

Estruturas de Dados

- Prof. Edkallenn Lima
- edkallenn@yahoo.com.br (somente para dúvidas)
- Blogs:
 - <http://professored.wordpress.com> (Computador de Papel – O conteúdo da forma)
 - <http://professored.tumblr.com/> (Pensamentos Incompletos)
 - <http://umcientistaporquinzena.tumblr.com/> (Um cientista por quinzena)
 - <http://eulinoslivros.tumblr.com/> (Eu Li nos Livros)
 - <http://linabiblia.tumblr.com/> (Eu Li na Bíblia)
- Redes Sociais:
 - <http://www.facebook.com/edkallenn>
 - <http://twitter.com/edkallenn>
 - <https://plus.google.com/u/0/113248995006035389558/posts>
 - Pinterest: <https://www.pinterest.com/edkallenn/>
 - Instagram: <http://instagram.com/edkallenn> ou [@edkallenn](#)
 - LinkedIn: br.linkedin.com/in/Edkallenn
 - Foursquare: <https://pt.foursquare.com/edkallenn>
- Telefones:
 - 68 8401-2103 (VIVO) e 68 3212-1211.
- Os exercícios devem ser enviados SEMPRE para o e-mail: edkevan@gmail.com ou para o e-mail: edkallenn.lima@uninorteac.edu.br



Apresentação

- Nome
- Formação
- Local de trabalho ou função
- Experiência na área de informática (ou programação)



Metodologia/Abordagem

- Aulas expositivas
- Debates
- Totalmente práticas
- TODAS as aulas em laboratório
- Programação em **TODAS** as aulas com a abordagem *Live Code*
- Resolução de exercícios (muitos exercícios)



Avaliação

- 40% Trabalhos e exercícios + 60% Prova
- Data da nova NI: Cf. calendário
- Data da N2: Cf. calendário
- Data da 2^a chamada: Cf. calendário
- Final: Cf. calendário
- Calendário normal da faculdade



Importante

- Empenho
- Estudo
- Dedicação
- Base sólida = futuro promissor



Ementa/Plano de Curso

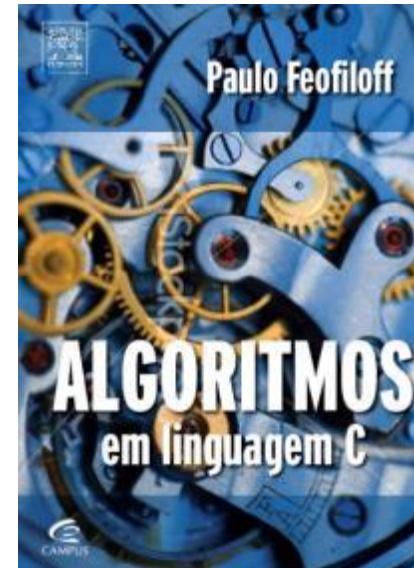
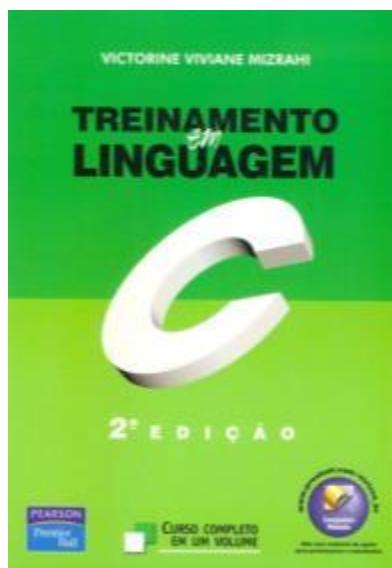
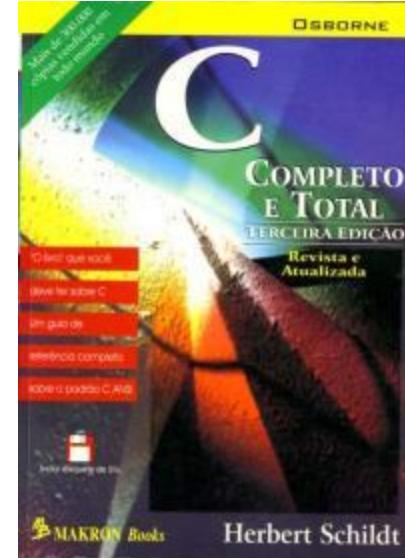
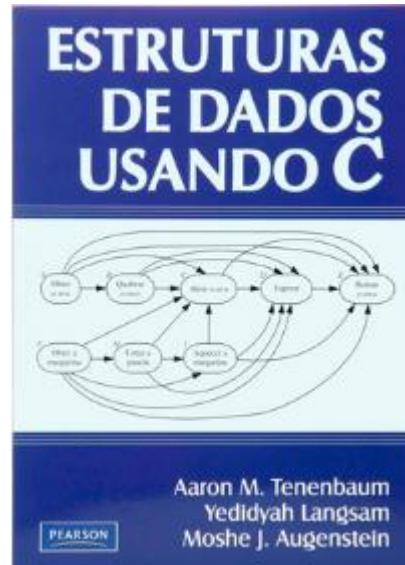
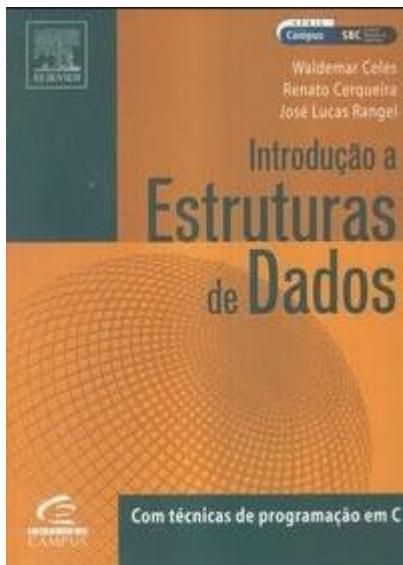
- Introdução a Estrutura de Dados
 - Visão geral da programação em C
 - Manipulação de Dados (tipos, constantes e variáveis). Atribuição. Blocos de instruções. Entrada e Saída. Condicionais. Repetição. Funções. Recursividade. Ponteiros
 - Visão geral da programação em Python (OPCIONAL)
 - Manipulação de Dados (tipos, constantes e variáveis). Atribuição. Blocos de instruções. Entrada e Saída. Condicionais. Repetição. Funções. Recursividade.
- Estruturas de Dados homogêneas
 - Vetores. Inserção e Busca. Remoção. Vetores bidimensionais. Vetores multidimensionais. Strings.
- Estruturas de Dados heterogêneas
 - Registros. Composição de registros com vetores.
- Arquivos
 - Sequenciais. Arquivos-texto. Arquivos de Registros.
- TAD (Tipo Abstrato de Dados)
 - Definições teóricas. Implementações.
- Análise e Complexidade de Algoritmos
 - Análise Assintótica. Notação O. Recorrências. Técnicas de Projeto de Algoritmos (método guloso, divisão e conquista e programação dinâmica)
- Busca/Ordenação
 - Busca Linear. Busca Binária. Ordenação (Bubble, Insertion, Selection, Merge e QuickSort)
- Estruturas de Dados dinâmicas
 - Listas. Pilhas. Filas. Tabelas Hash. Árvores e Dicionários.



Bibliografia

- CELES, Waldemar; CERQUEIRA, Renato; RANGEL, José Lucas. Introdução a Estruturas de dados. Rio de Janeiro: Campus, 2004. 250 p. (Coleção SBC)
- TENENBAUM, Aaron M.; LANGSAM, Yedidyah, AUGENSTEIN, Moshe. Estruturas de dados usando C. São Paulo: Makron Books, 1995. 904 p.
- SCHILDT, Herbert. C, Completo e total – 3^a. Edição revista e atualizada. São Paulo. Makron Books. 1996.
- MIZRAHI, Victorine Viviane. Treinamento em linguagem C. São Paulo. Pearson Education do Brasil. 1990.
- FEOFILOFF, Paulo. Algoritmos em Linguagem C. Rio de Janeiro. Elsevier. 2009.
- HOLZNER, Steven. Programando em C++: um guia prático para a programação profissional. Série de Programação Peter Norton. Rio de Janeiro. Campus, 1993.
- SCHILDT, Herbert. Turbo C++: Guia do usuário. São Paulo. Makron Books, McGraw-Hill, 1992.





Na Internet

- <http://www.numaboa.com.br/informatica/tutoriais/c>
- <http://www.juliobattisti.com.br/tutoriais/katia/duarte/cbasico001.asp>
- <http://www.icmc.usp.br/~sce182/>
- <http://www.dca.fee.unicamp.br/cursos/EA876/apostila/HTML/node10.html>
- <http://www.ebah.com.br/content/ABAAAAAVsAD/estruturas-dados-waldemar-celes>
- <http://www.slideshare.net/CriatividadeZeroDocs/introduo-a-estrutura-de-dados>



Revisão de Modularização

- [Clique para abrir a aula](#)
- Aula 0.



Estrutura de Dados?

- **Estrutura de dados** é o ramo da computação que estuda os diversos mecanismos de **organização** de **dados** para atender aos diferentes requisitos de **processamento**.
- As estruturas de dados definem a **organização**, **métodos** de **acesso** e opções de **processamento** para a **informação** manipulada pelo programa
- Na Ciência da computação, uma **estrutura de dados** é um modo particular de armazenamento e organização de dados em um computador de modo que possam ser usados eficientemente.



Por que estudar Estruturas de Dados?

- Diferentes tipos de estrutura de dados são adequadas a diferentes tipos de aplicação e algumas são altamente especializadas, destinando-se a algumas tarefas específicas.
- Por exemplo, as Árvores-B são particularmente indicadas para a implementação de bases de dados
- A implementação de compiladores geralmente requer o uso de tabela de dispersão para a busca de identificadores.



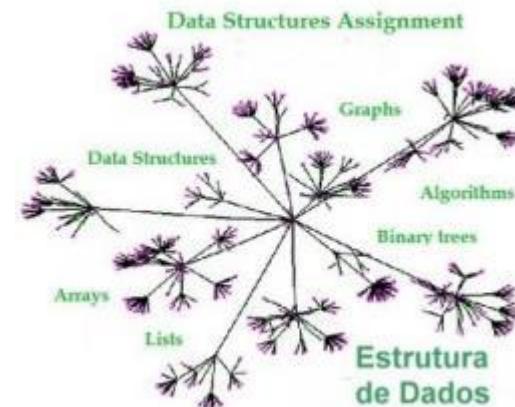
Estruturas de Dados

- Algoritmos manipulam **dados**.
- Quando estes dados estão organizados (dispostos) de forma coerente, caracterizam uma forma, uma **estrutura de dados**.
- A **organização** e os **métodos** para manipular essa estrutura é que lhe conferem singularidade.
- Ex: vetores, matrizes, listas, filas, pilhas, ávores, grafos, tabelas de dispersão, etc.



Tipos de Estruturas de Dados

- **Homogêneas** (vetores e matrizes)
- **Heterogêneas** (estruturas, registros)
- **Estáticas** – possuem tamanho fixo, definido.
- **Dinâmicas** (listas, filas, pilhas, árvores e grafos)





Venha C refrescar!
A água está
maravilhosa!



Estudo de C

- A linguagem C é uma das mais bem sucedidas linguagens de **alto nível** já criadas
- É considerada uma das linguagens de programação **mais utilizadas** de todos os tempos.
- Porém, muitas pessoas a consideram uma linguagem de **difícil aprendizado**, seja pela dificuldade de compreender certos conceitos (como os **ponteiros**), seja pela falta de clareza com que a estrutura da linguagem é descrita.



Estudo de C

- Neste curso, vamos “**descomplicar**” a Linguagem C
- Usaremos a IDE Code::Blocks, uma IDE de código aberto e multiplataforma que suporta múltiplos compiladores, entre eles a linguagem C.
- Teremos conhecimento **básico** e **avançado** de C, mas sem entrar nos meandros mais íntimos deste conhecimento.
- O Foco são as **estruturas de dados**



Estudo de C

- É uma das mais (senão a mais) popular **linguagem de alto nível** já criada.
- Define-se como linguagem de alto nível aquela que possui um nível de abstração relativamente elevado, que está mais próximo da linguagem humana do que do código de máquina.
- Ela foi criada em 1972, nos **laboratórios Bell**, por **Dennis Ritchie**.
- Foi revisada e padronizada pelo **ANSI** (American National Standards Institute) em 1989



Estudo de C

- Trata-se de uma linguagem **estruturalmente simples** e de grande **portabilidade**.
- Poucas são as arquiteturas de computadores para as quais não exista um **compilador C**.
- Além disso, o compilador da linguagem gera **códigos mais enxutos e velozes** do que muitas outras linguagens.



Estudo de C

- A linguagem C é uma linguagem **procedural**, ou seja, ela permite que um problema complexo seja facilmente decomposto em módulos, sendo cada **módulo** um problema mais simples.
- Além disso, ela fornece acesso de **baixo nível** à **memória**, o que permite o acesso e a programação direta do microprocessador.
- Ela também permite a implantação de programas utilizando instruções em **Assembly**, o que possibilita programar problemas em que a dependência do **tempo** é critica.



Estudo de C

- Por fim, a linguagem C foi criada para incentivar a programação **multiplataforma**.
- Ou seja, programas escritos em C podem ser compilados para uma grande variedade de plataformas e sistemas operacionais com apenas **pequenas alterações** no seu código-fonte.

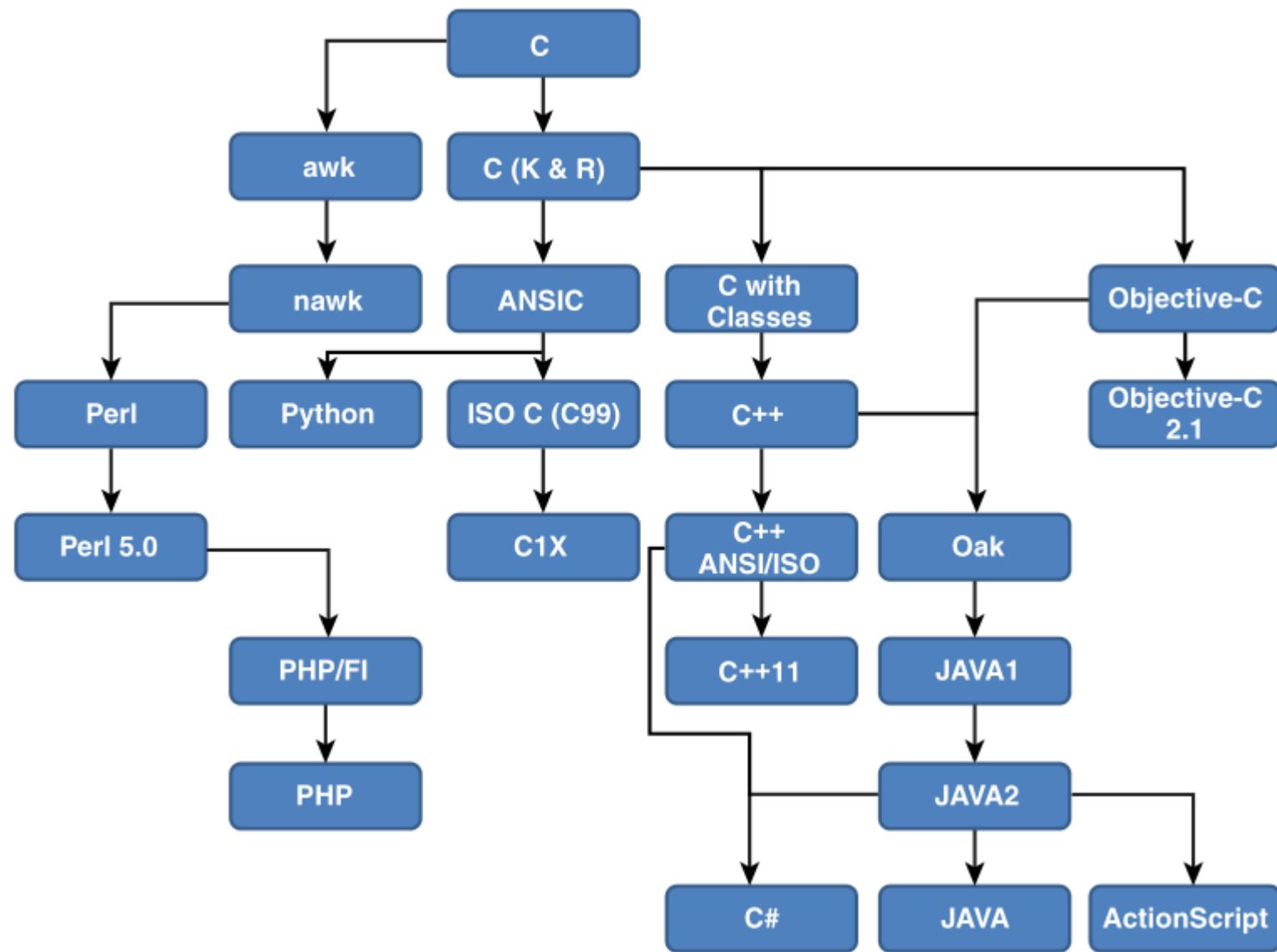


Influência de C

- A linguagem C **tem influenciado**, direta ou indiretamente, muitas linguagens desenvolvidas posteriormente, como **C++, Java, C# e PHP**.
- No próximo slide é possível ver uma breve história da evolução da linguagem C e de sua influência no desenvolvimento de outras linguagens de programação.



Influência de C



Influência de C

- Provavelmente a influência mais marcante da linguagem foi **a sua sintaxe**.
- Todas as linguagens mencionadas **combinam a sintaxe de declaração e a sintaxe da expressão da linguagem C** com sistemas de tipo, modelos de dados etc.
- O próximo slide mostra como um comando de impressão de números variando de 1 a 10 pode ser implementado em diferentes linguagens influenciadas por C



Influência de C

C

```
for(i = 1; i <= 10;i++)  
{  
    printf("%d\n",i);  
}
```

PHP

```
for ($i = 1; $i <= 10; $i++)  
{  
    echo $i;  
}
```

Java

```
for(int i=1;i<=10;i++)  
{  
    System.out.println(i);  
}
```

Perl

```
for($i = 1; $i<=10; $i++)  
{  
    print $i;  
}
```



Por que usar C?

- Código de **alta performance** para jogos, aplicativos de missão crítica?
- Arduino, FPGA, hardware livre?
- Criar/Usar **bibliotecas externas** em **apps** ou programas?
- Código **portável**?
- Uso de código de **baixo nível**, registradores ou **endereços** de memória?
- Velocidade, espaço e **portabilidade** são importantes?
- Programas **pequenos** e **rápidos**?



Padrões de C

- ANSI C → fim dos anos 1980, usado para código mais antigo
- C99 → 1999
- C11 → 2011
- Há grandes diferenças entre os padrões.
Vamos vê-las ao longo do caminho



Feb 2018	Feb 2017	Change	Programming Language	Ratings	Change
1	1		Java	14.988%	-1.69%
2	2		C	11.857%	+3.41%
3	3		C++	5.726%	+0.30%
4	5	▲	Python	5.168%	+1.12%
5	4	▼	C#	4.453%	-0.45%
6	8	▲	Visual Basic .NET	4.072%	+1.25%
7	6	▼	PHP	3.420%	+0.35%
8	7	▼	JavaScript	3.165%	+0.29%
9	9		Delphi/Object Pascal	2.589%	+0.11%
10	11	▲	Ruby	2.534%	+0.38%
11	-	▲	SQL	2.356%	+2.36%
12	16	▲	Visual Basic	2.177%	+0.30%
13	15	▲	R	2.086%	+0.16%
14	18	▲	PL/SQL	1.877%	+0.33%
15	13	▼	Assembly language	1.833%	-0.27%
16	12	▼	Swift	1.794%	-0.33%
17	10	▼	Perl	1.759%	-0.41%
18	14	▼	Go	1.417%	-0.69%
19	17	▼	MATLAB	1.228%	-0.49%
20	19	▼	Objective-C	1.130%	-0.41%



Estudo de C

- C é **estruturada**
- C é para programadores de verdade (true-hackers)
- É **compilada** e **fácil** (pequena quantidade de keywords)
- É **Padronizada** (ANSI C)
- Dennis Ritchie criou a C para desenvolver o Unix (porque dar manutenção em Assembly é o inferno!)
- É a linguagem que viabilizou a compilação de programas **complexos**, trabalhando em um ambiente extremamente otimizado, usando o máximo dos recursos das máquinas da época (as primeiras versões de C rodavam em um PDP-11)
- O **Unix** é o sistema operacional de verdade mais bem-sucedido até hoje, e C evoluiu como linguagem, gerando C++, Visual C, Objective C, ANSI C e muitas outras
- O **kernel** do iOS e do MacOS é o **Darwin**, um sistema Unix baseado no NEXSTEP e... Open Source.
- Pré-processador, macros, ponteiros, acesso a baixo nível, meta-programação, sobrecarga de operadores e outros recursos.





- Este é um **PDP-7**, um "minicomputador" da década de **1960** que possuía apenas **8 kbytes** de memória RAM (O Unix começou a ser desenvolvido em um computador como este)
- Utilizava **fitas magnéticas** para o armazenamento de dados.
- Hoje em dia, qualquer **agenda eletrônica** ou **celular** possui muito mais memória e poder de processamento do que ele,
- Mas na época era um equipamento relativamente poderoso, que custava **USS 72.000**.





- Dennis Ritchie (em pé) e Ken L. Thompson (sentado), inventores do Unix, no Bell Laboratories em frente a um PDP-11 em 1970

Por que estudar C?

06/08/2012 02h37 - Atualizado em 06/08/2012 10h38

Robô Curiosity pousa em Marte e Nasa comemora início da missão

Controladores da agência espacial festejaram pouso por 10 minutos. Jipe é o maior e mais moderno veículo já feito para explorar o planeta.

Do G1, com agências internacionais *

 [Tweetar](#) 707
 [Recomendar](#) 3,6 mil
409 comentários



PUBLICIDADE

clickOn



4 DIAS NO HOTEL
BRISTOL JANGADA
EM FORTALEZA
De R\$1.358
Por R\$679
QUERO!

Seus amigos no G1

veja o que eles estão lendo



[Conecte-se com Facebook](#)

Conecte-se com seus amigos e saiba o que eles estão lendo. [Veja mais](#)

Ciência e Saúde

veja tudo sobre >

Substância encontrada no sabonete pode danificar músculos, diz estudo

HÁ 2 HORAS

'Se contatarem marcianos me avisem imediatamente', diz...

HÁ 2 HORAS

Por que estudar C?



www.nasa.gov/mission_pages/msl/

informática Dicionários Links ANTIGUIDADES INF... CELULAR E SMARTP... cinema GAMES E EMULADOR JORNALIS MATEMATICA E CIE... WORDPR...

NEWS

News, features & press releases

MISSIONS

Current, future, past missions & launch dates

MULTIMEDIA

Images, videos, NASA TV & more

CONNECT

Social media channels & NASA apps

ABOUT NASA

Leadership, organization, budget, careers & more

Search

For Public | For Educators | For Students | For Media Send Share

Curiosity - Mars Science Laboratory

Curiosity: Could Mars Have Once Harbored Life?



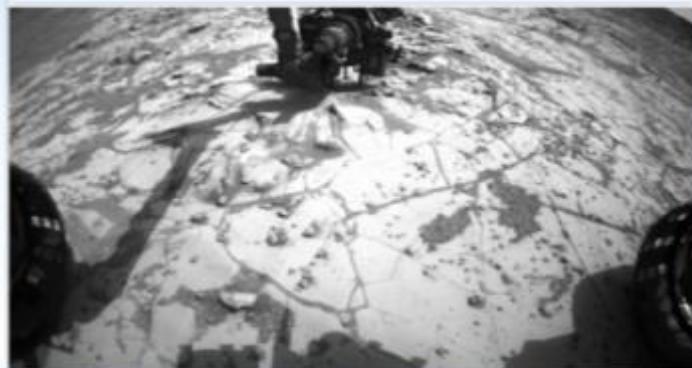
Images
Multimedia
News & Media Resources
Launch
[All NASA Missions](#)

Journey to Mars



Mars Science Laboratory

Latest News



Crystal-Rich Rock 'Mojave' is Next Mars Drill Target

This week, NASA's Curiosity Mars rover is preparing to drill its second sample of

Latest Videos

NASA | Need To Know: Sample Analysis at Mars Findings



NASA | Need To Know, Sampl... <

Curiosity Rover Report: The Making of Mount Sharp (Dec. 8, 2014)

Curiosity Rover Report: A Taste

witter.com...    Mars Science La...   Spotify  Estruturas de Da...  Apresentação d... PT ▶ 🔍



mars.jpl.nasa.gov/msl/

Informática Dicionários Links ANTIGUIDADES INF... CELULAR E SMARTP... cinema GAMES E EMULADOR JORNALIS MATEMATICA E CIE... WORDPRESS

NASA Jet Propulsion Laboratory California Institute of Technology

JPL HOME EARTH SOLAR SYSTEM STARS & GALAXIES SCIENCE & TECHNOLOGY
BRING THE UNIVERSE TO YOU: JPL Email News | RSS | Mobile | Video

Mars Science Laboratory Curiosity Rover

HOME MISSION NEWS MULTIMEDIA PARTICIPATE! SEARCH ALL MARS

FOLLOW YOUR CURIOSITY



See the Latest Raw Images
Want to see the latest images from Curiosity? Click here to see images by day (sol).
[See latest raw images >>](#)

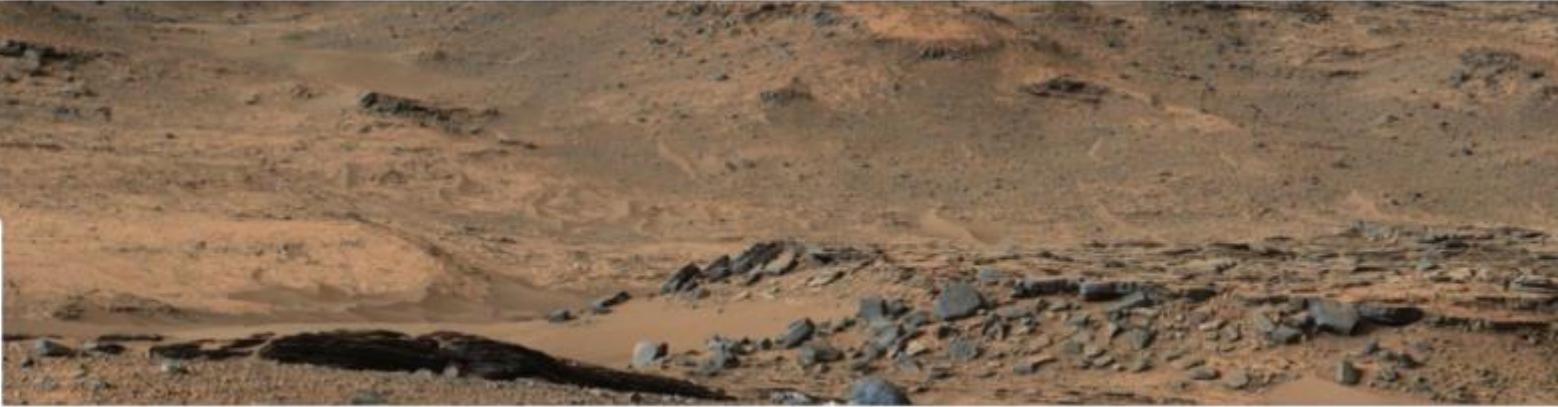
What's New? Recent Videos Fun Ask Dr. C Curiosity

Sistemas de Informação

https://twitter.com/marscuriosity

Informática Dicionários Links ANTIGUIDADES INF... CELULAR E SMARTP... cinema GAMES E EMULADOR JORNALIS MATEMATICA E CIE... WORDPRESS »

Inicio Notificações Mensagens Descobrir Buscar no Twitter Tweetar



TWEETS 2.874 SEGUINDO 161 SEGUIDORES 1,81 mi CURTIU 328 LISTAS 6 Seguir

Curiosity Rover 

@MarsCuriosity

NASA's latest mission to explore the surface of Mars. Roving the Red Planet since Aug. 5, 2012 (PDT) (Aug 6 UTC).

Gale Crater, Mars

mars.jpl.nasa.gov/msl/

Participa desde julho de 2008

 Tweetar para

Tweets **Tweets e respostas** **Fotos e vídeos**

Curiosity Rover @MarsCuriosity · 10 de fev.

Cue "Chariots of Fire" MT @MarsRovers Oppy nears 26.2-mile marathon mark on Mars go.nasa.gov/1Ce4566



Quem seguir · Atualizar · Ver todos

 Marcelo Adnet  @marcel... Seguir

 Jonny Ken Itaya @jonneken Seguir

 Giovanna Antonelli  @gi... Seguido por stenio canizio e ... Seguir

Contas populares · Encontrar amigos





Curiosity Rover @MarsCuriosity - 5 de fev

Call me DJ MSL. Smooth beats with my drill returned a great sample of Mars. go.nasa.gov/1DDxvtV



640

825

...

Ver tradução 





 **Curiosity Rover** @MarsCuriosity · 3 de dez

Pics I take on Mars get posted online. Some high-res, some thumbnail. Some B&W, others color: mars.jpl.nasa.gov/msl/multimedia...



1 mil

★ 1,2 mil

...



Curiosity comemora 1 ano marciano com nova selfie



Recomende



g+



2.134 views



-



Salvar notícia

NASA/JPL-Caltech/MSSS



Curiosity: selfie é união de várias imagens feitas pelo Curiosity entre abril e maio de 2014

Vanessa Daraya,
de INFO Online

São Paulo - O [Curiosity](#) completa um ano marciano (687 dias terrestres) nesta terça-feira (24). Para comemorar o aniversário, a NASA (agência espacial americana) divulgou uma selfie do jipe-robô em [Marte](#).

A selfie, na verdade, é a união de várias imagens feitas pelo Curiosity entre abril e maio de 2014.

As fotos foram feitas com a câmera Mars Hand Lens Imager, uma câmera que fica no braço robótico do Curiosity.



Por que estudar C?

- O sistema operacional padrão da NASA desde 1994 é o **vxWorks** (escrito em C), um sistema de **microkernel** de código fechado com 30 anos de maturidade, desenvolvido pela **Wind River**, uma subsidiária da Intel.
- Roda em mais de 1 bilhão de sistemas embarcados, inclusive nos roteadores AirPort Extreme da Apple e no humilde Linksys WRT54G



Por que estudar C?

- Onde mais roda o vxWorks?

CERTIFIED PRODUCTS

VxWorks makes it easier and more cost-effective for technology suppliers to meet the stringent certification requirements of the aerospace, defense, transportation, and other industries.



Wind River VxWorks 653 Platform



Wind River VxWorks Cert Platform



Wind River VxWorks MILS Platform

INDUSTRY PRODUCTS

VxWorks delivers a rich set of features out of the box, so customers can more quickly and efficiently build breakthrough devices for the Internet of Things (IoT). Optional industry and technology add-on profiles enhance VxWorks with a broad assortment of capabilities that help our customers meet technology and certification requirements specific to their industry.



Aerospace Profile for VxWorks



Industrial Profile for VxWorks



Medical Profile for VxWorks



Networking Profile for VxWorks



Consumer Profile for VxWorks



Por que estudar C?

- Produtos que utilizam o vxWorks:

- Veículos Mars Exploration Rover: Spirit e Opportunity
- Mars Reconnaissance Orbiter e a Sonda espacial Deep Impact
- Stardust
- O avião Boeing 787
- O sistema iDrive da BMW
- Roteador wireless Linksys WRT54G (versões 5.0 e posterior)
- Algumas impressoras Xerox da linhas Phaser, alguns multifuncionais da linha WorkCentre e outras impressoras PostScript
- Impressoras e multifuncionais da Fuji Xerox
- Experimental Physics and Industrial Control System (EPICS)
- DIGIC II processador de imagens da Canon
- SO-2510 telefone a satélite da Thuraya
- Apache Longbow - helicóptero de ataque
- ALR-67(V)3 Radar usado no A/F-18
- HN7000 receptores de internet a satélite da Hughesnet
- Mitel ICP 3000 IP PBX
- Nortel Succession 1000
- Controladores de Robos da KUKA e da ABB
- Módulos Siemens NCUI e HG1500
- Dvr Honeywell Rapid-Eye
- O avião P3-AM Força Aérea Brasileira
- Controladores de iluminação GrandMA



Por que estudar C?

- O vídeo da NASA fala de **700 mil linhas** de código para a manobra de EDL (Entrada, Descida e Pouso). Quanto isso representa, do total das sub-rotinas da **Curiosity**?
- Segundo dados fornecidos por um workshop em software de naves espaciais, a **Curiosity** tem no total 2,5 milhões de linhas de código.
- No caso, C. Não Objective C, Não Java, Não C++ e por tudo que há de mais sagrado, não Perl. O bom e velho C, desenvolvido seguindo as especificações do JPL.



Por curiosidade:

- 2,5 milhões de linhas parece muito, mas o conjunto de softwares da Apollo 11 tinha 145 mil.
- O ônibus espacial voava (e caía com estilo, na volta) com 400 mil linhas.
- O Windows XP tem 45 milhões de linhas, o OS X 10.4 tem 86 milhões e mesmo o Android nada menos que 12 milhões.



Por curiosidade:

- Com esses 2,5 milhões de linhas o **vxWorks** mantém a **Curiosity** funcionando, com todas as constantes checagens de segurança, diagnósticos e logs, além das tarefas normais.
- São mais de **160 threads**, algumas com capacidade de **disparar alarmes** e assumir prioridade máxima em frações de milionésimos de segundo, como as que controlam a temperatura dos vários componentes.



Por curiosidade:



- Exemplo da câmera com microscópio no braço da Curiosity. Isso é areia (da Terra!).
- A câmera consegue focar em uma área de $2,2 \times 1,7\text{cm}$, gerando imagens com resolução de 13,9 microns por pixel



Porque estudar C?

- Você gostaria de trabalhar em um local assim?



- Visão frontal em Irvine, Califórnia



Porque estudar C?

- Você gostaria de trabalhar em um local assim?

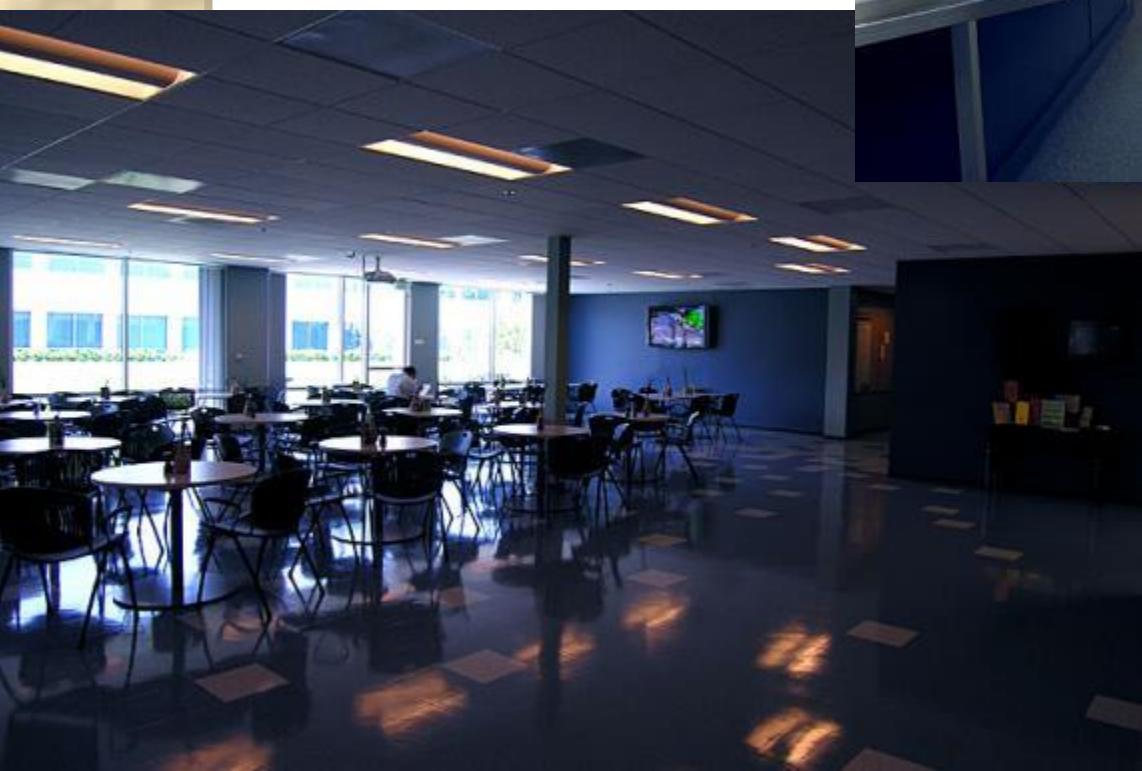


- Local de trabalho e área de “descanso”



Porque estudar C?

- Você gostaria de trabalhar em um local assim?



- Visão da cantina e da área de alimentação (gratuita)



Essa é a Blizzard (Activision/Blizzard)



The screenshot shows the official website of Blizzard Entertainment. At the top, there's a navigation bar with links for JOGOS, EMPRESA, COMUNIDADE, SUPORTE, and LOJA. Below the navigation is a large banner featuring characters from their games, including a female demon, a large bear-like creature, and a female warrior. A globe graphic is visible on the right side of the page. In the center, there's a section titled "OPORTUNIDADES DE CARREIRA" (Careers) with a paragraph of text. To the right of this text is a call-to-action button labeled "ver todos os empregos >". Below this button are two links: "FAQ de Emprego" and "Assistência Técnica do Sítio de Emprego".

BLIZZARD ENTERTAINMENT

JOGOS EMPRESA COMUNIDADE SUPORTE LOJA

Início > Empresa > Oportunidades de Carreira

OPORTUNIDADES DE CARREIRA

O **emprego dos sonhos** está esperando por você. Se está procurando algo que o desafie e inspire, que lhe ofereça enorme desenvolvimento pessoal e profissional, a Blizzard Entertainment é o lugar certo para você!

Você vai trabalhar com algumas das pessoas mais talentosas, motivadas e criativas da indústria dos jogos. Juntos, vocês criarão o entretenimento mais espetacular do mundo. A Blizzard Entertainment tem interesse em calivar e contratar os melhores e os mais brilhantes no ramo dos jogos; incentivamos um ambiente de criatividade e camaradagem com salários e benefícios altamente competitivos. Encontre o emprego dos seus sonhos hoje!

[ver todos os empregos >](#)

[FAQ de Emprego](#) >
[Assistência Técnica do Sítio de Emprego](#) >

Como faço para trabalhar na Blizzard?

Consulte na [seção de empregos](#) do nosso website uma lista atual de oportunidades.

Vocês têm estágios ou empregos de férias?

Visite a nossa [página de Relações Universitárias](#) para obter mais informações a respeito de oportunidades de estágio.

Como conseguir emprego de designer na Blizzard?

São raras as vagas de design de jogos. A melhor maneira de tentar conseguir uma vaga de designer de jogos na Blizzard é, primeiro, conseguir uma vaga de designer de níveis 3D. O designer de níveis bem-sucedido demonstra muitas das habilidades necessárias também para a função de designer de jogos.

Crio mapas excelentes no StarCraft. Como conseguir emprego de designer de mapas?

Os mapas do StarCraft e do Warcraft II Puds não são mais usados na programação. Os designers de níveis/mapas da Blizzard trabalham em projetos vindouros que exigem um novo grupo de habilidades em 3D. A melhor maneira de ser escolhido para um cargo de design de níveis 3D é jogar um jogo 3D que tenha um editor de mapas, como o Warcraft III. Você poderá, então, criar alguns níveis e enviá-los para avaliação, junto com o seu currículo.

O que preciso saber para ser programador da Blizzard?

A Blizzard tem várias vagas de programação. Precisamos de programadores para trabalhar na Battle.net, em jogos, em gráficos e 3D, ferramentas e outras áreas. Todas as vagas requerem conhecimentos profundos de C e C++. Quem tem um alicerce sólido nessa área pode adquirir outras habilidades. Conhecimentos de probabilidade, estatística, geometria e outras ciências matemáticas são úteis. Consulte a [seção de empregos](#) do nosso website para conhecer mais detalhes sobre as vagas de

Quero trabalhar na Blizzard, mas sou de outro país.

Caso não seja cidadão dos Estados Unidos, vai precisar de um visto de trabalho para trabalhar nos Estados Unidos. Um diploma universitário ou experiência profissional podem ajudá-lo a obter o visto de trabalho.

E se a vaga que eu quero não estiver sendo anunciada?

A Blizzard está sempre procurando pessoas dedicadas e talentosas, e o incentivamos a criar um Perfil de Candidato da Blizzard na [nossa seção de empregos](#). Ao criar o Perfil de Candidato, você poderá especificar os seus critérios preferidos de pesquisa de cargos, para futuras vagas aqui na Blizzard Entertainment.

O que fazer para ser testador de jogos? Posso fazer isso em casa?

Os testadores de jogos são, inicialmente, contratados como empregados temporários que devem ser residentes do sul da Califórnia. Exigimos que todos os testadores de jogos trabalhem na nossa sede em Irvine, Califórnia.

Vocês contratam designers de áudio independentes?

Temos uma equipe de som permanente, em tempo integral, que cuida de todos os nossos designs e composições de áudio. Se estiver interessado em um cargo de expediente integral ou num possível trabalho autônomo, consulte o nosso site de empregos para saber de vagas futuras.

O que significa quando envio o meu currículo, mas não recebo nenhuma resposta sua?

O nosso processo de análise é exaustivo, então, aguarde de duas a três semanas para que tomemos uma decisão a respeito da sua inscrição. Infelizmente, em razão do grande número de inscrições que recebemos diariamente, talvez não nos seja possível lhe dar um retorno se não tivermos aprovado a sua inscrição.



Dando um zoom!

O que preciso saber para ser programador da Blizzard?

A Blizzard tem várias vagas de programação. Precisamos de programadores para trabalhar na Battle.net, em jogos, em gráficos e 3D, ferramentas e outras áreas. Todas as vagas requerem conhecimentos profundos de C e C++. Quem tem um alicerce sólido nessa área pode adquirir outras habilidades. Conhecimentos de probabilidade, estatística, geometria e outras ciências matemáticas são úteis. Consulte a seção de empregos do nosso website para conhecer mais detalhes sobre as vagas de programação.

O grifo é do professor!



Senior Software Engineer, C++

BATTLE.NET

Office: Irvine, California, United States

< Voltar ao diretório



[E-mail este anúncio de emprego para um amigo](#)

[Curtir](#) 0



Blizzard Entertainment is looking for a talented and enthusiastic senior software engineer to join our Battle.net team in Irvine, California.

The new Battle.net is a full-featured online game service that will deliver the ultimate social and competitive experience for Blizzard Entertainment gamers everywhere. Designed specifically around Blizzard Entertainment titles, Battle.net will include a complete set of around-the-game features including a state-of-the-art matchmaking system, achievement system, social networking features, structured competitive play options, a marketplace, and much more. Our vision is to create an environment where gamers can compete online, develop an online persona, and stay connected to friends and the rest of the community while enjoying our games.

The senior software engineer must be able to work both independently and in conjunction with team members and product groups. Enthusiasm and flexibility in working on a variety of projects are also necessary. The ideal applicant is someone who enjoys technical challenges and the satisfaction of overcoming them.

This position within the Battle.net team is focused on the development of the Battle.net platform which provides APIs and technologies that underpin the features being developed for the service.

Requirements

- Mastery of C++
- Networking experience in TCP and UDP protocols
- Distributed systems experience
- A passion for video games
- Protocol design / optimization experience
- Security considerations (encryption, denial of service) experience
- Architecture (cloud, client / server) experience
- Knowledge of network and server security issues
- Experience with code optimization
- Bachelor's or Master's degree in Comp

Pluses

- Scripting language experience (Python, Lua)
- UML diagramming experience (class, sequence)
- Low-level network knowledge and diagnosis including packet capture (tcpdump, wireshark), routing, firewalls, DHCP, DNS, NAT busting
- Familiar with higher-level network protocols such as HTTP, SNMP, SMTP, FTP
- Multiple platform development experience (Linux, Windows, OSX)
- Database development experience (MySQL, Oracle)
- Excellent verbal and written communications skills
- Agile development exposure

Blizzard Entertainment, Inc. and its affiliated companies is an equal opportunity and affirmative action employer.

[Inscreva-se Online](#)



Software Engineer

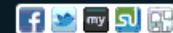
IT - NETWORK OPERATIONS

Office: Irvine, California, United States

< Voltar ao diretório

[E-mail este anúncio de emprego para um amigo](#)

[Curtir](#) 0



Blizzard Entertainment is looking for a talented and enthusiastic software engineer to join our network operations team in Irvine, California as a tools programmer. You will work closely with systems engineers to identify gaps in our production environment where a little bit of code will go a long way to improve our overall effectiveness. The primary focus of the tools developed will be to improve monitoring and automation in a large 24x7 customer-facing environment.

You must be able to work both independently and in conjunction with team members and product groups. Enthusiasm and flexibility in working on a variety of projects are also necessary. The ideal applicant is someone who enjoys technical challenges and the satisfaction of overcoming them.

Responsibilities

- Improve, extend, and maintain our existing in-house monitoring framework.
- Develop and implement new features for current monitoring tools.
- Identify gaps in monitoring and automation and develop solutions to fill them.
- Design and develop tools to empower the Network Operations Center.

Requirements

- A minimum of 3 years' programming experience
- Tools development experience
- Web programming experience
- Excellent Python, C, C++ programming skills
- Able to write clear, maintainable code
- Experience with Linux system administration and Shell scripting
- Excellent written and verbal communication skills, including documentation
- Excellent problem solving skills
- Able to work as part of a team

Pluses

- A degree in computer science, or another relevant field
- Experience programming in other relevant languages (Java, Ruby, etc.)
- Experience with network monitoring solutions
- Experience with Linux systems in production environments
- Linux systems and network programming experience
- Experience with Windows programming
- A passion for video gaming

Applicants must submit (i) a cover letter, (ii) and a resume. Only resumes with cover letters will be considered.

Blizzard Entertainment, Inc. and its affiliated companies is an equal opportunity and affirmative action employer.



Porque estudar C

- A imensa maioria dos **sistemas operacionais**, **vídeo games** e **jogos** em geral, software **embarcados** (para eletrodomésticos, telefone celular, **firewalls** e **routers**, terminais de pagamento tipo Cielo, etc), e qualquer coisa que precise de **desempenho** como bancos de dados ou **ferramentas** de cálculo intensivo, são escritos em C ou C++
- E não há previsão de mudanças.



Por que aprender a programar em C?

- Portabilidade entre máquinas e sistemas operacionais
- Linguagem procedural, simples e de aprendizado fácil.
- Trabalha com o paradigma de programas menores, com menos erros
- Baseada em construções simples, que usam recursos da máquina de forma eficiente
- Oferece recursos de modularização necessários para o desenvolvimento de sistemas de grande porte
- Uso amplamente difundido
- C Standard Library, Ponteiros, Structs
- Amplamente utilizada.



Organização de programas C

- A estrutura de um programa C é composta por **dados** e **funções**
- Os dados são as **variáveis** que são **inicializadas** antes do início do programa
- Funções são basicamente pequenas partes de um código organizadas em módulos agrupados e desenvolvidos de tal forma que conseguem realizar determinadas tarefas
- C possui uma função especial: a **main**, principal função do programa. É onde ocorre o início da execução de qualquer programa desenvolvido em C



Ambientes de desenvolvimento

- **Dev C++** <http://www.bloodshed.net/devcpp.html>
 - É um IDE (Integrated Development Environment) free. Compilador; Editor; Linkeditor; Loader e depurador.
- **Code::blocks** - <http://www.codeblocks.org/>
 - É um IDE free, totalmente extensível e configurável. Pode ser estendido com plugins. Importa projetos do Dev C++. Tem suporte a múltiplos compiladores.
- **Eclipse**
- **Visual C++ Express**
- **WxDev C++**
- **NetBeans**
- **Online:**
 - https://www.tutorialspoint.com/compile_c_online.php
 - <https://repl.it/>





Projeto Classes Debug

preenche_aleatorio_vetor_float.cpp

- random(int n): int
- main()
- exibe_vetor_float(float v[]): void
- preenche_vetor_random(float vet[]): void

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <time.h>
4 #define MAX 100 //tamanho maximo do vetor
5 /* Função :Gera numeros aleatorios do tipo float
6 Autor : Edkallenn - Data : 06/04/2012
7 Observações: Usa a função rand() e srand para gerar numeros aleatórios
8 */
9 void preenche_vetor_random(float []);
10 void exibe_vetor_float(float *);
11 int random(int n);
12
13 main(){
14     float x[MAX]; //vetor original
15     srand((unsigned)time(NULL)); //inicializa gerador de nos. aleatorios
16     preenche_vetor_random(x);
17     exibe_vetor_float(x);
18     //no DevC++ coloque aqui system("pause"); ou getchar();
19     return 0;
20 }
21 void exibe_vetor_float(float v[]){ //Exibe o vetor
22     int t;
23     printf("\nO vetor digitado eh\n");
24     for (t=0;t<MAX;t++)
25         printf("\t%5.2f\t", v[t]);
26 }
27 void preenche_vetor_random(float vet[]){
28     int i;
29     float valor, num;
30     for (i=0;i<MAX;++i){
31         num = 1 + random(MAX); //gera divisor para gerar float
32         valor = (1 + (10/num)) + random(MAX-1); //gera ate 100
33         vet[i]=valor;
34     }
35 }
36 int random(int n){ //funcao para gerar aleatorios
37     return rand() % n;
38 }
```

estruturas\ponto.c [TAD_ponto] - Code::Blocks 10.05

File Edit View Search Project Build Debug wxSmith Tools Plugins Settings Help

Build target: Debug

Management Projects Symbols Resources

Workspace TAD_ponto

- Sources
 - estruturas
 - ponto.c
 - testa_ponto.c
- Headers
 - estruturas

pto_cria(float x, float y) : Ponto*

estruturas\testa_ponto.c estruturas\ponto.c estruturas\ponto.h

```
1 #include <stdlib.h>
2 #include <stdio.h>
3 #include <math.h>
4 #include "ponto.h"
5
6 struct ponto {
7     float x;
8     float y;
9 };
10
11 Ponto *pto_cria (float x, float y){
12     Ponto *p = (Ponto*) malloc(sizeof(Ponto));
13     if (p == NULL) {
14         printf("Memoria insuficiente!\n");
15         exit(1);
16     }
17     p->x = x;
18     p->y = y;
19     return p;
20 }
21
22 void pto_libera (Ponto *p)
23 {
24     free(p);
25 }
26
27 void pto_acessa (Ponto *p, float *x, float *y)
28 {
29     *x = p->x;
30     *y = p->y;
31 }
```

Scripting console

Welcome to the script

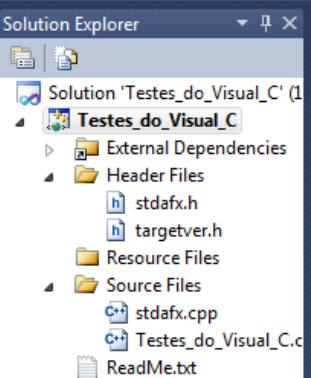
Logs & others

Code::Blocks Search results Cccc Build log Build messages CppCheck CppCheck messages Debugger Thread se

C:\Users\Ed\Dropbox\001 - Estrutura de dados - ed\programas-c\estruturas\ponto.c

WINDOWS-1252 Line 15, Column 17





```
serie_de_taylor.cpp x string_basico.cpp teste-ms_visual_C...us_2010_express.cpp Testes_do_Visual_C.cpp
(Unknown Scope)

/* Função : Série de Taylor
 / Autor : Edkallenn
 / Data : 04/10/2012
 / Observações: A série de Taylor é uma série de funções especificada em: http://pt.wikipedia.org/wiki/S%C3%A9rie\_de\_Taylor
 / Muito utilizada no Cálculo e na Análise Matemática
 / este programa usará a série de Taylor para desenvolver a exponencial (e ^ x) com x e N(precisao) digitado pelo usuário
 */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

double fat(int);
double exp(float, int);

int main(void){
    double resposta;
    float x;
    int num;
    printf("--- Programa para calcular a função exponencial e^x usando a série de Taylor ---\n\n");
    printf("Digite o valor de x: "); scanf("%f", &x);
    printf("Digite a precisão (qtde de iterações - um inteiro): "); scanf("%d", &num);
    //printf("Fatorial de %d eh %f\n\n", num, fat(num));
    resposta = exp(x, num);
    printf("\n\nEXPONENCIAL -> e ^ %.3f eh: %f\n\n", x, resposta);
    return 0;
}

double fat(int x){
    if ((x == 1) || (x == 0))
        return 1;
    else
        return x * fat(x-1);
}

double exp(float a, int num){
    double value;
    int i;
    value=1;
    for(i=1;i<=num;i++){
        printf("\nPASSO %d %f e %f\n", i, pow(a,i), fat(i));
        value= value + (pow(a,i) / fat(i));

    }
    return value;
}
```

100 %

Output

Show output from: 



Home Features Downloads Forums Wiki

Main

- Home
- Features
- Screenshots
- Downloads
 - Binaries
 - Source
 - SVN
- Plugins
- User manual
- Licensing
- Donations

Quick links

- FAQ
- Wiki
- Forums
- Forums (mobile)
- Nightlies
- BugTracker
- PatchTracker
- Browse SVN
- Browse SVN log



Please select a setup package depending on your platform:

- Windows 2000/XP/Vista/7//8
- Linux 32-bit
- Linux 64-bit
- Mac OS X

NOTE: There are also more recent *nightly builds* available in the [forums](#) or (for Debian and Fedora users) in Jens' [Debian repository](#) and Jens' [Fedora repository](#). Please note that we consider nightly builds to be *stable*, usually.

MIRRORS: BerliOS mirrors all files *usually* at SourceForge using a "BerliOS robot" [here](#). As this sometimes doesn't work, we have mirrored all file releases at SourceForge, too [here](#). The latter is managed by us.

IMPORTANT NOTE: If you try to download from BerliOS and get a "Too many clients!" - error, you should retry to download the file. According to a BerliOS - admin, this can happen several times, before the download starts. Alternatively use one of the mirrors.

NOTE: We have a [Changelog](#) for 13.12, that gives you an overview over the enhancements and fixes we have put in the new release.



Windows 2000 / XP / Vista / 7:

File	Date	Download from
codeblocks-13.12-setup.exe	27 Dec 2013	BerliOS or Sourceforge.net
codeblocks-13.12mingw-setup.exe	27 Dec 2013	BerliOS or Sourceforge.net
codeblocks-13.12mingw-setup-TDM-GCC-481.exe	27 Dec 2013	BerliOS or Sourceforge.net

NOTE: The codeblocks-13.12mingw-setup.exe file *includes* the GCC compiler and GDB debugger from TDM-GCC (version 4.7.1, 32 bit). The codeblocks-13.12mingw-setup-TDM-GCC-481.exe file includes the TDM-GCC compiler, version 4.8.1, 32 bit. While v4.7.1 is rock-solid (we use it to compile C::B), v4.8.1 is provided for convenience, there are some known bugs with this version related to the compilation of Code::Blocks itself.

IF UNSURE, USE "codeblocks-13.12mingw-setup.exe"

Code::blocks

- Vamos usar neste curso o **Code::Blocks**
- O **Code::Blocks**, uma **IDE** de código aberto e **multiplataforma** que suporta múltiplos compiladores
- O link do slide anterior inclui tanto a **IDE** do **Code::Blocks** como o compilador **GCC** e o debugger **GDB** da **MinGW**



Exercícios sugeridos

- - Fazer um relatório/resumo cobrindo os seguintes pontos:
- - O que é a linguagem C?
- - Características da Linguagem C. Seu Histórico e Influência.
- - Como um programa em C é organizado? Exemplos de programas simples em Linguagem C
- - O que é a Linguagem C++? Histórico e influência
- - Qual a diferença entre um arquivo fonte e um arquivo objeto?
- - Principais diferenças entre os padrões da Linguagem C (ANSI C, C99 e C11)
- - A influência da Linguagem C em outras linguagens
- - Usar as regras da disciplina Metodologia Científica
- - enviar em .DOC, .DOCX, .ODF ou .PDF PARA A PRÓXIMA SEMANA (PRÓXIMA AULA)
- Enviar email para [o Google Class](#)



Sites de Vídeos e cursos online

- Coursera (<https://www.coursera.org/>)
- Microsoft Virtual Academy (<http://www.microsoftvirtualacademy.com/>)
- CodeAcademy (<http://www.codecademy.com/pt/>)
- Khan Academy (<https://pt.khanacademy.org/>)
- FGV Online – membro do OpenCourseWareConsortium – OCWC (<http://www5.fgv.br/fgvonline/Cursos/Gratuitos>)
- e-aulas da USP (<http://eaulas.usp.br/portal/home>)
- OpenCulture – Free Online Courses (http://www.openculture.com/computer_science_free_courses)
- Unesp Aberta (<http://unesp.br/unespaberta>)
- CISCO Networking Academy (<https://www.netacad.com/home>)



No YouTube

- **Curso de Programação C/C++ -**
<https://www.youtube.com/playlist?list=PLE23E683C39C32DAA>
- **Vídeos de programação variados (incluindo Java):-**
<https://www.youtube.com/user/kurtphpr/playlists>
- **Khan Academy em Português**
(<https://www.youtube.com/channel/UCXj2oSzwg6G0iBjKg33joMQ>)
- **Bóson Treinamentos (TI):-**
<https://www.youtube.com/user/bosontreinamentos/playlists>
- **Curso CC50:-** <https://www.youtube.com/user/CursoCC50>
- **Canal De Aluno para Aluno:-**
<https://www.youtube.com/user/italogross/playlists>



Citação

- “**Ciência da computação não é a ciência dos computadores, assim como a astronomia não é a ciência dos telescópios.**”
- — **Edsger W. Dijkstra**, cientista da computação, ganhador do **Prêmio Turing de 1972** por suas contribuições fundamentais em algoritmos, programas e linguagens de programação, sistemas operacionais e processamento distribuído.
- Exemplos de suas **contribuições**: o **algoritmo** para o problema do **caminho mínimo** (também conhecido como algoritmo de Dijkstra), o **sistema operacional THE** e a construção de **semáforos para coordenar múltiplos processadores e programas**.
- Outro conceito desenvolvido pelo cientista foi a auto-estabilização na área de sistemas distribuídos, uma forma alternativa de garantir a confiança de um sistema



Definições

- **Plataforma** – Conjunto constituído pelo processador e pelos softwares que caracterizam o ambiente de programação e execução (incluindo o SO e o compilador).
- **Arquitetura** – Refere-se ao processador e aos softwares de controle usados para caracterizar o ambiente computacional (tamanho da palavra, representação binária de inteiros, tratamento de sinais etc.).
- **Implementação** – Refere-se ao compilador e aos demais softwares auxiliares (montadores, ligadores, etc.)



Processo de compilação

- Código-fonte:
- Código objeto:
- Código executável:



Processo de compilação

- Código-fonte:

Código escrito em uma linguagem de programação.

- Código objeto:

- Código executável:



Processo de compilação

- Código-fonte:

Código escrito em uma linguagem de programação.

- Código objeto:

Código gerado na linguagem de máquina da arquitetura alvo.

- Código executável:



Processo de compilação

- Código-fonte:

Código escrito em uma linguagem de programação.

- Código objeto:

Código gerado na linguagem de máquina da arquitetura alvo.

- Código executável:

Código gerado na linguagem de máquina da arquitetura alvo, que pode ser diretamente executado pelo processador.



Etapas da compilação

- O compilador C realiza a compilação dos programas-fonte em quatro etapas:
- **Pré-processamento.** O texto do programa é transformado lexicamente.

Resultado → unidade de compilação.
- **Compilação.** Análise sintática e semântica.

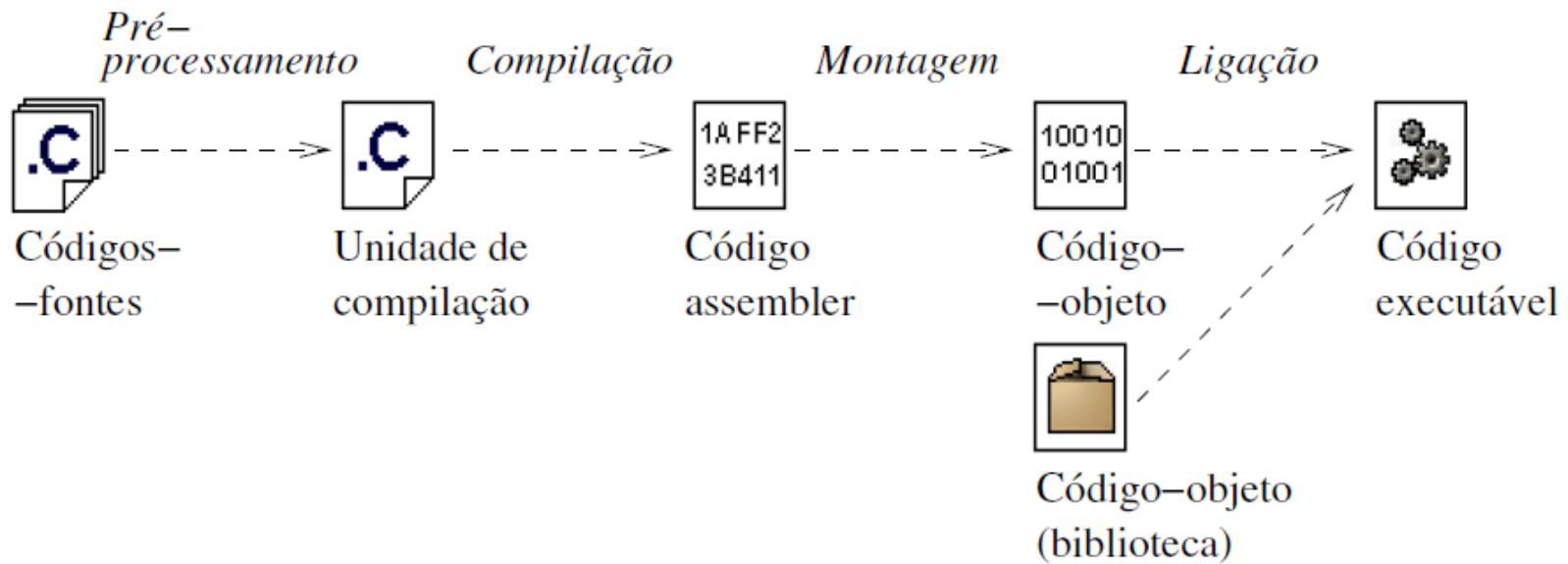
Resultado → código assembler correspondente.
- **Montagem.** Transformação para linguagem de máquina.

Resultado → código-objeto.
- **Ligação.** Combinação dos códigos-objeto que compõem o programa.

Resultado → código executável.



Etapas da Compilação



Tipos de arquivos

- .c Programas-fonte: `prog.c`, `calculo.c`.
- .h Arquivos-cabeçalho: `prog.h`, `calculo.h`.
- .s Programas assembler: `prog.s`, `calculo.s`.
- .o Programas-objeto: `prog.o`, `calculo.o`.

Os programas executáveis não possuem uma extensão padrão. (o compilador gcc usa o nome `a.out` por omissão).



Tipos de Arquivos

Arquivos-cabeçalhos são códigos-fonte que contêm a declaração de variáveis, macros e funções.

Bibliotecas são arquivos especiais que contêm o código-objeto de funções.

Alguns arquivos-cabeçalho e bibliotecas do sistema:

`stdio.h`

Funções de entrada e saída

`libc.a`

`math.h`

Funções matemáticas

`libm.a`



Inclusão de arquivos-cabeçalho

A inserção do código dos arquivos-cabeçalho nos programas-fonte se faz com a diretiva **#include**

- **#include <stdio.h>** inclui o arquivo-cabeçalho do sistema **stdio.h**.
- **#include "calcula.h"** inclui o arquivo-cabeçalho do usuário **calcula.h**, cujo código deve estar implementado em algum arquivo objeto, provavelmente o **calcula.o**.



Comando de compilação

```
gcc prog.c
```

Compila o programa que está no arquivo `prog.c` e gera um executável, armazenando-o no arquivo `a.out`.

```
gcc prog.c aux.c ent_sai.c
```

Compila o programa cujo código está distribuído nos arquivos `prog.c`, `aux.c` e `ent_sai.c` e gera um executável no arquivo `a.out`.



Comando de compilação

```
gcc -o prog prog.c
```

Compila o programa que está no arquivo `prog.c` e gera um executável no arquivo `prog`.

```
gcc prog.c aux.c ent_sai.c -o prg_exem
```

Compila o programa cujo código está distribuído nos arquivos `prog.c`, `aux.c` e `ent_sai.c` e gera um executável no arquivo `prg_exem`.



Estrutura básica de um programa em C

```
void main()    // Primeira função a ser executada
{
    /*Aqui ficaria as declarações locais e as instruções
     do programa */
}
```



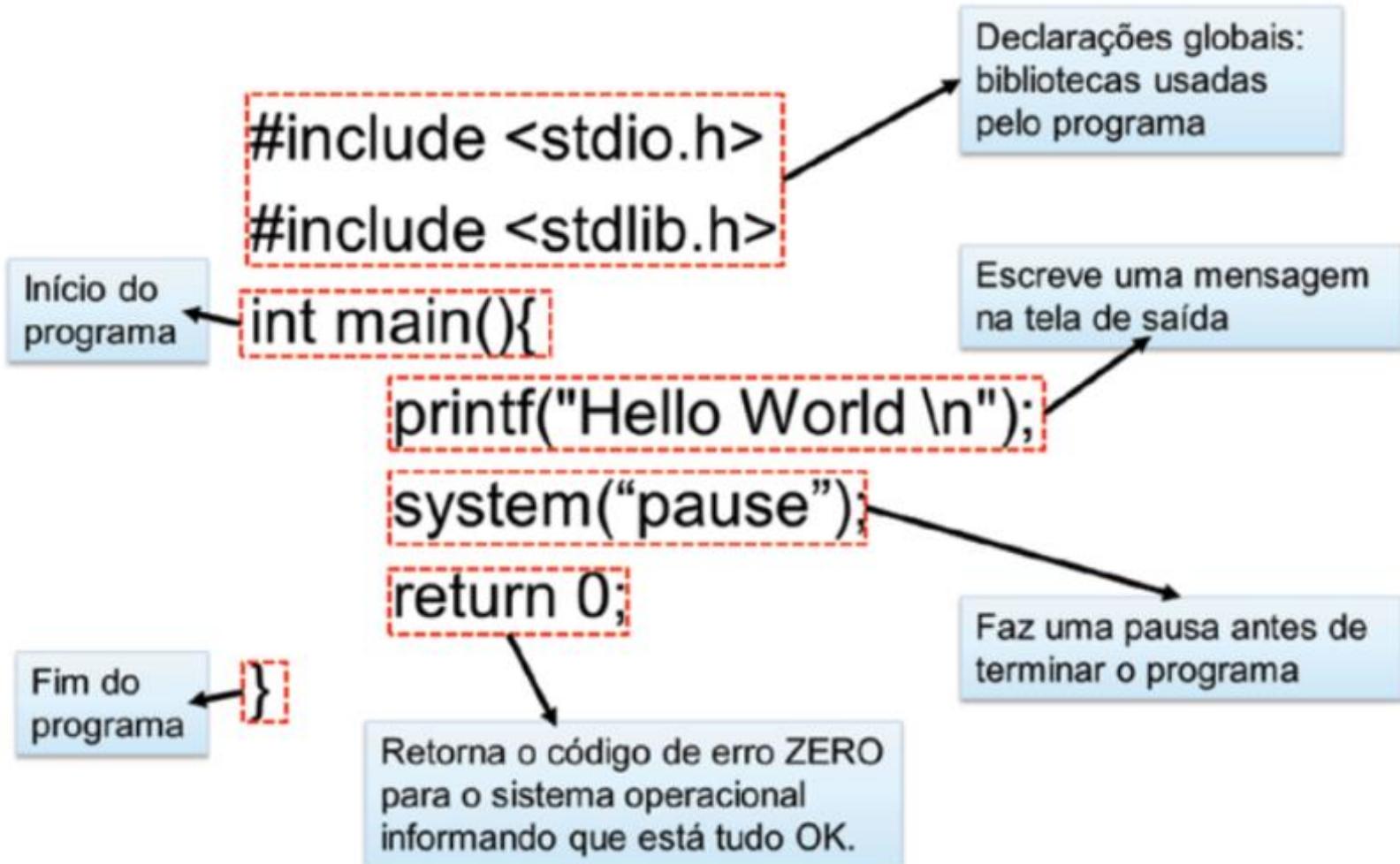
Esqueleto de um programa em C

- Todo programa escrito em linguagem C que vier a ser desenvolvido **deve possuir o esqueleto** mostrado abaixo

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main() {
5     printf("Olá Mundo! \n");
6     system("pause");
7     return 0;
8 }
```



Análise



Indentação do código

- A indentação do código é extremamente importante.
- Trata-se de uma **convenção** de escrita de códigos-fonte que visa a modificar a estética do programa para auxiliar a sua leitura e interpretação
- A **indentação** torna a leitura do código-fonte muito mais fácil e facilita a sua modificação.



Indentação

- A **indentação** é o **espaçamento** (ou tabulação) colocado antes de começar a escrever o código na linha.
- Ela tem como objetivo indicar a **hierarquia** dos elementos.
- No nosso exemplo, os comandos **printf()**, **system()** e **return** possuem a mesma hierarquia e estão todos contidos dentro do comando **main()**
- O ideal é sempre criar um novo nível de **indentação** para um novo **bloco de comandos**



Indentação

- A indentação é importante, pois o nosso exemplo anterior poderia ser escrito em apenas três linhas, sem afetar o seu desempenho, mas com alto grau de dificuldade de leitura para o programador

```
1 | #include <stdio.h>
2 | #include <stdlib.h>
3 | int main() { printf("Olá Mundo! \n"); system("pause"); return 0; }
```



Como C funciona?

```
#include <stdio.h>

int main()
{
    puts ("C Rocks!");
    return 0;
}
```

rocks.c

1

Fonte

Você começa criando um arquivo fonte. Este arquivo contém código em C legível para humanos.



No Windows, isso será
chamado `rocks.exe`,
ao invés de `rocks`



rocks

2

Compila

Você executa o seu código-fonte através de um compilador, que por sua vez verifica os erros e, quando está satisfeito, compila o código-fonte.

3

Output

O compilador cria um novo arquivo chamado *executável*. Este arquivo contém linguagem de máquina, uma fila de 1s e 0s capaz de ser compreendida pelo computador. E este é o programa que pode ser executado.



Estudo de C – palavras-chave

- Palavras-chave:
- **auto, double, int, struct
break, else, long, switch
case, enum, register, typedef
char, extern, return, union
const, float, short, unsigned
continue, for, signed, void
default, goto, sizeof, volatile
do, if, static, while**
- Importante: **C é CASE SENSITIVE.**
- **Todas as Instruções terminam com um ;**



Estudo de C

- **Tipos de Dados básicos:**
- Existem quatro tipos de dados básicos são eles: caractere, inteiro, ponto flutuante e ponto flutuante de precisão dupla (char, int, float e double)
- O tipo de dado **int (inteiro)** serve para armazenar valores numéricos inteiros. Existem vários tipos de inteiros, cada um de um tamanho diferente (dependendo do sistema operacional e/ou arquitetura do processador):
- O tipo **char ocupa 1 byte**, e serve para armazenar caracteres ou inteiros. Isso significa que o programa reserva um espaço de 8 bits na memória RAM ou em registradores do processador para armazenar um valor. Com vetores do tipo char é possível criar cadeias de caracteres (strings).



Tipos de dados básicos

- O tipo de dado **float** serve para armazenar números de ponto flutuante, ou seja, com casas decimais.
- O tipo de dado **double** serve para armazenar números de ponto flutuante de dupla precisão, normalmente tem o dobro do tamanho do float e portanto o dobro da capacidade.
- O tipo **void** declara explicitamente uma função que não retorna valor algum ou cria ponteiros genéricos. Ambos os usos serão discutidos mais adiante.



Tipos básicos

Tipo	Bits	Intervalo de valores
char	8	-128 A 127
int	32	-2.147.483.648 A 2.147.483.647
float	32	1,175494E-038 A 3,402823E+038
double	64	2,225074E-308 A 1,797693E+308
void	8	sem valor



Estudo de C

- Tabela de tipos básicos(c/ alguns modificadores de tipo):

Tipo	Tamanho	Menor valor	Maior valor
char	1 byte	-128	+127
unsigned char	1 byte	0	+255
short int (short)	2 bytes	-32.768	+32.767
unsigned short int	2 bytes	0	+65.535
int (*)	4 bytes	-2.147.483.648	+2.147.483.647
long int (long)	4 bytes	-2.147.483.648	+2.147.483.647
unsigned long int	4 bytes	0	+4.294.967.295
float	4 bytes	-10^{38}	$+10^{38}$
double	8 bytes	-10^{308}	$+10^{308}$

(*) depende da máquina, sendo 4 bytes para arquiteturas de 32 bits



Estudo de C – Tipos básicos

- Digite o seguinte programa no compilador e execute

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

// Função : Programa para mostrar o tamanho dos tipos em C
// Autor : Edkallenn
// Data : 06/04/2012
// Observações: sizeof é um operador em tempo de compilação unário que retorna o
//               tamanho, em bytes, da variável ou especificador de tipo,
//               em parenteses que ele precede.

main()
{
    printf("Tamanho do tipo de dado int: %d bytes\n", sizeof(int));
    printf("Tamanho do tipo de dado char: %d byte\n", sizeof(char));
    printf("Tamanho do tipo de dado float: %d bytes\n", sizeof(float));
    printf("Tamanho do tipo de dado double: %d bytes\n", sizeof(double));

    system("pause");
}
```



Estudo de C

- **Definição de Variáveis – Escopo**
- Variáveis são posições da memória previamente identificadas para armazenar as informações a serem utilizadas pelo programa. Dependendo do local onde essas duas variáveis são declaradas podemos chamá-las de **Locais** ou **Globais**.
- **Variáveis locais:** São aquelas declaradas **dentro de funções**. Essas variáveis só podem ser utilizadas pelas instruções que estão **dentro do bloco em que foram declaradas**.
- Um bloco de código inicia-se em abre-chaves ({}) e termina em fecha-chaves (}). Essas variáveis locais só existem no momento em que o bloco de instrução que as contém estão sendo executado.



Estudo de C

- **Definição de Variáveis – Escopo**
- **Variáveis Globais:** As declarações das variáveis globais são feitas fora de todas as funções do programa. As variáveis globais ao contrário das locais podem ser acessadas de qualquer parte do programa e são guardadas até o término da execução do programa.
- Se uma variável **for declarada e não for especificado o seu valor, ele é indefinido, é lixo.**
- É importante inicializar as variáveis para evitar erros de compilação e, principalmente, de lógica.



Estudo de C – Nomes de Variáveis

- Os nomes da variáveis sejam elas locais ou globais identificação deve seguir as seguintes regras:
 1. O nome da variável deve conter um ou mais caracteres;
 2. O primeiro caractere sempre deve ser uma letra;
 3. Os caracteres subsequentes podem ser letras, números ou o caractere “_” que geralmente é usado para indicar a separação entre duas palavras;
 4. O nome da variável ou identificador não pode ser igual às palavras-chaves já apresentadas, nem ter o mesmo nome de funções determinadas pelo usuário ou iguais as funções localizadas na biblioteca C.



Estudo de C

- Nomes de variáveis

Válidos	Inválidos
count	lcount
teste23	hi!there
high_balance	high...balance
registro	register



Estudo de C

- **Declaração e inicialização de variáveis**
- Todas as variáveis precisam ser declaradas antes de serem utilizadas através da instrução:

<tipo de dados> nome_variavel;

Toda instrução em C deve estar acompanhada de “;”

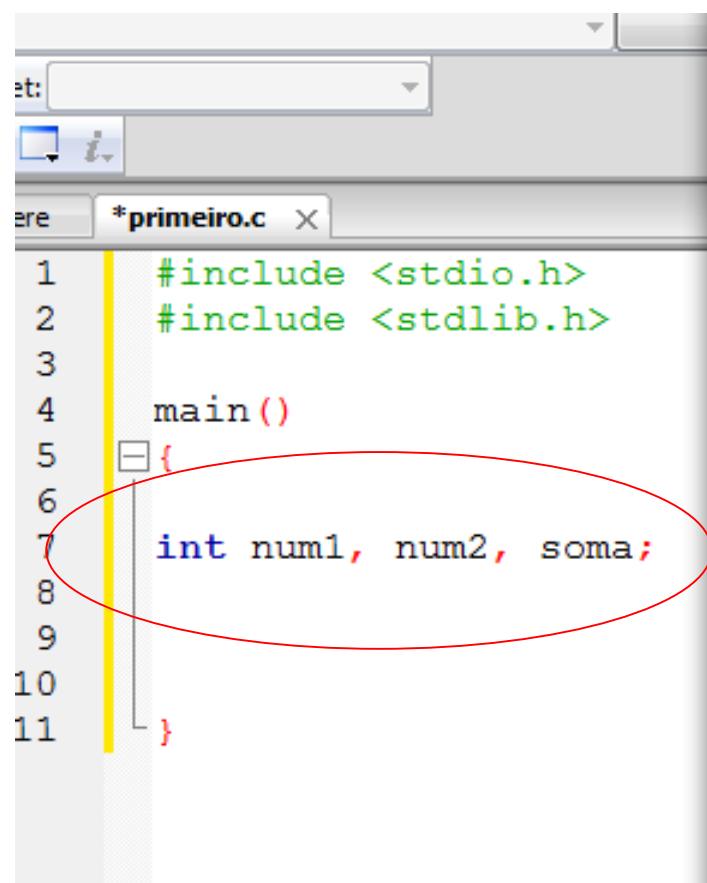
- A atribuição de valores para as variáveis é feita utilizando o sinal de igual “=” e obedece a seguinte sintaxe:

Nome_da_variavel = expressao;

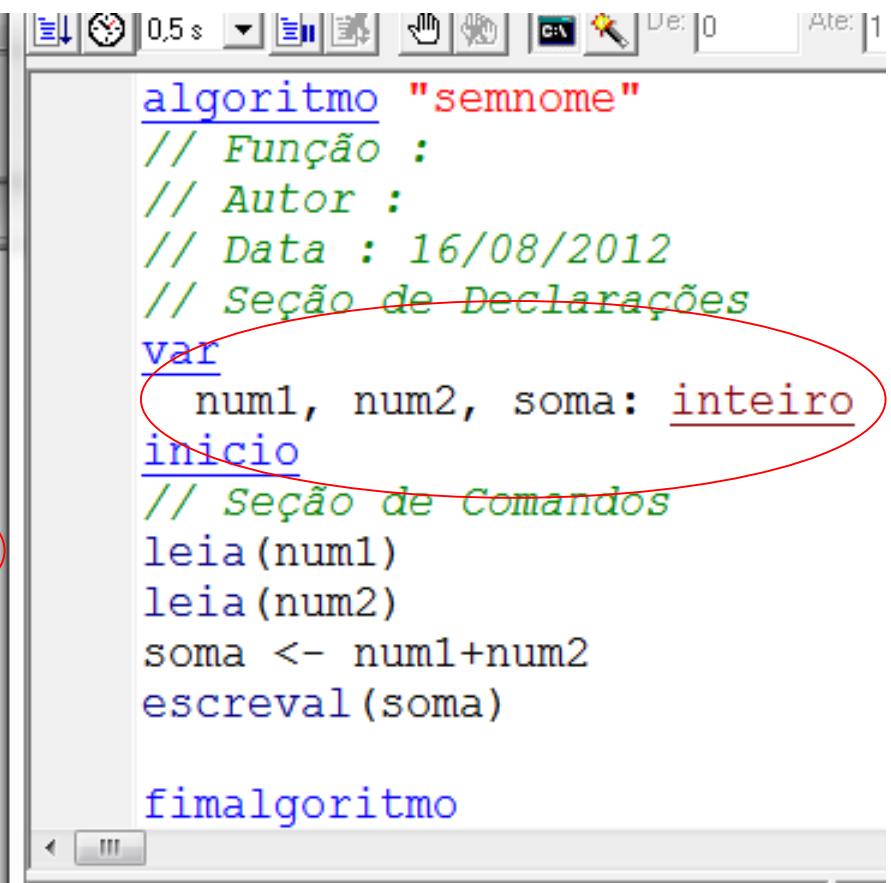
- A expressão pode ser uma simples constante ou uma fórmula matemática
- Existem os modificadores de tipo de acesso: const, volatile e de tipo de armazenamento extern, static, e do tipo register que serão estudados conforme seu uso for adequado.



Diferenças: Portugol x C/C++



```
primeiro.c
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 main()
5 {
6     int num1, num2, soma;
7
8
9
10}
11
```

A red oval highlights the declaration of variables num1, num2, and soma.

```
algoritmo "semnome"
// Função :
// Autor :
// Data : 16/08/2012
// Seção de Declarações
var
    num1, num2, soma: inteiro
inicio
    // Seção de Comandos
    leia(num1)
    leia(num2)
    soma <- num1+num2
    escreval(soma)

fimalgoritmo
```

A red oval highlights the variable declarations num1, num2, and soma, and the assignment statement soma <- num1+num2.

Constantes

- Constantes são valores fixos que o programa não altera em tempo de execução
- Constantes podem ser de quaisquer um dos 4 tipos básicos de dados
- Constantes de caracteres são envolvidas por aspas simples ". Exemplo, 'a' e '%'
- Constantes inteiras são inteiros com ou sem sinal (10, -100)
- Constantes em ponto flutuante requerem ponto decimal seguido pela parte fracionaria (11.123)
- C permite uso de notação científica para constantes de ponto flutuante



Exemplos de uso de constantes

Tipo de dado	Exemplos de constantes			
int	1	123	21000	-234
long int	35000L	-34L		
char	'a'	'b'	'v'	'F'
unsigned int	10000U	987U	40000	
float	123.23F		4.34e-3F	
double	123.23	12312333	-0.9876324	
long double	1001.2L			



Constantes hexadecimais, octais e string

- C permite constantes inteiras em hexadecimal ou octal em lugar de decimal
- Hexadecimal consiste de um 0x seguido por uma constante na forma hexadecimal
- Octal começa com 0. Exemplos:

```
int hex = 0x80;      // 128 em decimal
int oct = 012;        // 10 em decimal
```
- A string é um conjunto de caracteres colocado entre aspas duplas. Ex: “isso é uma string” é uma string.
- Embora C permita constantes string ela não tem um tipo de dado string.
- **IMPORTANTE: NÃO CONFUNDA STRING COM CARACTERES.** Uma constante de um único caracter é colocada entre aspas simples, como em ‘a’. Porém, “a” é uma string contendo uma única letra.



Constantes de barra invertida

- São constantes especiais para caracteres não imprimíveis como CR (Retorno de Carro, ou Nova Linha – O **Enter** de um editor de textos)

Código	Significado
\b	Retrocesso (BS)
\f	Alimentação de formulário (FF)
\n	Nova Linha (LF)
\r	Retorno de Carro (CR)
\t	Tabulação Horizontal (HT)
\“	Aspas duplas
\‘	Aspas simples
\0	Nulo
\	Barra Invertida
\v	Tabulação Vertical
\a	Alerta (beep)
\N	Constante Octal (N é uma constante octal)
\xN	Constante Hexadecimal (N é uma constante hexadecimal)



Exemplo prático 01

- Considere o seguinte problema:
 - Determinar o valor de $y = \sin(1,5)$.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 //#include <math.h>      // necessário para sin()
4
5 int main()
6 {
7     float y;
8     y = sin(1.5);
9     printf("y = %f", y);
10    printf("\n");
11    system("PAUSE");
12    return 0;
13 }
14 }
```



Erros de sintaxe

- Atenção!

- O **programa executável** só será gerado se o texto do programa não contiver **erros de sintaxe**.
- Exemplo: considere uma **string**. Ah?! O que é isso?! Uma **sequência de caracteres delimitada por aspas**.
- Se isso é uma string e se tivéssemos escrito:

```
printf("y = %f,y);
```

- O compilador iria apontar um erro de sintaxe nesta linha do programa e exibir uma mensagem tal como:

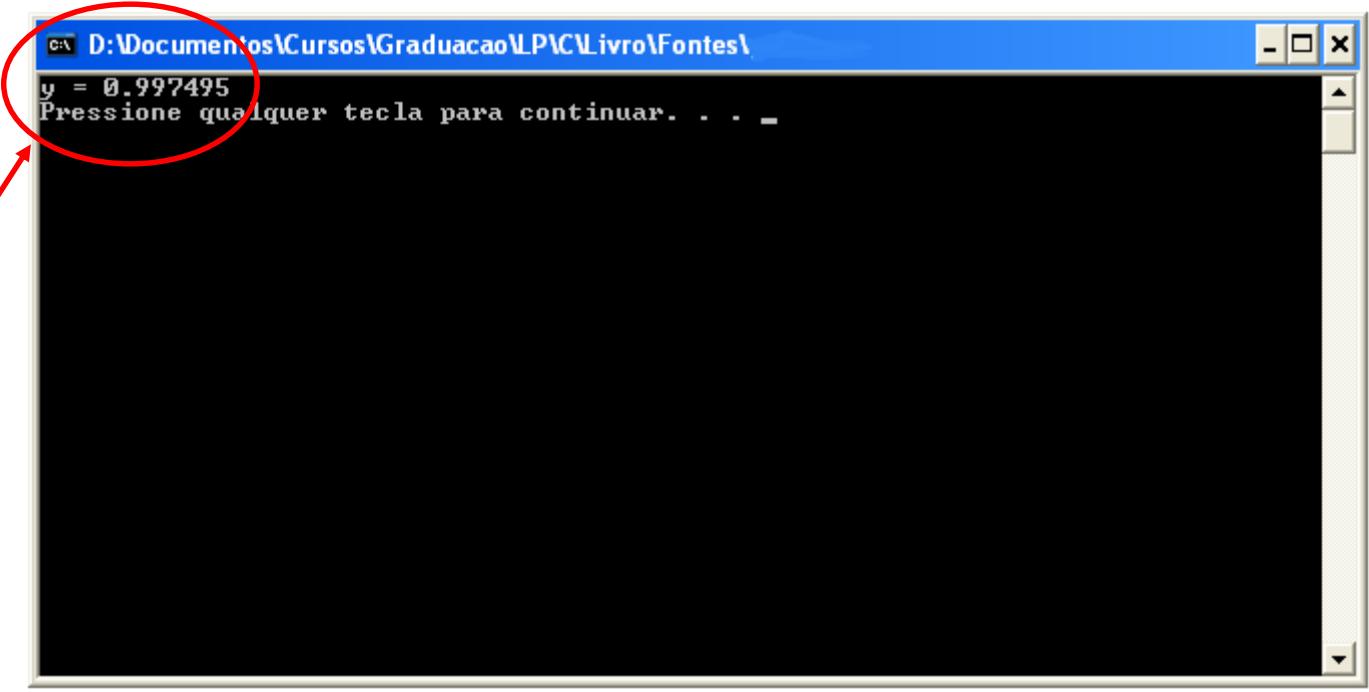
undetermined string or character constant



Exemplo prático 01

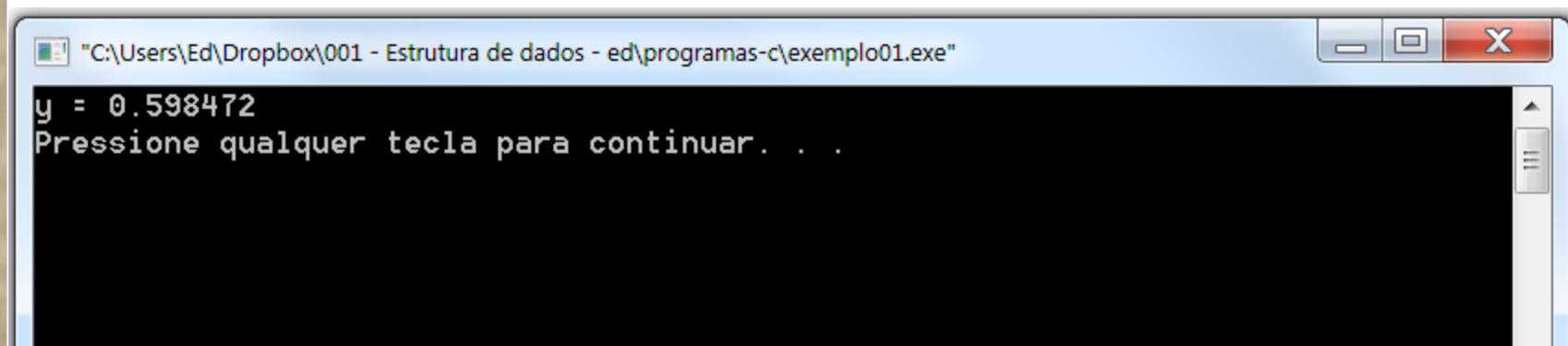
- Se o nome do programa é **exemplo01.c**, então após a **compilação**, será produzido o programa executável **exemplo01.exe**.
- Executando-se o programa exemplo01.exe (ou clicando-se em Run), o resultado será:

Problema
Resolvido!



Erros de lógica

- Atenção!
 - Não basta obter o programa executável!! Será que ele está correto?
 - Se ao invés de: `y = sin(1.5);`
 - Tivéssemos escrito: `y = sin(2.5);`
 - O compilador também produziria o programa `exemplo01.exe`, que executado, iria produzir:



A screenshot of a Windows command-line window titled "C:\Users\Ed\Dropbox\001 - Estrutura de dados - ed\programas-c\exemplo01.exe". The window contains the following text:
y = 0.598472
Pressione qualquer tecla para continuar. . .



Erros de lógica

- Embora um resultado tenha sido obtido, ele **não é correto**.
- Se um programa executável não produz os resultados corretos, é porque ele contém **erros de lógica** ou **bugs**.
- O processo de identificação e correção de erros de lógica é denominado **depuração** (**debug**).
- O nome de um texto escrito em uma linguagem de programação é chamado de **programa-fonte**.
- Exemplo: o programa **exemplo01.c** é um **programa-fonte**.



Arquivos de cabeçalho

- Note que o programa-fonte exemplo01.c começa com as linhas:

```
#include <stdio.h>
#include <stdlib.h>
```
- **Todo** programa-fonte em linguagem C começa com linhas deste tipo.
- O que elas indicam?
 - Dizem ao compilador que o programa-fonte vai utilizar arquivos de cabeçalho (extensão .h, de **header**).
 - E daí? O que são estes arquivos de cabeçalho?
 - Eles **contêm informações** que o compilador precisa para construir o programa executável.



Arquivos de cabeçalho

Como assim?

- Observe que o programa exemplo01.c inclui algumas **funções**, tais como:
sin – função matemática seno.
printf – função para exibir resultados.
- Por serem muito utilizadas, a linguagem C mantém funções como estas em **bibliotecas**.
- Atenção! O conteúdo de um arquivo de cabeçalho também é um texto.



Arquivos de cabeçalho

- Ao encontrar uma instrução **#include** em um programa-fonte, o compilador traduz este texto da mesma forma que o faria se o texto tivesse sido digitado no programa-fonte.
- Portanto, as linhas:

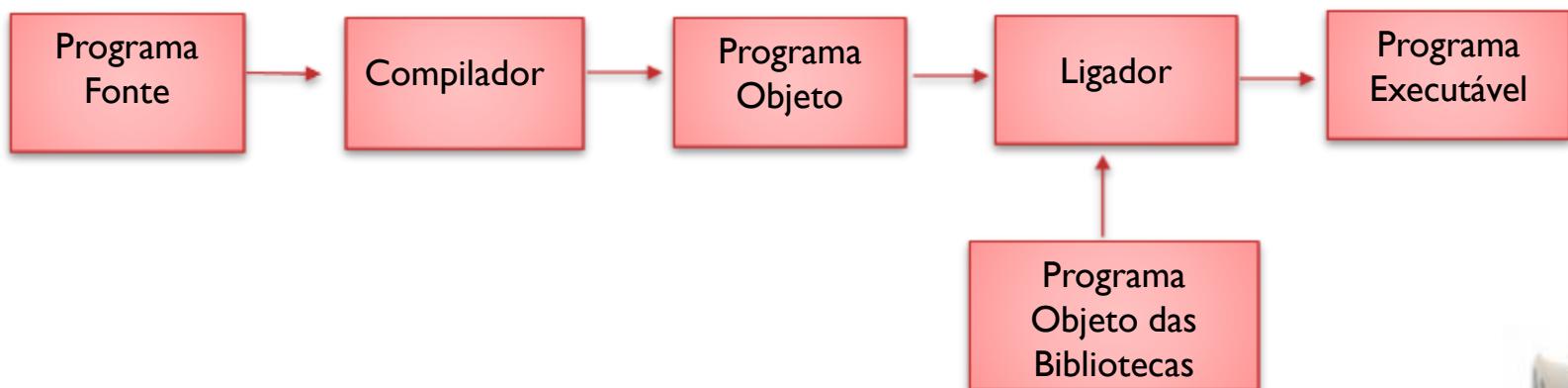
```
#include <stdio.h>
#include <stdlib.h>
```

indicam ao compilador que o programa exemplo01.c utilizará as instruções das bibliotecas **stdio** e **stdlib**.



Processo de compilação

- O processo de compilação, na verdade, se dá em duas etapas:
 - Fase de tradução: programa-fonte é transformado em um programa-objeto.
 - Fase de ligação: junta o programa-objeto às instruções necessárias das bibliotecas para produzir o programa executável.

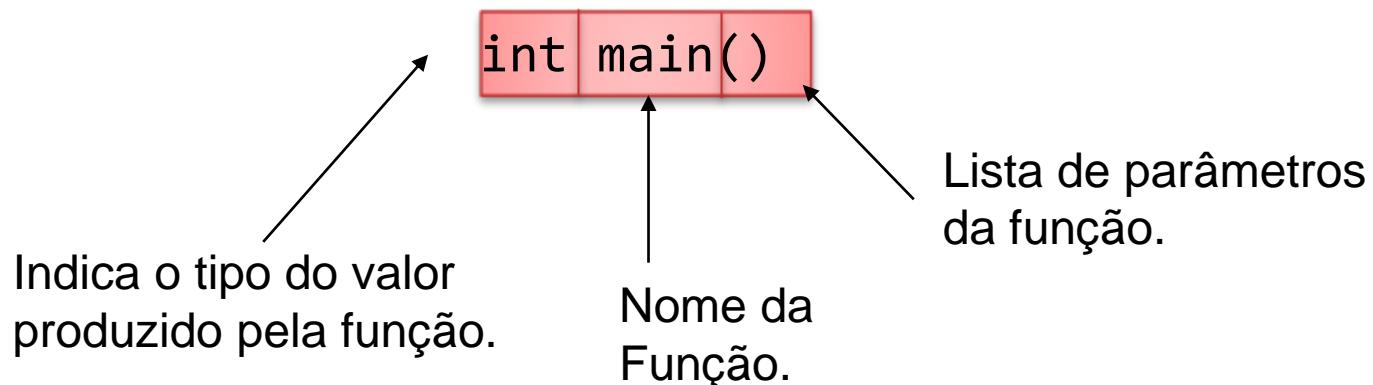


Função main

- A próxima linha do programa é:

```
int main()
```

- Esta linha corresponde ao cabeçalho da função **main** (a função principal, daí o nome **main**).
- O texto de um programa em Linguagem C pode conter muitas outras funções e **SEMPRE** deverá **conter a função main**.



Função main

- A Linguagem C é *case sensitive*. Isto é, considera as letras maiúsculas e minúsculas diferentes.
- Atenção!
 - O nome da função principal deve ser escrito com letras minúsculas: **main**.
 - **Main** ou **MAIN**, por exemplo, provocam erros de sintaxe.
- Da mesma forma, as palavras **int** e **char**, devem ser escritas com letras minúsculas.



Tipos de dados

- A solução de um problema de cálculo pode envolver vários tipos de dados.
- Caso mais comum são os **dados numéricos**:
 - **Números inteiros** (2, 3, -7, por exemplo).
 - **Números com parte inteira e parte fracionária** (1,234 e 7,83, por exemplo).
- Nas linguagens de programação, dá-se o nome de **número de ponto flutuante** aos números com parte inteira e parte fracionária.
- **Números de ponto flutuante** são os números reais que podem ser representados no computador.
- A representação com ponto flutuante segue padrões internacionais (**IEEE-754** e **IEC-559**).
- Exemplo:
 - Representação com **ponto fixo**: 12,34.
 - Representação com ponto flutuante: $0,1234 \times 10^2$.



Variáveis

- Os dados que um programa utiliza precisam ser armazenados na **memória** do computador.
- Cada posição de memória do computador possui um **endereço**.

8 1000	3.25 1001	'a' 1002	'g' 1003
'q' 1004	2 1005	** 1006	'1' 1007
1008	1009	1010	1011
1012	1013	1014	1015
1016	1017	1018	1019

← Memória



Variáveis

- A partir dos endereços, é possível para o computador saber qual é o valor armazenado em cada uma das posições de memória.
- Como a **memória pode ter bilhões de posições**, é difícil controlar em qual endereço está armazenado um determinado valor!
- Para facilitar o controle sobre onde armazenar informação, os programas utilizam **variáveis**.
- Uma variável corresponde a um **nome simbólico** de uma posição de memória.
- Seu **conteúdo pode variar** durante a execução do programa.



Variáveis

- Cada variável pode possuir uma quantidade diferente de bytes, uma vez que os tipos de dados são representados de forma diferente.
- Portanto, a cada variável está associado um tipo específico de dados.
- Logo:
 - O tipo da variável define quantos bytes de memória serão necessários para representar os dados que a variável armazena.



Revisão - Variáveis

- A Linguagem C dispõe de **quatro tipos básicos de dados**. Assim, as variáveis poderão assumir os seguintes tipos:

Tipo	Tamanho (bytes)	Valor
char	1	Um caractere (ou um inteiro de 0 a 127).
int	4	Um número inteiro.
float	4	Um número de ponto flutuante (SP).
double	8	Um número de ponto flutuante (DP).



Modificadores de tipo

- A linguagem C define alguns **modificadores de tipo**. Alguns deles são: **short**, **long**, **unsigned**.
- Um modificador de tipo altera o intervalo de valores que uma variável pode armazenar.
- Ao tipo **float** não se aplica nenhum dos modificadores, ao tipo **double** aplica-se apenas o modificador **long** e ao tipo **char** aplica-se somente o tipo **unsigned**.
- O modificador de tipo **short** instrui o compilador a representar valores inteiros usando **16 bits**.
- Logo, uma variável **short int** pode armazenar valores inteiros no intervalo: -2^{15} a $2^{15} - 1$.



Modificadores de tipo

- Para as variáveis do tipo **char**, o compilador reserva 8 bits.
- Assim, variáveis do tipo **char** podem armazenar valores inteiros no intervalo -2^7 a $2^7 - 1$.
- O modificador de tipo **unsigned** instrui o compilador a não considerar o primeiro bit como sinal. Assim, variáveis **unsigned char** podem representar valores positivos maiores. O maior valor será: $2^8 - 1$.



- Dentro do programa, as variáveis são identificadas por seus **nomes**.
- Portanto, um programa deve **declarar** todas as variáveis que irá utilizar.
- **Atenção!**
 - A **declaração de variáveis deve ser feita antes que a variável seja usada**, para garantir que a quantidade correta de memória já tenha sido reservada para armazenar seu valor.



Escrevendo um programa em C

- Escrever um programa em Linguagem C corresponde a escrever o **corpo** da função principal (**main**).
- O **corpo** de uma função sempre começa com abre-chaves **{** e termina com fecha-chaves **}**.

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

int main(int argc, char *argv[])
{
    float y;
    y = sin(1.5);
    printf("y = %f", y);
    printf("\n");
    system("PAUSE");
    return 0;
}
```

Corpo da
função →



Escrevendo um programa em C

- A primeira linha do corpo da função principal do programa **exemplo01.c** é:

```
float y;
```

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 //#include <math.h>      // necessário para sin()
4
5 int main()
6 {
7     float y;
8     y = sin(1.5);
9     printf("y = %f", y);
10    printf("\n");
11    system("PAUSE");
12    return 0;
13 }
14
```



Escrevendo um programa em C

- Esta linha declara uma variável **y** para armazenar um número de ponto flutuante (SP).
- **A declaração de uma variável não armazena valor algum** na posição de memória que a variável representa.
- Ou seja, no caso anterior, vai existir uma posição de memória chamada **y**, mas ainda não vai existir valor armazenado nesta posição.



Escrevendo um programa em C

- Um valor pode ser **atribuído** a uma posição de memória representada por uma variável pelo **operador de atribuição =**.
- O operador de atribuição requer à esquerda um nome de variável e à direita, um valor.
- A linha seguinte de **exemplo01.c** atribui um valor a **y**:

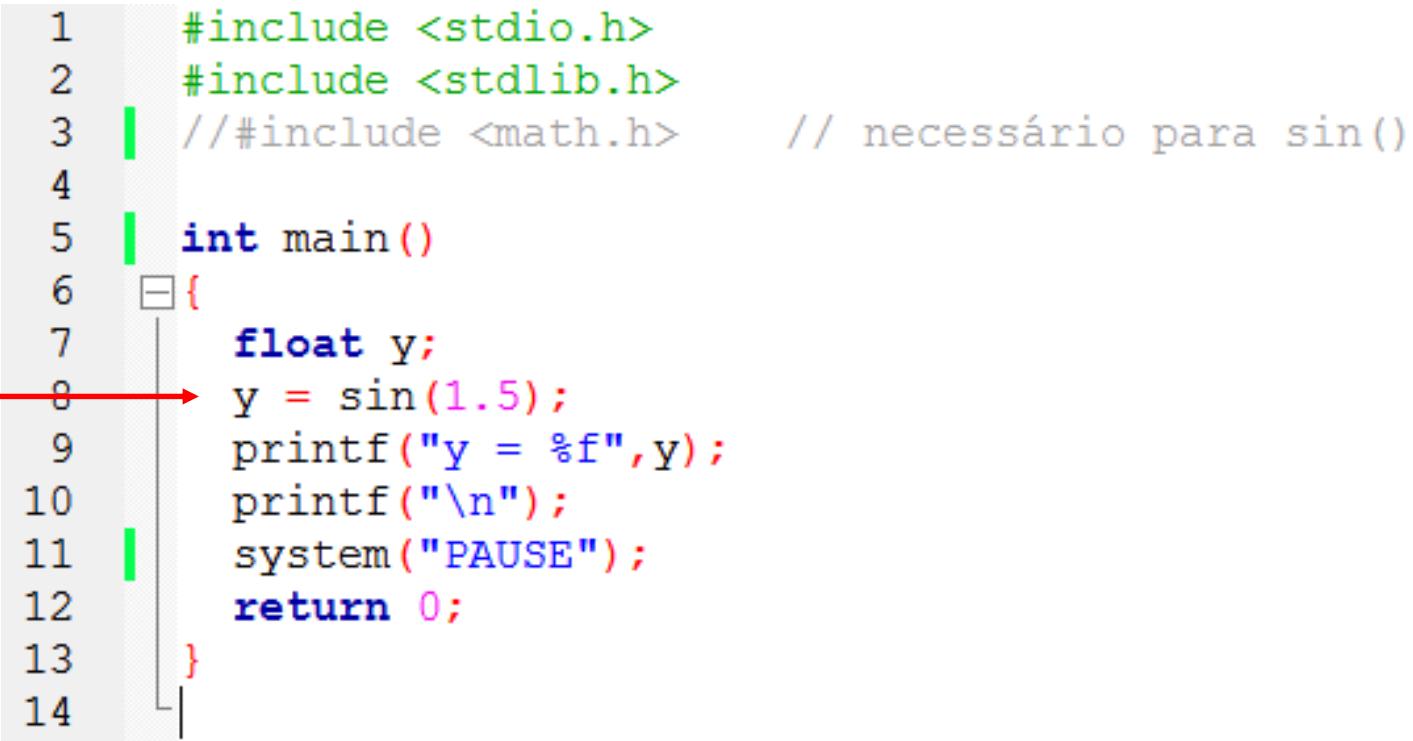
```
1  #include <stdio.h>
2  #include <stdlib.h>
3  // #include <math.h>      // necessário para sin()
4
5  int main()
6  {
7      float y;
8      y = sin(1.5);
9      printf("y = %f", y);
10     printf("\n");
11     system("PAUSE");
12     return 0;
13 }
14
```



Escrevendo um programa em C

- No lado direito do operador de atribuição existe uma referência à função **seno** com um parâmetro **1.5** (uma constante de ponto flutuante representando um valor em **radianos**.)

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  // #include <math.h>      // necessário para sin()
4
5  int main()
6  {
7      float y;
8      → y = sin(1.5);
9      printf("y = %f", y);
10     printf("\n");
11     system("PAUSE");
12     return 0;
13 }
14
```

A red rectangular box highlights the line '#include <math.h>' at line 3. A red arrow points from this box to the 'sin' function call at line 8.

Escrevendo um programa em C

- Em uma linguagem de programação chamamos o valor entre parênteses da função, neste exemplo, o valor 1.5, de **parâmetro da função**.
- Da mesma forma, diz-se que **sin(1.5)** é o **valor da função sin** para o parâmetro **1.5**.
- O operador de atribuição na linha **y = sin(1.5)** obtém o valor da função (0.997495) e o armazena na posição de memória identificada pelo nome **y**.
- Esta operação recebe o nome de: **atribuição de valor a uma variável**.



Escrevendo um programa em C

- **Atenção:** O valor armazenado em uma variável por uma operação de atribuição depende do tipo da variável.
- Se o tipo da variável for **int**, será armazenado um valor inteiro (caso o valor possua parte fracionária, ela será desprezada).
- Se o tipo da variável for **float** ou **double**, será armazenado um valor de ponto flutuante (caso o valor não possua parte fracionária, ela será nula).



Escrevendo um programa em C

- **Exemplo:**

- Considere as seguintes declarações:

```
int a;  
float b;
```

- Neste caso, teremos:

Operação de atribuição	Valor armazenado
------------------------	------------------

$a = (2 + 3) * 4$	20
-------------------	----

$b = (1 - 4) / (2 - 5)$	1.0
-------------------------	-----

$a = 2.75 + 1.12$	3
-------------------	---

$b = a / 2.5$	1.2
---------------	-----



Escrevendo um programa em C

- As próximas linhas do programa **exemplo01.c** são:

```
printf("y = %f",y);
printf("\n");
```

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 //#include <math.h>      // necessário para sin()
4
5 int main()
6 {
7     float y;
8     y = sin(1.5);
9     printf("y = %f",y);
10    printf("\n");
11    system("PAUSE");
12    return 0;
13 }
14
```

- A função **printf** faz parte da biblioteca **stdio.h**



Escrevendo um programa em C

- A função **printf** é usada para exibir resultados produzidos pelo programa e **pode ter um ou mais parâmetros.**
- O primeiro parâmetro da função **printf** é sempre uma **string**, correspondente à sequência de caracteres que será exibida pelo programa.

```
printf("y = %f",y);  
printf("\n");
```



Escrevendo um programa em C

- Essa sequência de caracteres pode conter alguns **tags** que representam valores. Estes tags são conhecidos como **especificadores de formato**.

```
printf("y = %f", y);  
printf("\n");
```

Especificador
de formato

- Um especificador de formato começa sempre com o símbolo **%**. Em seguida, pode apresentar uma **letra** que indica o tipo do valor a ser exibido.
- Assim, **printf("y = %f", y)** irá exibir a letra **y**, um espaço em branco, o símbolo **=**, um espaço em branco, e um valor de ponto flutuante.



Escrevendo um programa em C

- Na função **printf**, para cada **tag** existente no primeiro parâmetro, deverá haver um novo parâmetro que especifica o valor a ser exibido.

```
printf("a = %d, b = %c e c = %f", a, 'm', (a+b));
```

- A linguagem C utiliza o símbolo \ (barra invertida) para especificar alguns caracteres especiais:

Caractere	Significado
\a	Caractere (invisível) de aviso sonoro.
\n	Caractere (invisível) de nova linha.
\t	Caractere (invisível) de tabulação horizontal.
\'	Caractere de apóstrofo



Escrevendo um programa em C

- Observe a próxima linha do programa exemplo01.c:

```
printf("\n");
```

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 //#include <math.h>      // necessário para sin()
4
5 int main()
6 {
7     float y;
8     y = sin(1.5);
9     printf("y = %f", y);
10    printf("\n");
11    system("PAUSE");
12    return 0;
13 }
14
```

- Ela exibe “o caractere (invisível) de nova linha”. Qual o efeito disso?
- Provoca uma mudança de linha! Próxima mensagem será na próxima linha.



Escrevendo um programa em C

- Observe agora a próxima linha do programa:

```
system("PAUSE");
```

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  // #include <math.h>      // necessário para sin()
4
5  int main()
6  {
7      float y;
8      y = sin(1.5);
9      printf("y = %f", y);
10     printf("\n");
11     system("PAUSE");
12     return 0;
13 }
14
```

- Ela exibe a mensagem
“Pressione qualquer tecla para continuar...” e interrompe a execução do programa.



Escrevendo um programa em C

- A execução será retomada quando o usuário pressionar alguma tecla.
- A **última linha** do programa **exemplo01.c** é:

```
return 0;
```

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 // #include <math.h>      // necessário para sin()
4
5 int main()
6 {
7     float y;
8     y = sin(1.5);
9     printf("y = %f", y);
10    printf("\n");
11    system("PAUSE");
12    return 0;
13 }
14
```



Escrevendo um programa em C

- É usada apenas para satisfazer a sintaxe da linguagem C.
- O comando **return** indica o valor que uma função produz.
- Cada função, assim como na matemática, deve produzir um único valor.
- Este valor deve ter o mesmo tipo que o declarado para a função.



Escrevendo um programa em C

- No caso deste programa, a função principal foi declarada como sendo do tipo **int**. Ou seja, ela deve produzir um valor inteiro.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  // #include <math.h>      // necessário para sin()
4
5  int main()
6  {
7      float y;
8      y = sin(1.5);
9      printf("y = %f", y);
10     printf("\n");
11     system("PAUSE");
12     return 0;
13 }
14
```

- A linha **return 0;** indica que a função principal irá produzir o valor inteiro 0.

Escrevendo um programa em C

- Mas e daí?!! O valor produzido pela função principal não é usado em lugar algum!
- Logo, não faz diferença se a última linha do programa for:

return 0;

return 1;

ou

return 1234;



Escrevendo um programa em C

- Neste caso, o fato de a função produzir um valor não é relevante.
- Neste cenário, é possível declarar a função na forma de um **procedimento**.
- Um **procedimento** é uma função do tipo **void**, ou seja, uma função que produz o valor **void (vazio, inútil, à-toa)**. Neste caso, ela não precisa do comando **return**.



Escrevendo um programa em C

- Note que os parâmetros da função **main** também não foram usados neste caso.
- Portanto, podemos também indicar com **void** que a lista de parâmetros da função principal é vazia.
- Assim, podemos ter outras formas para o programa:

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

void main(void)
{
    float y;
    y = sin(1.5);
    printf("y = %f", y);
    printf("\n");
    system("PAUSE");
    return;
}
```

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

void main(void)
{
    float y;
    y = sin(1.5);
    printf("y = %f", y);
    printf("\n");
    system("PAUSE");
}
```



Trabalho

- Fazer um relatório/resumo sobre o vídeo "**Craqueando o Stuxnet, uma cyber-arma do século 21**" de Ralph Langner (Ralph Langner: Cracking Stuxnet, a 21st-century cyber weapon)
- Vídeo disponível em:
[http://www.ted.com/talks/ralph_langner_cracking_stuxnet_a_21s
t_century_cyberweapon](http://www.ted.com/talks/ralph_langner_cracking_stuxnet_a_21st_century_cyberweapon)

ou em qualquer celular/smartphone/tablet com apps do TED ou acesso à Internet.

- Comentar sobre a “cyberguerra” e sobre segurança cibernética
- Enviar para o email edkallenn.lima@uninorteac.edu.br ou edkevan@gmail.com – título **[ED - Trab02 - Stuxnet]**
- Ralph Langner - Security consultant - Ralph Langner is a German control system security consultant. He has received worldwide recognition for his analysis of the Stuxnet malware.



**VER A LISTA DE
EXERCÍCIOS QUE ESTARÁ
DISPONÍVEL NO BLOG E NO
DROPBOX.**

