Um computador faz exatamente o que você manda ele fazer, mas isso pode ser muito diferente do que você gostaria que ele fizesse.

Introdução ao C - Parte 1

Paulo Ricardo Lisboa de Almeida

DSBD

UFPR
UNIVERSIDADE FEDERAL DO PARANÁ

Antes de começar

Serão apresentados exemplos para que você possa migrar de Pascal para C

Sem entrar em detalhes da linguagem

Conceitos aprofundados sobre C serão discutidos em disciplinas específicas

Instalando compilador

Existem compiladores C para as mais diversas arquiteturas e sistemas operacionais

Nos exemplos, vamos usar sistemas baseados em Linux

Instalando compilador

A maioria das distribuições Linux contam com as ferramentas necessárias para compilar programas C por padrão

Caso não tenha, você pode instalar abrindo um terminal e inserindo o comando

```
sudo apt install build-essential
```

Olá Mundo

Primeiro deve-se criar o arquivo que vai conter o **código fonte** do programa

Arquivos de código fonte em C devem ter a extensão .c

Abra o terminal

```
touch olaMundo.c
```

Agora basta editar o arquivo usando um editor de texto de sua preferência. Dicas:

```
subl olaMundo.c
```

```
xed olaMundo.c
```

```
vim olaMundo.c
```

subl (sublime) e o xed são editores de texto gráficos, e o vim é um editor de linha de comando.
Caso você não tenha esses editores instalados, basta instalar via os comandos:

```
sudo apt-get install sublime-text
sudo apt-get install xed
sudo apt-get install vim
```

Olá Mundo

Com o editor aberto, insira o seguinte

```c
#include<stdio.h>

int main(){
      printf("Ola mundo\n");
      return 0;
}
```

Olá Mundo

```c
#include<stdio.h>

int main(){
      printf("Ola mundo\n");
      return 0;
}
```

Em C, a diretiva **#include** inclui (importa) determinado arquivo. O arquivo **stdio.h** é uma biblioteca que possui as rotinas necessárias para realizar entrada e saída. A maioria dos nossos programas vai precisar importar stdio.h.

Os arquivos de biblioteca padrão geralmente estão no diretório `/usr/include` , mas você não precisa se preocupar com isso.

Olá Mundo

```c
#include<stdio.h>

int main(){
      printf("Ola mundo\n");
      return 0;
}
```

A **função main** em C é onde o seu programa começa. Todo programa precisa ter um main. Existem algumas formas diferentes de se criar o main, mas a mais comum (e a que vamos usar na disciplina) é:

```c
int main(){
    //coloque seu programa aqui
}
```

Olá Mundo

```c
#include<stdio.h>

int main(){
      printf("Ola mundo\n");
      return 0;
}
```

As **chaves** indicam o início e o fim de um **bloco**. No exemplo, o abre chave indica o início das instruções que estão dentro do main, e o final da chave indica o final das instruções que estão dentro do main. Ao abrir um novo bloco, **sempre adicione um novo nível de indentação**.

Olá Mundo

```c
#include<stdio.h>

int main(){
      printf("Ola mundo\n");
      return 0;
}
```

Em C, todo **comando** deve terminar com **;**

Olá Mundo

```c
#include<stdio.h>

int main(){
     printf("Ola mundo\n");
      return 0;
}
```

Use a função printf para imprimir algo na tela. Coloque o texto a ser impresso entre aspas duplas.
Formato do printf:

```
printf("texto");
```

O \n é um caractere especial que imprime uma quebra de linha.

Olá Mundo

```c
#include<stdio.h>

int main(){
     printf("Ola mundo\n");
     return 0;
}
```

O comando **return** retorna um valor de uma função. Nesse caso ele finaliza o main retornando 0. Vamos ver um pouco mais sobre funções e retornos na próxima aula. Por enquanto, sempre assuma que qualquer programa deve ter o seguinte:

```c
int main(){
     //seu programa aqui
     return 0;
}
```

Olá Mundo

Para compilar

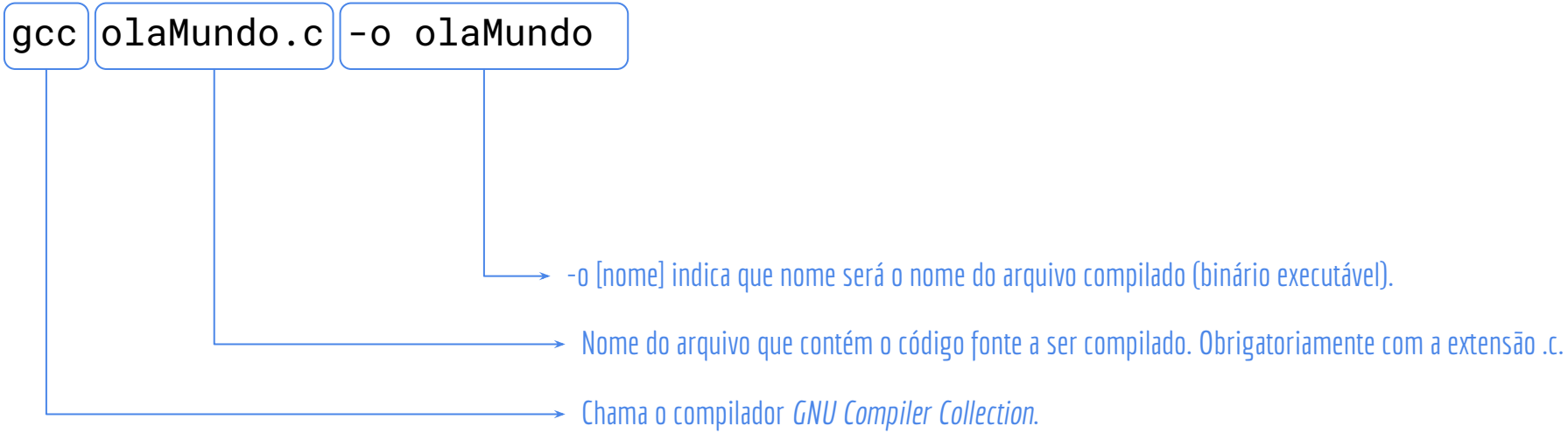
Salve o arquivo de código fonte, e execute o comando

```
gcc olaMundo.c -o olaMundo
```

Olá Mundo

Para compilar

Salve o arquivo de código fonte, e execute o comando



Olá Mundo

Depois de compilado, basta pedir para o sistema operacional executar o programa.

No Linux

```
./olaMundo
```

Variáveis

Para declarar uma variável, use

```
tipoVariavel nomeVariavel;
```

Você ainda pode declarar múltiplas variáveis de um mesmo tipo, na forma

```
tipoVariavel nomeVar1, nomeVar2, nomeVar3;
```

Variáveis

Para declarar uma variável, use

```
tipoVariavel nomeVariavel;
```

Alguns **tipos** possíveis de variáveis

int -› inteiro com sinal
char -› caractere
float -› ponto flutuante de precisão simples
double -› ponto flutuante de precisão simples

Veja uma lista com os possíveis tipos aqui: www.cplusplus.com/doc/tutorial/variables

Variáveis

Para declarar uma variável, use

```
tipoVariavel nomeVariavel;
```

Para **nomes de variáveis**, use nomes significativos, e em **camelCase**. Exemplos:

char nome;

int idade;

double precoProduto;

int numeroCasa, qtdeMoradores;

Imprimindo variáveis

Use o printf para imprimir o valor de variáveis na tela. Para isso, faça o printf normalmente, adicionando um %[formato] para imprimir a variável.

```
printf(“Um texto seguido do valor da var1 %[formato]” e da var2 %[formato]”, var1, var2)
```

Use o especificador de %*[formato]* correto para cada tipo de variável. Alguns exemplos:

%d para imprimir um inteiro com sinal

%c para imprimir um caractere

%f para imprimir um ponto flutuante

Veja uma lista completa em www.cplusplus.com/reference/cstdio/printf

Atribuindo valores

Para atribuir o valor a uma variável, utilize o operador =

Você pode atribuir um valor a uma variável logo quando ela é criada, ou posteriormente. Fica a seu critério.

Você deve colocar caracteres entre aspas simples.

Exemplos:

```
int meuValor = 10;

char caractere;

caractere = ‘c’;
```

Operadores

Os operadores básicos são

- `-` subtração
- `+` adição
- `*` multiplicação
- `/` divisão
- `%` módulo (resto da divisão)

Atenção

Variáveis não são inicializadas automaticamente

Isso seria perda de tempo. Se você precisa que uma variável possua determinado valor, atribua esse valor a ela

Nunca assuma que uma variável criada possuir automaticamente, por exemplo, o valor zero

Erro comum que leva a problemas sérios

Uma variável criada sem valor atribuído possui **lixo de memória**

Exemplo

```c
#include<stdio.h>

int main(){
      int meuInteiro = 10;
      char meuCaractere;
      float pi;

      meuCaractere = 'P';
      pi = 3.14;

      printf("Os valores sao:%d %c %f\n", meuInteiro, meuCaractere, pi);

      meuInteiro = meuInteiro + 1;
      printf("Somado 1\nAgora meuInteiro vale %d\n", meuInteiro);

      return 0;
}
```

Lendo do teclado

Pra ler do teclado, utilize a função

```
scanf("%[modificador] %[modificador] …", &variavel1, &variavel2, …);
```

Onde [modificador] são os mesmos do printf, e servem para informar se estamos lendo um inteiro, char, float, ...

O **&** antes de cada variável é **obrigatório**

Aprenda o motivo desse & nas disciplinas de oficina de programação

Exemplo

```c
#include<stdio.h>

int main(){
      int meuInteiro;
      float meuFloat;

      printf("Lendo um de cada vez: ");
      scanf("%d", &meuInteiro);
      scanf("%f", &meuFloat);
      printf("Voce digitou %d e %f\n", meuInteiro, meuFloat);

      printf("Lendo tudo junto: ");
      scanf("%d %f", &meuInteiro, &meuFloat);
      printf("Voce digitou %d e %f\n", meuInteiro, meuFloat);

      return 0;
}
```

Problema comum

Execute o programa a seguir. O que acontece de errado?

```c
#include<stdio.h>

int main(){
	int valor;
	char caractere;

	scanf("%d", &valor);
	scanf("%c", &caractere);

	printf("Você digitou %d e %c", valor, caractere);
	return 0;
}
```

Problema comum

Execute o programa a seguir. O que acontece de errado?

Pesquise na internet uma solução para esse problema.

```c
#include<stdio.h>

int main(){
	int valor;
	char caractere;

	scanf("%d", &valor);
	scanf("%c", &caractere);

	printf("Você digitou %d e %c", valor, caractere);
	return 0;
}
```

Problema comum

Execute o programa a seguir. O que acontece de errado?

Pesquise na internet uma solução para esse problema.

Provavelmente a internet (nossa fonte inesgotável de fake news e outras asneiras) indicou que você faça um *fflush(stdin)*

Esse é um **excelente exemplo do motivo de precisarmos tomar cuidado com a internet**

Cheia de pseudoespecialistas que não tem ideia do que estão fazendo

Problema comum

Nunca utilize fflush(stdin)

fflush é sim uma função muito útil, mas não para este fim

Você vai aprender mais sobre ela nas disciplinas focadas em C

Usar fflush(stdin) pode inviabilizar redirecionamentos de stream

O comportamento do fflush na entrada padrão (stdin) **não é definido**

O seu sistema operacional pode limpar o buffer, mas ele não é obrigado a fazer isso

www.cplusplus.com/reference/cstdio/fflush

Leia a Seção 7.19.5.2 da especificação da linguagem

www.open-std.org/jtc1/sc22/WG14/www/docs/n1256.pdf

Problema comum

Para resolver o problema, basta entendê-lo

Digitamos um valor e depois teclamos “enter”

O valor digitado foi armazenado em “valor”, mas o “enter” gerou uma quebra de linha (‘\n’), que não tinha para onde ir, e ficou armazenada no buffer

Quando você solicitou um novo caractere, o shell utilizou o \n que estava armazenado no buffer e o inseriu na variável *caractere*

Por isso um novo caractere não foi solicitado

Já tínhamos digitado um caractere, o \n do enter

```c
#include<stdio.h>

int main(){
	int valor;
	char caractere;

	scanf("%d", &valor);
	scanf("%c", &caractere);

	printf("Você digitou %d e %c", valor, caractere);
	return 0;
}
```

Problema comum

Solução

Note que foi adicionado um espaço depois do %d. A especificação do scanf diz que um espaço representa qualquer caractere de separação. Então o scanf vai ler o inteiro e armazenar em valor, e vai ler um caractere de separação (no nosso caso o \n gerado pelo enter) e jogá-lo fora.

```c
#include<stdio.h>

int main(){
      int valor;
      char caractere;

      scanf("%d ", &valor);
      scanf("%c", &caractere);

      printf("Você digitou %d e %c", valor, caractere);
      return 0;
}
```

Um caractere de separação pode ser, por exemplo, um espaço, um tab, ou uma quebra de linha.

Condicionais

Condicionais em C podem ser feitas com ifs, que podem ou não serem seguidos de elses. Formato:

```
if(expressao){
     //executa caso a expressão seja verdade
}else{
     //executa caso não seja verdade
}
```

Condicionais

Condicionais em C podem ser feitas com ifs, que podem ou não serem seguidos de elses. Formato:

```
if(expressao){
    //executa caso a expressão seja verdade
}else{
    //executa caso não seja verdade
}
```

`expressao` deve ser uma expressão lógica. Os comparadores lógicos básicos são

>	maior que	<=	menor igual a
<	menor que	==	igual
>=	maior igual a	!=	diferente

Condicionais

Os comparadores lógicos básicos são

>	maior que	<=	menor igual a
<	menor que	==	igual
>=	maior igual a	!=	diferente

Os conectivos lógicos básicos são

||	ou (são dois | juntos - chamamos de pipes)	&&	e
!	negação		

Comentários

Inicie um comentário de uma linha usando //

Comentário de múltiplas linhas são no formato

/* começa comentário

termina comentário */

Exemplo

```c
#include<stdio.h>

int main(){
	int valor1 = 10;
	char valor2 = 20;

	//entre no if se o resultado NAO valor1 IGUAL a valor2 E
	//valor2 for maior ou igual a 20
	if(!(valor1 == valor2) && valor2 >= 20){
		printf("Executou o if\n");
	}else{
		printf("Executou o else\n");
	}

	return 0;
}
```

Loop While

O loop while tem o seguinte formato

```
while(expressao){
     corpo do while
}
```

Onde `expressao` é qualquer expressão lógica válida

Exemplo

```c
#include<stdio.h>

int main(){
	int valor = 5;

	while(valor > 1){
		printf("Valor: %d\n", valor);
		valor--;//atalho para valor = valor -1
	}
	return 0;
}
```

Loop for

O loop for tem o seguinte formato

```
for(inicializacao;expressao;passo){
    corpo do while
}
```

`inicializacao` é o valor inicial dado as variáveis (de controle por exemplo). Em versões mais recentes do C, você pode declarar variáveis em `inicializacao` se necessário.

`expressao` é qualquer expressão lógica válida

`passo` é a atualização feita nas variáveis (de controle por exemplo) a cada iteração do laço. O passo não é executado na primeira iteração.

Os campos `inicializacao, expressao e passo` são opcionais e podem ficar em branco

Exemplo

```c
#include<stdio.h>

int main(){
      //i++ é um atalho para i = i + 1
      for(int i=0; i < 10; i++){
            printf("%d\n", i);
      }

      return 0;
}
```

Exercícios

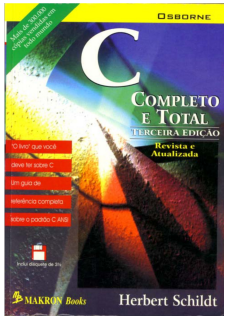
1. Crie um programa que recebe como entrada um número em segundos, e gera como resposta esse número convertido para horas, minutos e segundos no formato hh:mm:ss.
2. Leia um número n do teclado, e imprima o n-ésimo número da sequência de fibonacci na tela
3. Leia um número inteiro positivo do teclado e verifique se ele é palíndromo, ou seja, se a sua sequência de dígitos é exatamente a mesma, tanto se for analisado da esquerda para a direita quanto da direita para a esquerda.
4. Leia um número inteiro positivo e informe se este número tem a seguinte propriedade: ao ser elevado ao quadrado, gera um número que termina com os seus mesmos algarismos. O programa deve ler as entradas e gerar as saídas conforme os exemplos abaixo.

```
Digite um número inteiro: 5
5 tem a propriedade:
5*5=25, termina com 5
Digite um número inteiro: 76
76 tem a propriedade:
76*76=5776, termina com 76
Outro exemplo de execução:
Digite um número inteiro: 33
33 nao tem a propriedade:
33*33=1089, termina com 89
```

Referências

www.open-std.org/jtc1/sc22/WG14/www/docs/n1256.pdf

H. Schildt. C completo e total. 1996



Licença