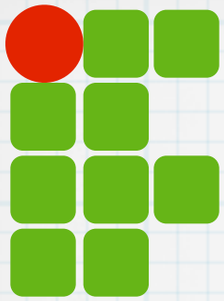
INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE

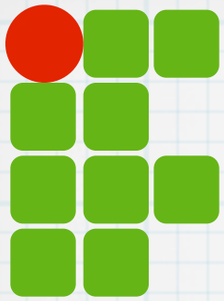
# Algoritmos

## ANSI C - Organização de um programa



# Agenda

* Escopo de variáveis
  * Local
  * Global
* Macros
  * `#define`
* Organizando o programa
  * Arquivos `.h`
  * Arquivos `.c`
  * A diretiva `#include`
  * Compilando tudo



# Escopo

* Escopo é um contexto delimitante aos quais valores e expressões estão associados.
  * Local
    * Variáveis declaradas em funções e/ou blocos de código
    * Visíveis até um fim da função/bloco
    * São armazenadas na pilha (stack)
  * Global
    * Variáveis de declaradas fora de funções
    * Visíveis em qualquer parte do programa
    * São armazenadas na área global

# Escopo local

* Criada no ponto de declaração
* Destruída no final da função/bloco onde foi declarada

```
float calculaMedia(float a, float b){
  float soma,media;
  soma = a+b;
  media = soma/2;
  return media;
}
```

# Escopo local

* Criada no ponto de declaração
* Destruída no final da função/bloco onde foi declarada

```
float calculaMedia(float a, float b){
  float soma,media;
  soma = a+b;
  media = soma/2;
  return media;
}
```

As variáveis soma e media só podem ser usada DENTRO da função calcularMédia

# Escopo local

* Criada no ponto de declaração
* Destruída no final da função/bloco onde foi declarada

```
float calculaMedia(float a, float b){
  float soma,media;
  soma = a+b;
  media = soma/2;
  return media;
}
```

As variáveis soma e media só podem ser usada DENTRO da função calcularMédia

# Escopo local

* Criada no ponto de declaração
* Destruída no final da função/bloco onde foi declarada

```
float calculaMedia(float a, float b){
  float soma,media;
  soma = a+b;
  media = soma/2;
  return media;
}
```

As variáveis soma e media só podem ser usada DENTRO da função calcularMédia

Final da validade das variáveis locais

# Escopo global

* Declaradas FORA de qualquer função
* É visível/válida para toda função APÓS a declaração
  * Todo o programa

```c
int umaFuncao(int x, int y, int z, int w);

float calculaMedia(float a, float b){
  float soma;
  soma = (a+b)*umaFuncao(1,2,3,4);
  return soma/2;
}

int x1;

int umaFuncao(int x, int y, int z, int w){
  int ret=0;
  x1=x*y+w;
  /* funcao qualquer */
  return ret;
}

void outraFuncao(int a, int b, int c[10]){
  /* Outra função qualquer */
  return x1*a/c[b];
}
```

# Escopo global

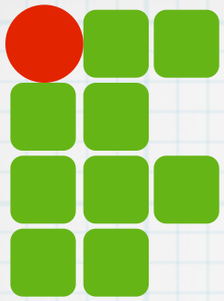
* Declaradas FORA de qualquer função
* É visível/válida para toda função APÓS a declaração
  * Todo o programa

A variável "x1" pode ser usada em qualquer bloco de código definido APÓS a sua declaração

```c
int umaFuncao(int x, int y, int z, int w);

float calculaMedia(float a, float b){
  float soma;
  soma = (a+b)*umaFuncao(1,2,3,4);
  return soma/2;
}

int x1;

int umaFuncao(int x, int y, int z, int w){
  int ret=0;
  x1=x*y+w;
  /* funcao qualquer */
  return ret;
}

void outraFuncao(int a, int b, int c[10]){
  /* Outra função qualquer */
  return x1*a/c[b];
}
```

# Observações

* Variáveis locais e globais ficam em diferentes áreas de memória
  * Não detalharemos aqui
* O uso de variáveis locais é mais indicado para manter o código limpo



# Macro

* Uma macro é um fragmento de código com um nome
* Onde o nome for usado é substituído, ANTES da compilação, pelo trecho de código definido
* Pode receber argumentos
* Usamos a diretiva #define
  * #define nome_macro [valor]
* Não possui ponto-e-vírgula (;) no final

# Macros

- Definir constantes
  - `#define PI 3.1415`
- Em qualque ponto do código onde PI for usado, será substiruído por 3.1415

```
#define PI 3.1415
#define TAMANHO 1000
#define G 9.8
...
int main(int argc, char **argv) {
    double x,y,z,r,v,v0,t;
    ...
    x = PI*r*r;
    v = v0+G*t;
    ...
    return 0;
}
```

# Macros

- Definir constantes
  - `#define PI 3.1415`
- Em qualque ponto do código onde PI for usado, será substiruído por 3.1415

Definição

```
#define PI 3.1415
#define TAMANHO 1000
#define G 9.8
...
int main(int argc, char **argv) {
    double x,y,z,r,v,v0,t;
    ...
    x = PI*r*r;
    v = v0+G*t;
    ...
    return 0;
}
```

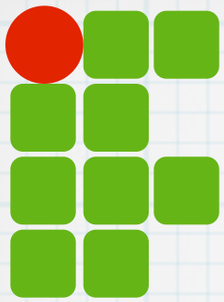
# Macros

- Definir constantes
  - `#define PI 3.1415`
- Em qualque ponto do código onde PI for usado, será substiruído por 3.1415

```
#define PI 3.1415
#define TAMANHO 1000
#define G 9.8
...
int main(int argc, char **argv) {
    double x,y,z,r,v,v0,t;
    ...
    x = PI*r*r;
    v = v0+G*t;
    ...
    return 0;
}
```

Definição

Uso



# Macros

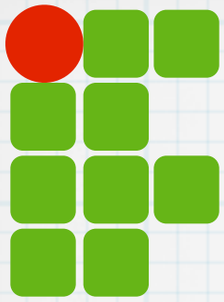
- Com argumentos
  - `#define token(<arg1>,arg2,...) comandos`
- Exemplo
  - `#define SOMA(A,B) A+B`

```
#define SOMA(A,B) A+B

int main(int argc, char **argv) {
  double x,y,r,v,v0,t;
  ...
  t = SOMA(x,y);
  ..
  return 0;
}
```

Esta linha é substituída por

`t = x+y ;`

ANTES do código ser compilado



# Macros

* Cuidado com argumentos

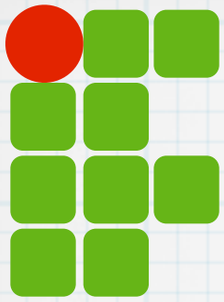
```c
#define SOMA(A,B) A+B

int main(int argc, char **argv) {
  double x,y,r,v,v0,t;
  ...
  t = 10*SOMA(x,y);
  ...
  return 0;
}
```

# Macros

* Cuidado com argumentos

```
#define SOMA(A,B) A+B

int main(int argc, char **argv) {
  double x,y,r,v,v0,t;
  ...
  t = 10*SOMA(x,y);
  ...
  return 0;
}
```

```
t = 10*x+y;
```

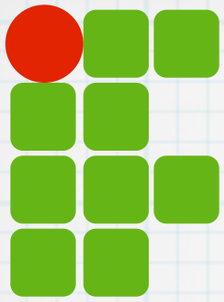


# Macros

* Cuidado com argumentos
  * Colocar parênteses

```c
#define SOMA(A,B) (A+B)

int main(int argc, char **argv) {
  double x,y,r,v,v0,t;
  ...
  t = 10*SOMA(x,y);
  ...
  return 0;
}
```

→ `t = 10*(x+y);`

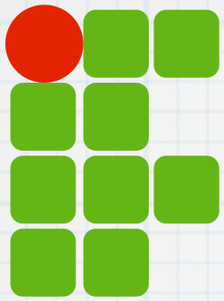


# Macros

* Cuidado com argumentos
  * Colocar parênteses

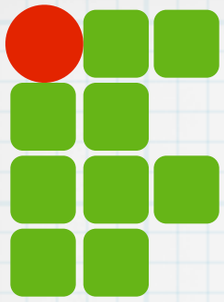
```
#define SOMA(A,B) (A+B)

int main(int argc, char **argv) {
  double x,y,r,v,v0,t;
  ...
  t = 10*SOMA(x,y);
  ...
  return 0;
}
```

```
t = 10*(x+y);
```



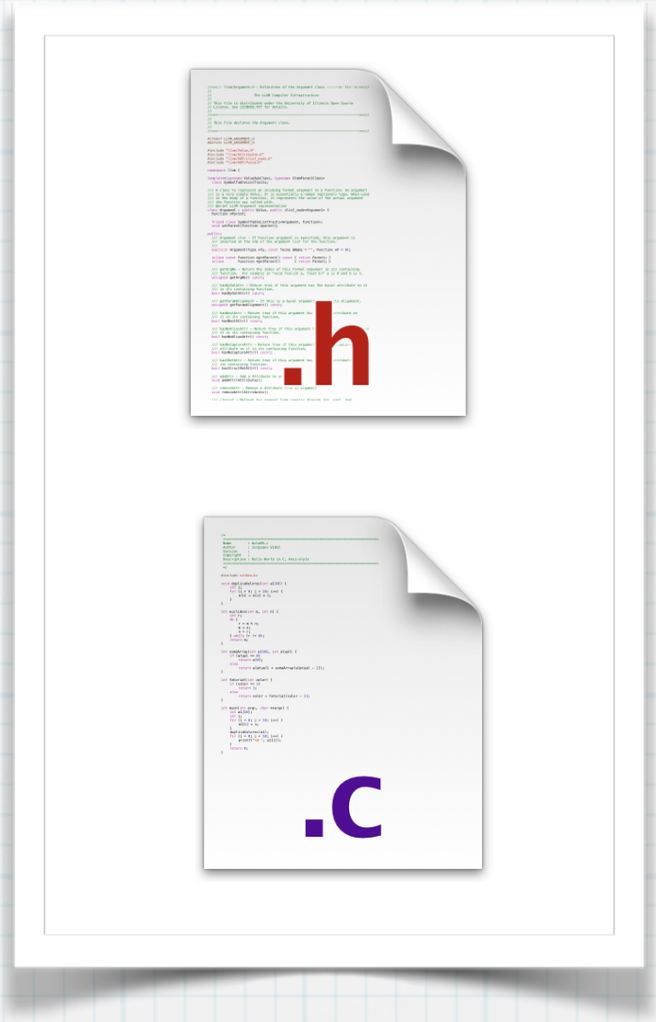
# Organização de um programa

* Programas podem conter muitas funções
* Muitas linhas de códigos
* Complexidade a ser gerenciada
* Podemos separar o programa em vários arquivos
  * O compilador agrupa todos os arquivos no executável



# Tipos de arquivo

* Em C temos, basicamente, dois tipos de arquivos
  * Arquivos de cabeçalho
    * Extensão .h
  * Arquivos de código
    * Extensão .c



# Arquivos de cabeçalho

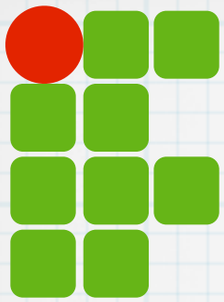
- Inclui arquivos de cabeçalho
- Contém a definição das funções
- Contém a definição das variáveis globais


```c
#ifndef GEOMETRIA_H_
#define GEOMETRIA_H_

#include <stdlib.h>

#define PI 3.1415
#define AREA_CIRCULO(R) (PI*R*R)

double seno(double angulo);

double cosseno(double angulo);

double tangente(double angulo);


#endif /* GEOMETRIA_H_ */
```

# Arquivos de código

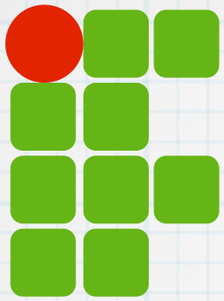
* Inclui arquivos de cabeçalho
* Contém a implementação das funções

```c
#include <stdio.h>
#include <stdlib.h>
...
int func1(args) {
    ...
    return r;
}
...
int func2(args) {
    ...
    return 0;
}
```



# Diretiva #include

- Duas formas
  - `#include <arquivo>`
    - Lê `arquivo` de uma lista de diretório configurados pela compilador
  - `#include “arquivo”`
    - Lê `arquivo` no mesmo diretório do arquivo sendo compilado
- Inclui o conteúdo de `arquivo` substituindo a linha da diretiva



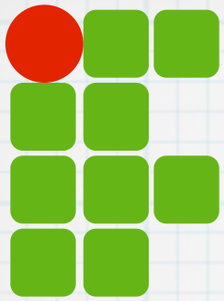
# Diretiva #include

geometria.h

```
#ifndef GEOMETRIA_H_
#define GEOMETRIA_H_

double areaCirculo(double raio);

double circunferencia(double raio);

double seno(double angulo);

double cosseno(double angulo);

double tangente(double angulo);

#endif
```

geometria.c

```
#include "geometria.h"

const double PI = 3.1415;

double areaCirculo(double raio){
     double area;
     area = PI * raio * raio;
     return area;
}

double circunferencia(double raio){
     double c;
     c = 2 * PI * raio;
    return c;
}
```



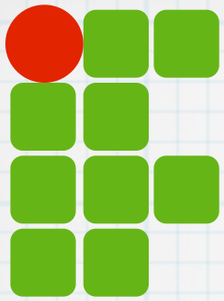
# Diretiva #include

geometria.h

```
#ifndef GEOMETRIA_H_
#define GEOMETRIA_H_

double areaCirculo(double raio);

double circunferencia(double raio);

double seno(double angulo);

double cosseno(double angulo);

double tangente(double angulo);

#endif
```

geometria.c

```
#include "geometria.h"

const double PI = 3.1415;

double areaCirculo(double raio){
     double area;
     area = PI * raio * raio;
     return area;
}

double circunferencia(double raio){
     double c;
     c = 2 * PI * raio;
    return c;
}
```



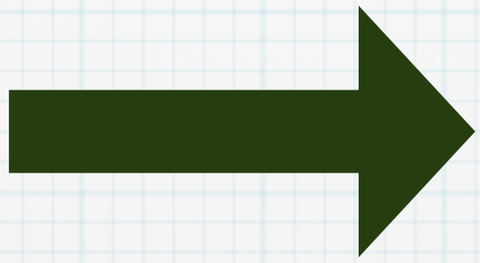
# Diretiva #include

**geometria.h**

```
#ifndef GEOMETRIA_H_
#define GEOMETRIA_H_

double areaCirculo(double raio);

double circunferencia(double raio);

double seno(double angulo);

double cosseno(double angulo);

double tangente(double angulo);

#endif
```

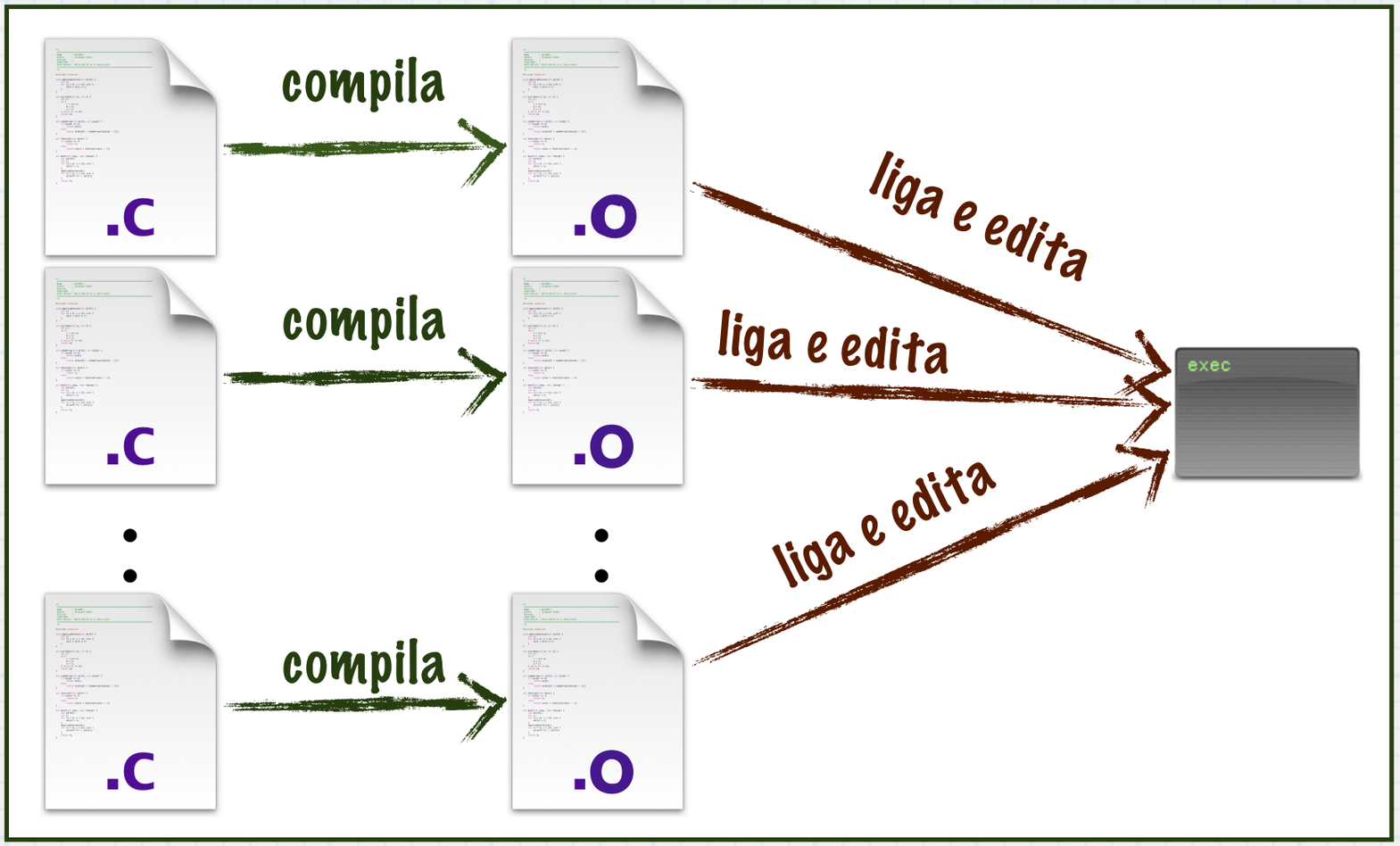
**geometria.c**

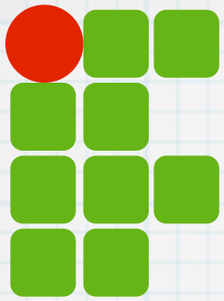
```
#include "geometria.h"

const double PI = 3.1415;

double areaCirculo(double raio){
     double area;
     area = PI * raio * raio;
     return area;
}

double circunferencia(double raio){
     double c;
     c = 2 * PI * raio;
    return c;
}
```



```
#ifndef GEOMETRIA_H_
#define GEOMETRIA_H_

double areaCirculo(double raio);

double circunferencia(double raio);

double seno(double angulo);

double cosseno(double angulo);

double tangente(double angulo);

#endif

const double PI = 3.1415;

double areaCirculo(double raio){
     double area;
     area = PI * raio * raio;
     return area;
}

double circunferencia(double raio){
     double c;
     c = 2 * PI * raio;
    return c;
}
```

# Compilação


compila
.c
.o
compila
.c
.o
compila
.c
.o
liga e edita
liga e edita
liga e edita
exec

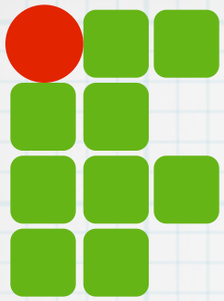


# Compilação

* Todos os arquivos de uma só vez

```
gcc -Wall -ansi -o exec arq01.c arq02.c
arq03.c ...
```

* Compila a lista de arquivos e gera o arquivo "exec"
* Deve existir uma função "main" implementada em um dos arquivos



# Compilação

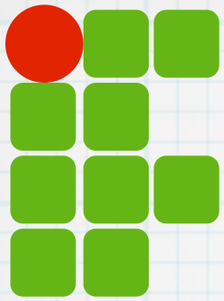
* Arquivos separados e geração do executável

```
gcc -Wall -ansi -c arq01.c
```

* Gera um arquivo .o que não é executável

```
gcc -o exec arq01.o arq02.o arq03.o ...
```

* Gera "exec" a partir dos vários arquivos compilados
* Em pelo menos um arquivo deve existir a função "main"



# Exemplo

geometria.h

```
#ifndef GEOMETRIA_H_
#define GEOMETRIA_H_

double areaCirculo(double raio);

double circunferencia(double raio);

double seno(double angulo);

double cosseno(double angulo);

double tangente(double angulo);

#endif
```

geometria.c

```
#include "geometria.h"

const double PI = 3.1415;

double areaCirculo(double raio){
     double area;
     area = PI * raio * raio;
     return area;
}

double circunferencia(double raio){
     double c;
     c = 2 * PI * raio;
    return c;
}
```

```
#include <stdio.h>

#include "geometria.h"

int main(int argc, char **argv) {
  double raio;
  scanf ("%f",&raio);
  printf("A area do circulo e %f \n",areaCirculo(raio));
  printf("A circunferencia e %f \n",circunferencia(raio));
  return 0;
}
```

```
Terminal — bash — 80×15
Jorgiano:src jorgiano$ ls
geometria.c geometria.h main.c
Jorgiano:src jorgiano$ gcc -Wall -ansi -o circulo main.c geometria.c
Jorgiano:src jorgiano$ ls
circulo     geometria.c geometria.h main.c
Jorgiano:src jorgiano$
```

# Exemplo

**geometria.h**

```c
#ifndef GEOMETRIA_H_
#define GEOMETRIA_H_

double areaCirculo(double raio);

double circunferencia(double raio);

double seno(double angulo);

double cosseno(double angulo);

double tangente(double angulo);

#endif
```

**geometria.c**

```c
#include "geometria.h"

const double PI = 3.1415;

double areaCirculo(double raio){
     double area;
     area = PI * raio * raio;
     return area;
}

double circunferencia(double raio){
     double c;
     c = 2 * PI * raio;
    return c;
}
```

```c
#include <stdio.h>

#include "geometria.h"

int main(int argc, char **argv) {
  double raio;
  scanf ("%f",&raio);
  printf("A area do circulo e %f \n",areaCirculo(raio));
  printf("A circunferencia e %f \n",circunferencia(raio));
  return 0;
}
```

```
Terminal — bash — 80×15
Jorgiano:src jorgiano$ ls
geometria.c geometria.h main.c
Jorgiano:src jorgiano$ gcc -Wall -ansi -o circulo main.c geometria.c
Jorgiano:src jorgiano$ ls
circulo     geometria.c geometria.h main.c
Jorgiano:src jorgiano$
```

# Exemplo

**geometria.h**

```
#ifndef GEOMETRIA_H_
#define GEOMETRIA_H_

double areaCirculo(double raio);

double circunferencia(double raio);

double seno(double angulo);

double cosseno(double angulo);

double tangente(double angulo);

#endif
```

**geometria.c**

```
#include "geometria.h"

const double PI = 3.1415;

double areaCirculo(double raio){
      double area;
      area = PI * raio * raio;
      return area;
}

double circunferencia(double raio){
      double c;
      c = 2 * PI * raio;
     return c;
}
```

```
#include <stdio.h>

#include "geometria.h"

int main(int argc, char **argv) {
  double raio;
  scanf ("%f",&raio);
  printf("A area do circulo e %f \n",areaCirculo(raio));
  printf("A circunferencia                    io));
  return 0;
}
```

Compila os dois arquivos de uma vez

```
Terminal — bash — 80×15
Jorgiano:src jorgiano$ ls
geometria.c geometria.h main.c
Jorgiano:src jorgiano$ gcc -Wall -ansi -o circulo main.c geometria.c
Jorgiano:src jorgiano$ ls
circulo     geometria.c geometria.h main.c
Jorgiano:src jorgiano$
```

# Exemplo

**geometria.h**

```c
#ifndef GEOMETRIA_H_
#define GEOMETRIA_H_

double areaCirculo(double raio);

double circunferencia(double raio);

double seno(double angulo);

double cosseno(double angulo);

double tangente(double angulo);

#endif
```

**geometria.c**

```c
#include "geometria.h"

const double PI = 3.1415;

double areaCirculo(double raio){
     double area;
     area = PI * raio * raio;
     return area;
}

double circunferencia(double raio){
     double c;
     c = 2 * PI * raio;
    return c;
}
```

```c
#include <stdio.h>

#include "geometria.h"

int main(int argc, char **argv) {
  double raio;
  scanf ("%f",&raio);
  printf("A area do circulo e %f \n",areaCirculo(raio));
  printf("A circunferencia                    io));
  return 0;
}
```

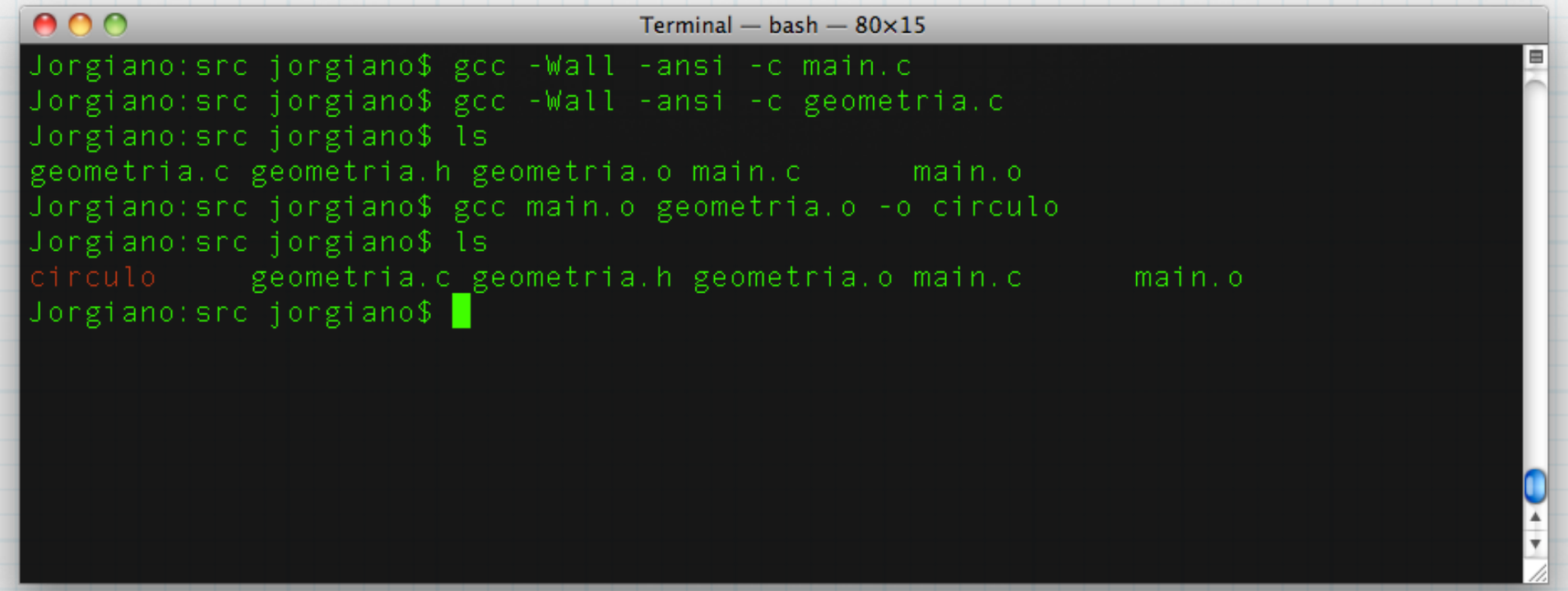
Compila os dois arquivos de uma vez

```
Terminal — bash — 80×15
Jorgiano:src jorgiano$ ls
geometria.c geometria.h main.c
Jorgiano:src jorgiano$ gcc -Wall -ansi -o circulo main.c geometria.c
Jorgiano:src jorgiano$ ls
circulo     geometria.c geometria.h main.c
Jorgiano:src jorgiano$
```

Gera um único executável

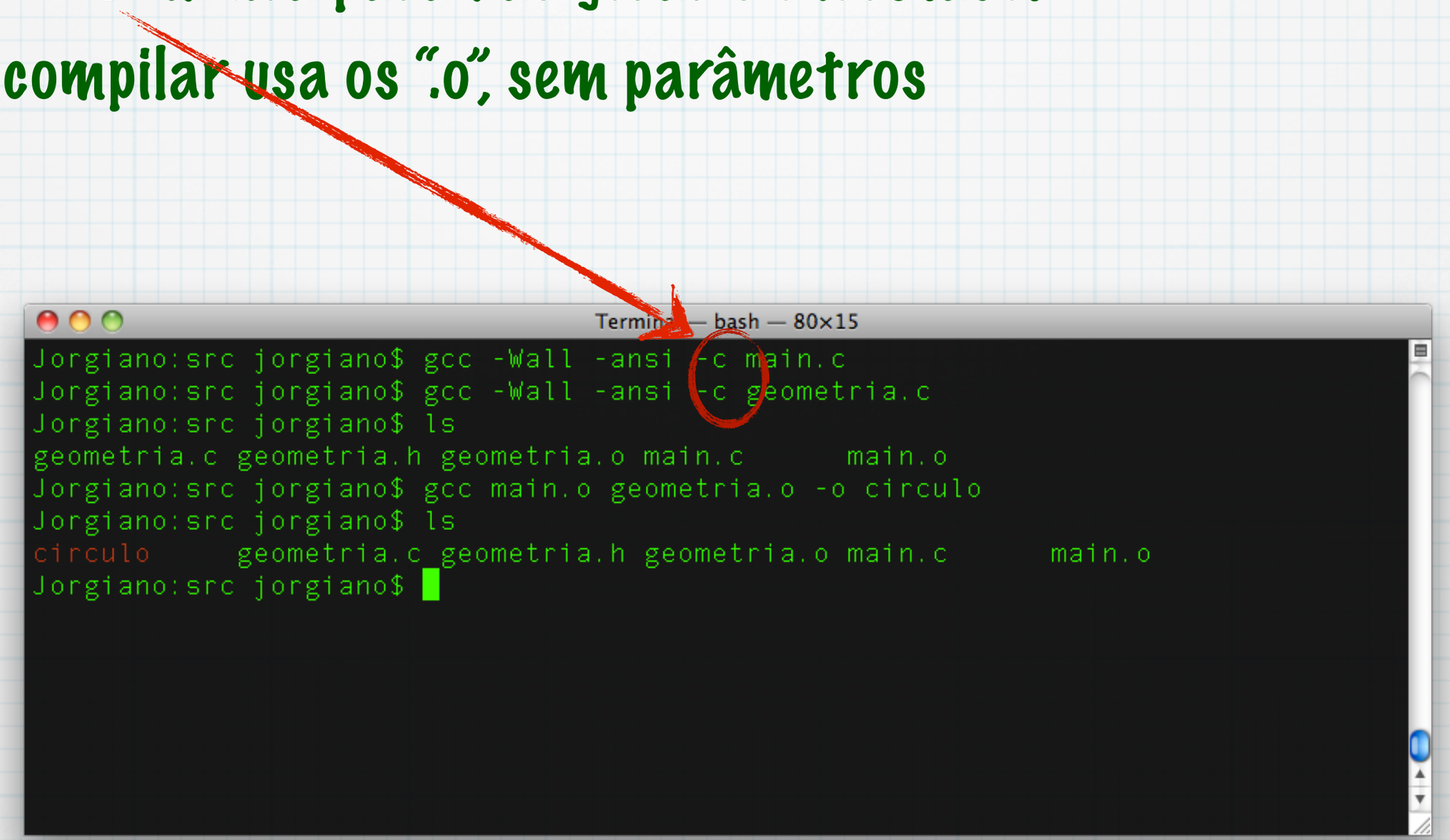
# Exemplo

* Compilar um arquivo por vez
  * opção "-c" indica para não gerar o executável
  * para compilar usa os ".o", sem parâmetros



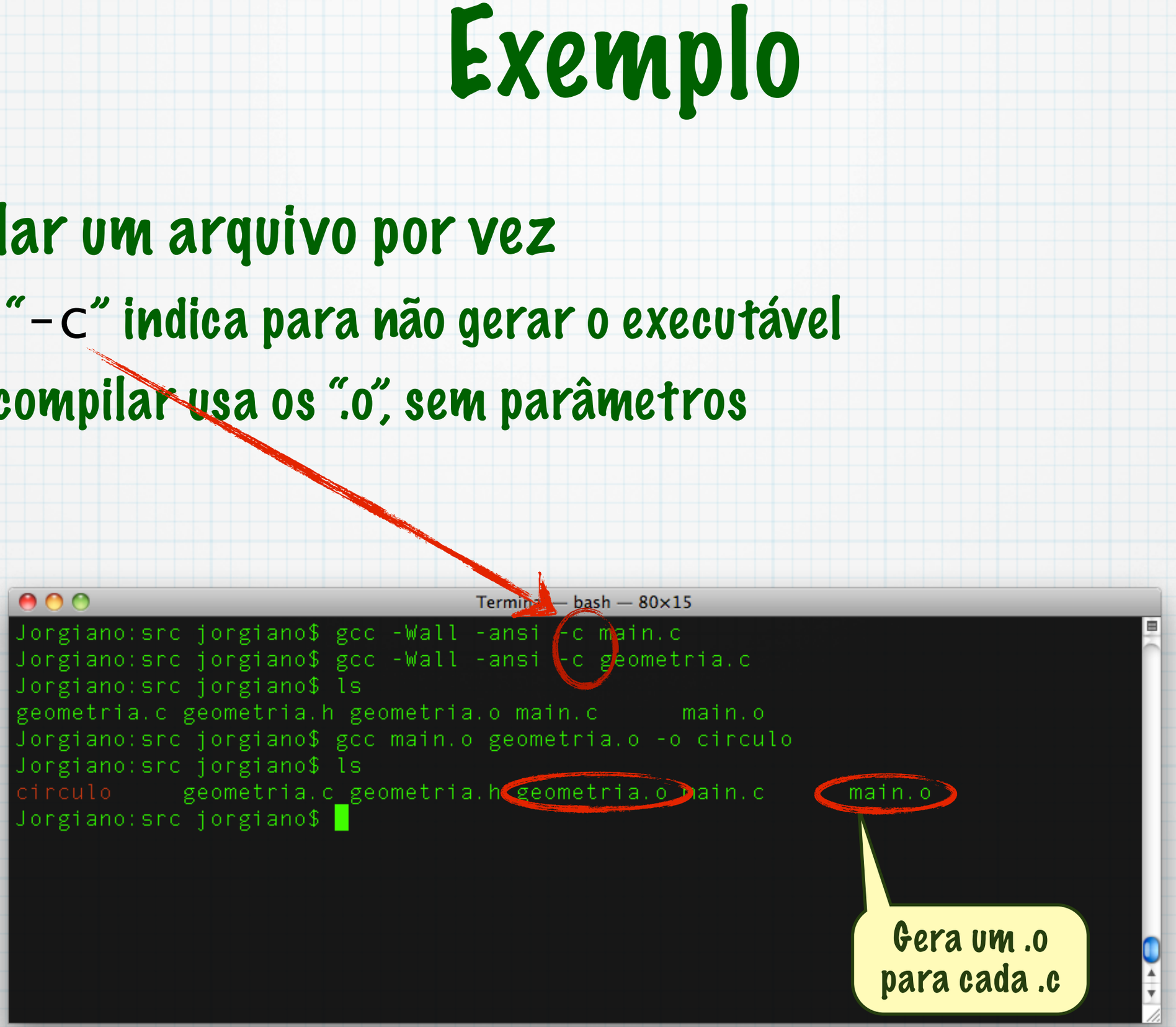
# Exemplo

* Compilar um arquivo por vez
  * opção "-c" indica para não gerar o executável
  * para compilar usa os ".o", sem parâmetros



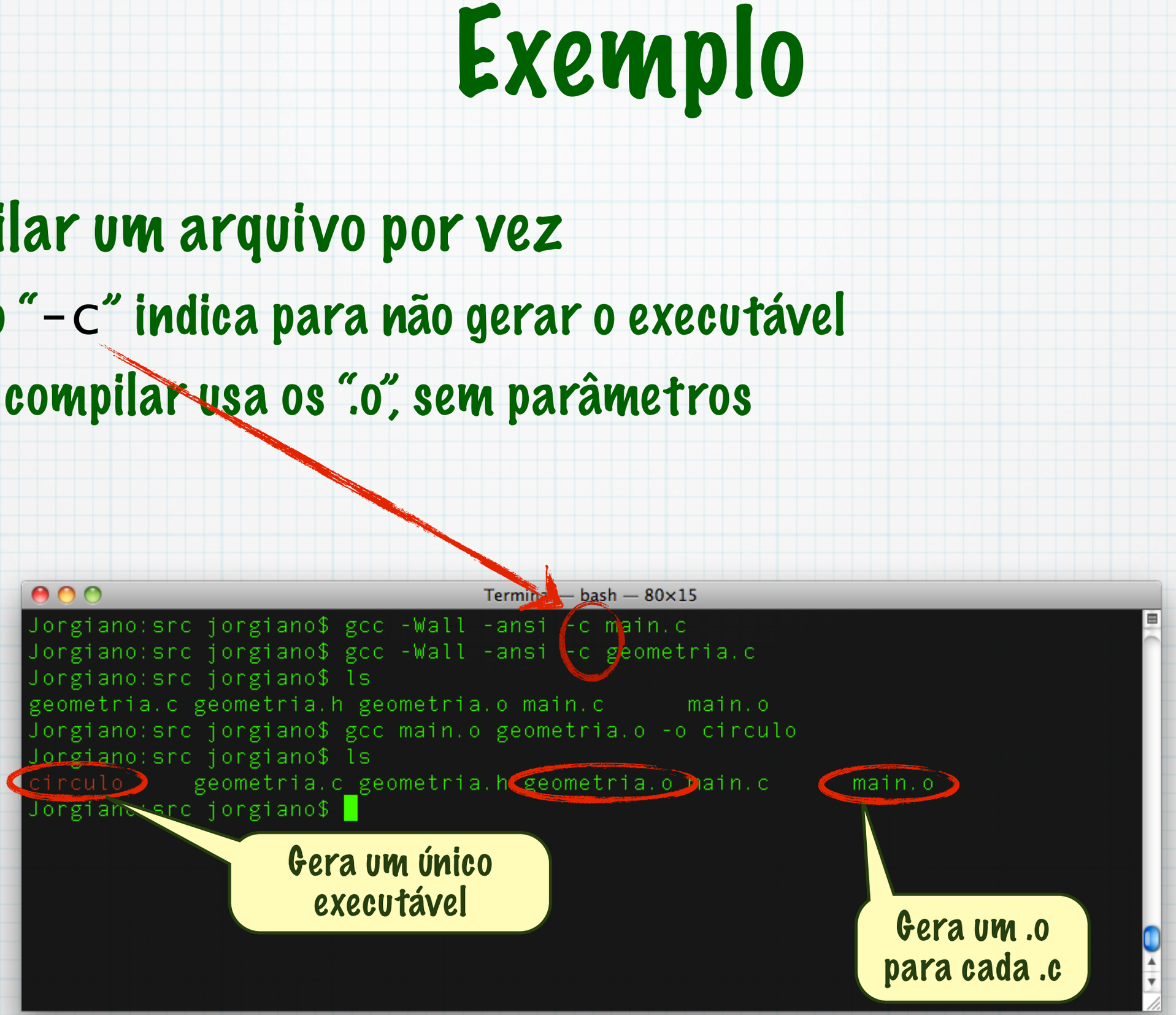
# Exemplo

* Compilar um arquivo por vez
  * opção "-c" indica para não gerar o executável
  * para compilar usa os ".o", sem parâmetros



# Exemplo

* Compilar um arquivo por vez
  * opção "-c" indica para não gerar o executável
  * para compilar usa os ".o", sem parâmetros



# Conclusão

* Separar as funções em vários arquivos organiza o programa
* Fácil de reusar código através de bibliotecas de funções
* Fácil de testar funções
* Melhorar divisão de tarefas para o desenvolvimento

# Dúvidas?