações;  
    }

# Estruturas Condicionais

- Expressão ? DeclaraçãoVerdadeiro : DeclaraçãoFalso
- letra == 'A' ? printf ("Eh a vogal A") : printf ("Não eh a vogal A");
- if (letra == 'A')  
    printf ("Eh a vogal A");  
else  
    printf ("Não eh a vogal A");

# Estruturas Condicionais

- switch (expressão do tipo inteiro)

```
{
 case 'A':
 case 'a':
 declarações1;
 break;
 case '?':
 declarações2;
 break;
 ...
 default:
 declaraçõesN;
}
```

```
if (letra == 'A' || letra == 'a')
 declarações1;
else if (letra == '?')
 declarações2;
else if ...

else
 declaraçõesN;
```

# Estruturas de Repetição

- **do**  
  {  
    declarações;  
  }  
**while** (expressão-condição);
- `int indice = 0;`  
  `do`  
  {  
    `printf ("%i\n", indice);`  
    `indice++;`  
  }  
  `while (indice < 10);`

# Estruturas de Repetição

- **while** (expressão-condição)  
{  
    declarações;  
}
- `int indice = 0;`  
`while (indice < 10)`  
{  
    `printf ("%i\n", indice);`  
    `indice++;`  
}

# Estruturas de Repetição

- `for (inicialização; expressão-condição; pós-ação)`  
    {  
        declarações;  
    }
- `int indice;`  
  `for (indice = 0; indice < 10; ++indice)`  
    `printf ("%i\n", indice);`
- `int indice = 0;`  
  `for ( ; indice < 10; indice++)`  
    `printf ("%i\n", indice);`

# Estruturas de Repetição

- `for (;;);`
  - Laço de repetição infinito executando uma declaração vazia.



# Operadores Matemáticos

|    |                                    |                                   |        |
|----|------------------------------------|-----------------------------------|--------|
| =  | atribuição                         | int/int                           | → int  |
| +  | soma                               | real/real                         | → real |
| -  | subtração                          | int/real                          | → real |
| *  | multiplicação                      | real/int                          | → real |
| /  | divisão                            |                                   |        |
| %  | resto da divisão                   |                                   |        |
| ++ | incremento unitário                |                                   |        |
| -- | decremento unitário                |                                   |        |
| += | forma reduzida da soma             | soma = soma + 1/i; → soma += 1/i; |        |
| -= | forma reduzida da subtração        |                                   |        |
| *= | forma reduzida da multiplicação    |                                   |        |
| /= | forma reduzida da divisão          |                                   |        |
| %= | forma reduzida do resto da divisão |                                   |        |

# Operadores Lógicos

|    |                                                 |    |                       |
|----|-------------------------------------------------|----|-----------------------|
| == | comparação de igualdade                         | >> | shift para a direita  |
| != | diferente                                       | << | shift para a esquerda |
| >  | maior                                           |    |                       |
| <  | menor                                           | ~  | inverte os bits       |
| >= | maior ou igual                                  | &  | and bit a bit         |
| <= | menor ou igual                                  |    | or bit a bit          |
|    |                                                 | ^  | xor bit a bit         |
| !  | not lógico                                      |    |                       |
| && | and lógico                                      |    |                       |
|    | or lógico                                       |    |                       |
| *  | ponteiro para (na definição de um ponteiro)     |    |                       |
| *  | o conteúdo de (no uso de um ponteiro)           |    |                       |
| &  | o endereço de (uso do endereço de uma variável) |    |                       |