Programação

C++

Aula 01

Jorgiano Vidal

Agenda

- Sobre o instrutor
- Sobre o curso
- Quem usa C++?
- Por que estudar C++?
- O que você precisa ter?
- Histórico
- Filosofia do C++
- Olá mundo!
- Variáveis
- Tipos primitivos
- Entrada e saída básica
- Conversão implícita
- Conversão explícita
- Operadores aritméticos

Sobre o instrutor

- Jorgiano Vidal
- jorgiano.vidal@escolar.ifrn.edu.br

Sobre o curso

- Para programadores
- Detalhes da linguagem
- Algoritmos podem ser abordados
- Implementação OO (se houver tempo)

Quem usa?

- Amazon
- Intel
- IBM
- Facebook
- Google
- Apple
- Microsoft
- Adobe
- Chrome
- Firefox
- Opera
- Safari
- MS-Office
- Java
- Windows
- Photoshop
- Assassin's creed
- Call of Duty
- Starcraft
- Halo
- World of WarCraft
- Mass Effect

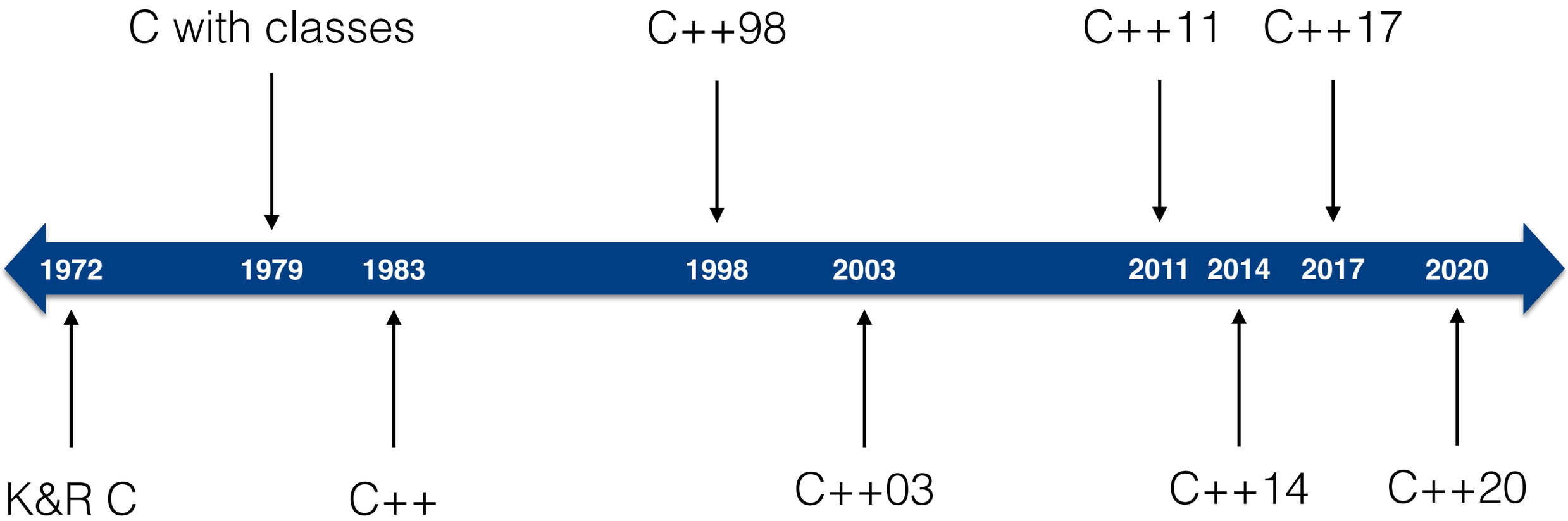
Por que?

- Popular
  - Tiobe Index: 4ª colocação, Atrás de Python, C e JAVA maio de 2023
- Eficiente
  - Gerenciamento de recursos (memória/processamento) otimizado
- Uso amplo
  - Sistemas embarcados
  - Jogos

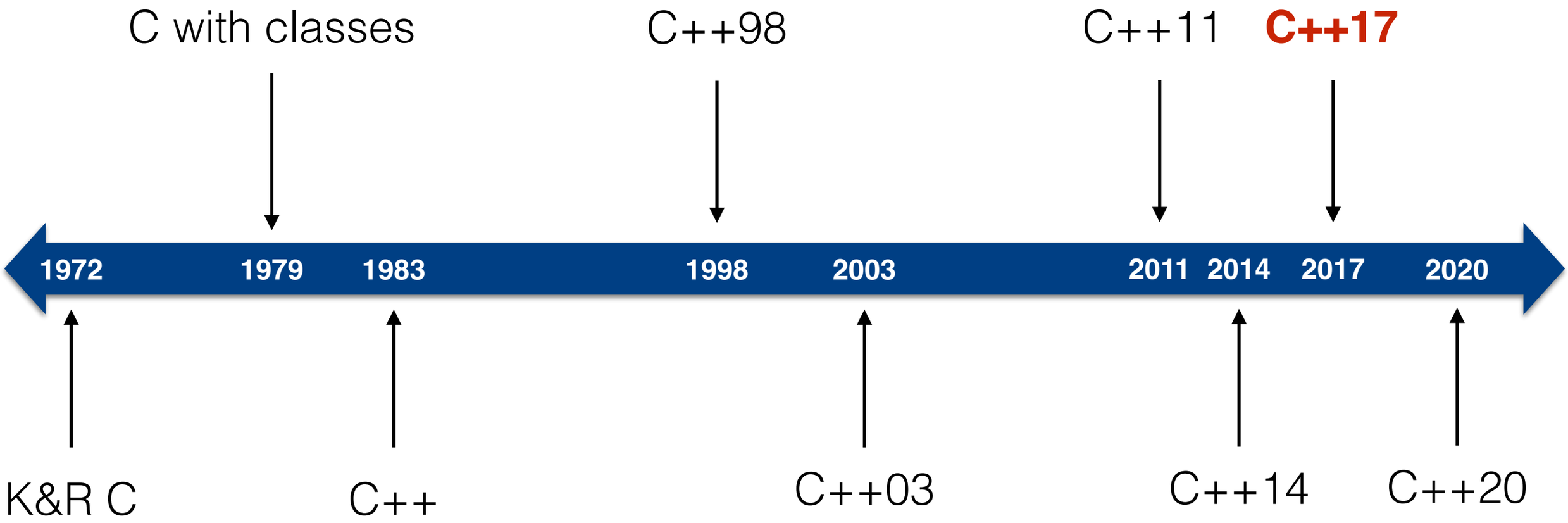
O que é necessário?

- Ferramentas
  - Compilador C++
  - Editor de texto
- Alternativa: IDE (*Integrated Development Environment*)
  - DevC++, Code::blocks, Visual Studio, CLion, Eclipse CDT, etc
- Usaremos:
  - Visual Studio Code:
    - https://code.visualstudio.com/
  - g++
    - https://gcc.gnu.org/

Histórico


C with classes
C++98
C++11
C++17
1972
1979
1983
1998
2003
2011
2014
2017
2020
K&R C
C++
C++03
C++14
C++20

Histórico


C with classes
C++98
C++11
C++17
1972
1979
1983
1998
2003
2011
2014
2017
2020
K&R C
C++
C++03
C++14
C++20

Histórico

Bjarne Stroustrup

Histórico

Bjarne Stroustrup

- Rápida
- Simples de usar
- Multi-plataforma
- **Funcionalidades de alto nível**

Histórico

Bjarne Stroustrup

- Rápida
- Simples de usar
- Multi-plataforma
- **Funcionalidades de alto nível**

Filosofia

Filosofia

- Somente adicionar nova característica se ela resolver um **problema real**

Filosofia

- Somente adicionar nova característica se ela resolver um **problema real**
- Programadores são livres para escolherem **estilo** de programação

Filosofia

- Somente adicionar nova característica se ela resolver um **problema real**
- Programadores são livres para escolherem **estilo** de programação
- Permitir **controle total** ao programador se ele desejar

Filosofia

- Somente adicionar nova característica se ela resolver um **problema real**
- Programadores são livres para escolherem **estilo** de programação
- Permitir **controle total** ao programador se ele desejar
- Não sacrificar **desempenho**

Filosofia

- Somente adicionar nova característica se ela resolver um **problema real**
- Programadores são livres para escolherem **estilo** de programação
- Permitir **controle total** ao programador se ele desejar
- Não sacrificar **desempenho**
- Forçar **segurança** em tempo de compilação sempre que possível

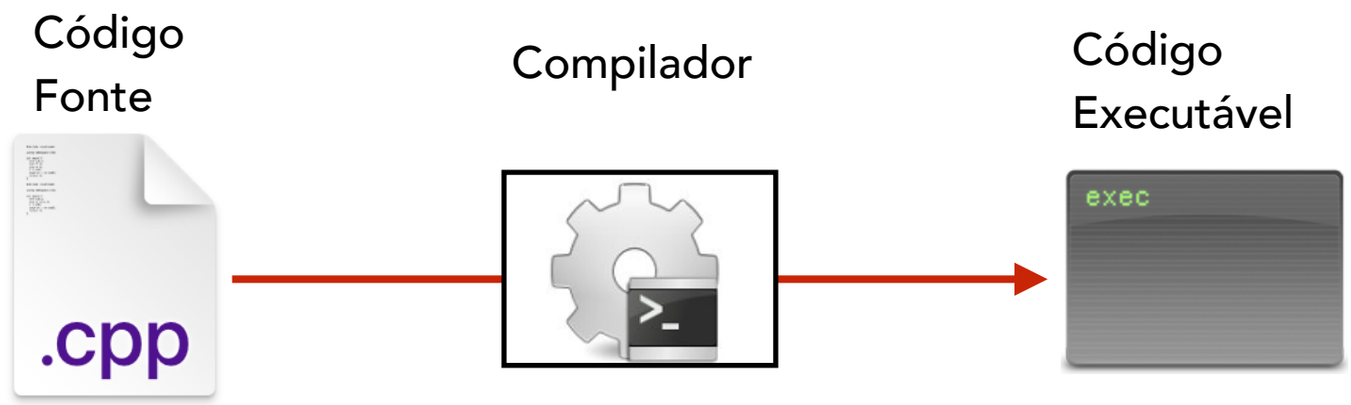
Olá mundo

```cpp
#include <iostream>

int main(){
   std::cout << "Olá mundo!" << std::endl;
   return 0;
}
```

```cpp
#include <iostream>

int main(){
   std::cout << "Olá mundo!\n";
   return 0;
}
```

Compilação e execução

- Transforma código fonte C++ em código executável em linguagem de máquina
- Necessário compilador
  - G++ - Gnu Compiler Collection (http://gcc.gnu.org)
  - clang++ - C++ Language (http://clang.llvm.org)
  - Existem outros



Compilação e execução

- Para compilar

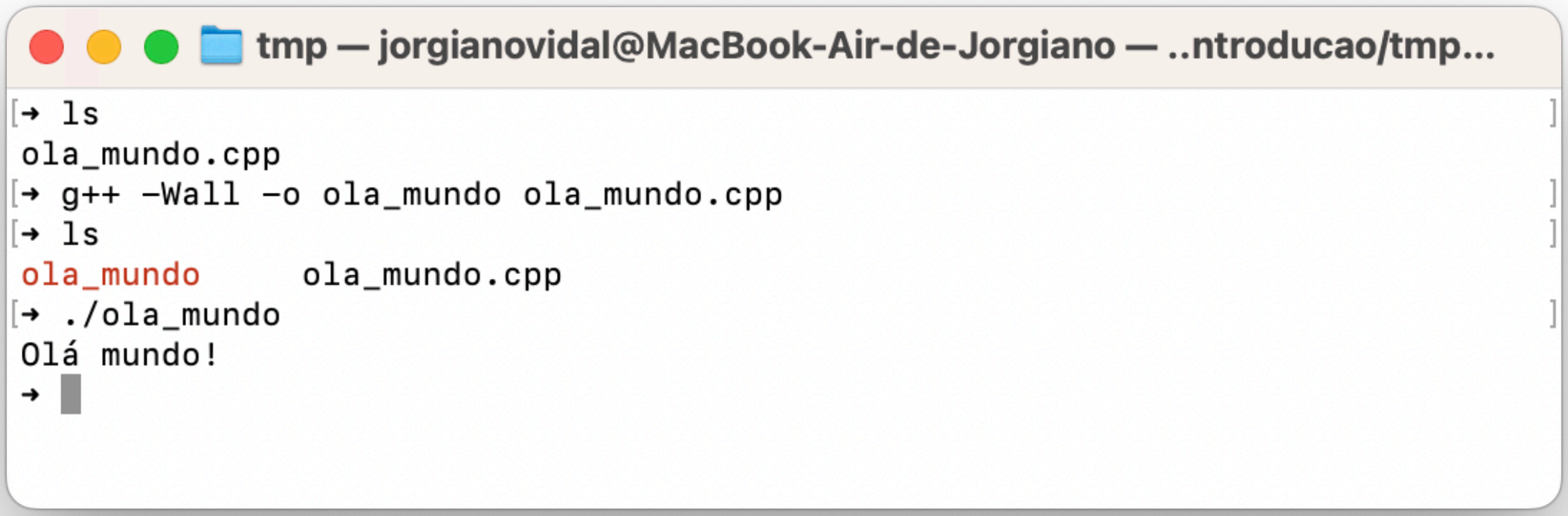
```
g++ -o ola_mundo ola_mundo.cpp
```

- Para executar

```
./olamundo
/home/jorgiano/cpp/ola_mundo
```

ola_mundo.cpp

```cpp
#include <iostream>

int main(){
  std::cout << "Olá mundo!\n";
  return 0;
}
```



Variáveis e atribuição

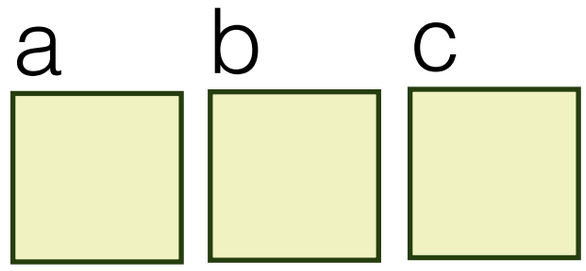
```cpp
#include <iostream>



int main(){
   int a,b;
   int c;
   a = 10;
   b = 20;
   c = a+b;
   std::cout << c << std::endl;
   return 0;
}
```

Variáveis e atribuição

```cpp
#include <iostream>



int main(){
   int a,b;
   int c;
   a = 10;
   b = 20;
   c = a+b;
   std::cout << c << std::endl;
   return 0;
}
```

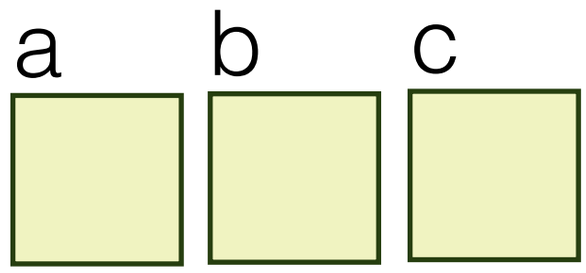
Variáveis e atribuição

```cpp
#include <iostream>



int main(){
   int a,b;
   int c;
   a = 10;
   b = 20;
   c = a+b;
   std::cout << c << std::endl;
   return 0;
}
```



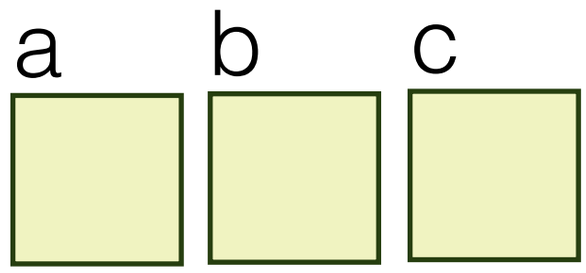
Variáveis e atribuição

```cpp
#include <iostream>



int main(){
   int a,b;
   int c;
   a = 10;
   b = 20;
   c = a+b;
   std::cout << c << std::endl;
   return 0;
}
```



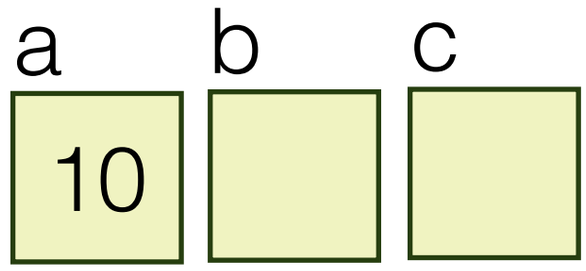
Variáveis e atribuição

```cpp
#include <iostream>



int main(){
   int a,b;
   int c;
   a = 10;
   b = 20;
   c = a+b;
   std::cout << c << std::endl;
   return 0;
}
```



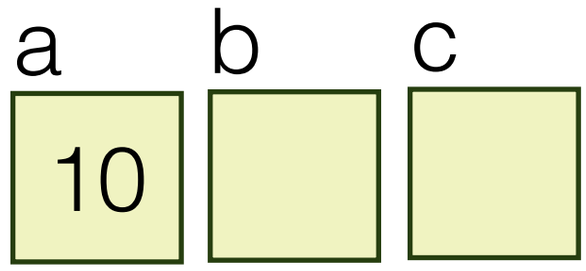
Variáveis e atribuição

```cpp
#include <iostream>



int main(){
   int a,b;
   int c;
   a = 10;
   b = 20;
   c = a+b;
   std::cout << c << std::endl;
   return 0;
}
```



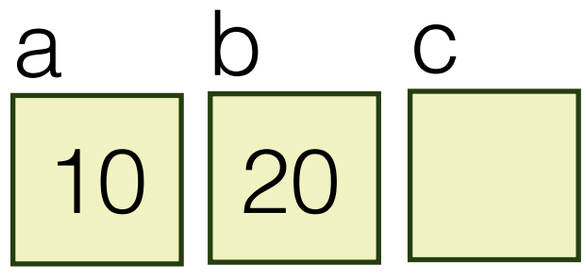
Variáveis e atribuição

```cpp
#include <iostream>



int main(){
   int a,b;
   int c;
   a = 10;
   b = 20;
   c = a+b;
   std::cout << c << std::endl;
   return 0;
}
```



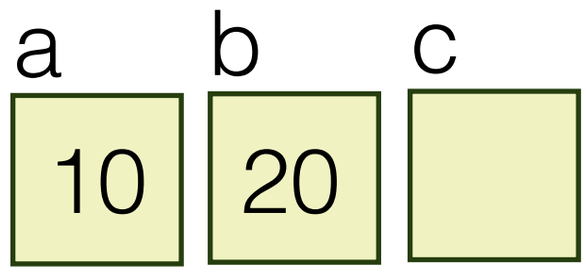
Variáveis e atribuição

```cpp
#include <iostream>



int main(){
   int a,b;
   int c;
   a = 10;
   b = 20;
   c = a+b;
   std::cout << c << std::endl;
   return 0;
}
```



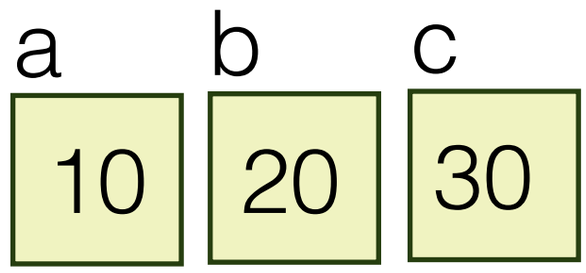
Variáveis e atribuição

```cpp
#include <iostream>



int main(){
   int a,b;
   int c;
   a = 10;
   b = 20;
   c = a+b;
   std::cout << c << std::endl;
   return 0;
}
```



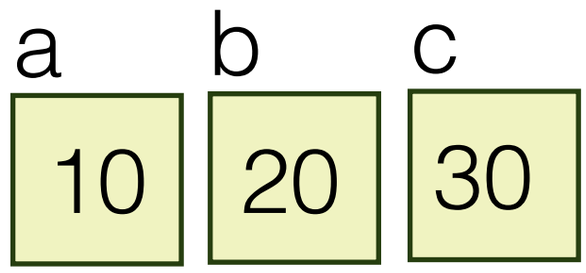
Variáveis e atribuição

```cpp
#include <iostream>



int main(){
   int a,b;
   int c;
   a = 10;
   b = 20;
   c = a+b;
   std::cout << c << std::endl;
   return 0;
}
```



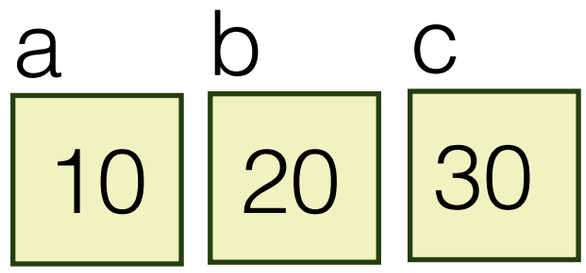
Variáveis e atribuição

```cpp
#include <iostream>



int main(){
   int a,b;
   int c;
   a = 10;
   b = 20;
   c = a+b;
   std::cout << c << std::endl;
   return 0;
}
```



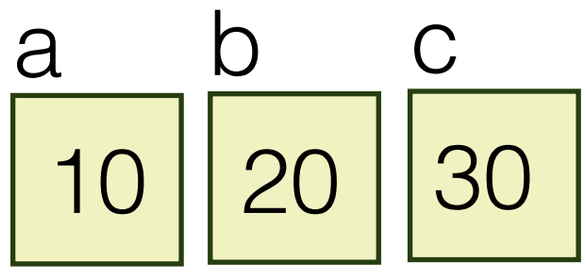
Variáveis e atribuição

```cpp
#include <iostream>



int main(){
   int a,b;
   int c;
   a = 10;
   b = 20;
   c = a+b;
   std::cout << c << std::endl;
   return 0;
}
```



Variáveis e atribuição

```cpp
#include <iostream>



int main(){
   int a,b;
   int c;
   a = 10;
   b = 20;
   c = a+b;
   std::cout << c << std::endl;
   return 0;
}
```



Entrada e saída

Entrada e saída

```cpp
#include <iostream>

int main(){
   int a,b,c;
   cin >> a;
   cin >> b;
   c = a+b;
   std::cout << c << std::endl;
   return 0;
}
```

Entrada e saída

```cpp
#include <iostream>

int main(){
    int a,b,c;
    cin >> a;
    cin >> b;
    c = a+b;
    std::cout << c << std::endl;
    return 0;
}
```

Entrada e saída

```cpp
#include <iostream>

int main(){
   int a,b,c;
   cin >> a;
   cin >> b;
   c = a+b;
   std::cout << c << std::endl;
   return 0;
}
```

Entrada e saída

```cpp
#include <iostream>

int main(){
  int a,b,c;
  cin >> a;
  cin >> b;
  c = a+b;
  std::cout << c << std::endl;
  return 0;
}
```

```cpp
#include <iostream>

int main(){
  int a,b,c;
  cin >> a >> b;
  c = a+b;
  std::cout << c << std::endl;
  return 0;
}
```

Tipos fundamentais

- O tipo vazio
  - Ausência de valor

```
void
```

- Lógico
  - `true` ou `false`

```
bool
```

- Caracter
  - Símbolo

```
char
```

- Inteiro
  - Arquitetura alvo

```
short   int   long     long long
```

- Real
  - IEEE-754

```
float     double
```

- Tipo ponteiro nulo
  - Sobrecarga

```
nullptr_t
```

Tipos fundamentais

- O tipo vazio
  - Ausência de valor

```
void
```

- Lógico
  - `true` ou `false`

```
bool
```

- Caracter
  - Símbolo

```
char
```

- Inteiro
  - Arquitetura alvo

```
short   int   long     long long
```

- Real
  - IEEE-754

```
float     double
```

- Tipo ponteiro nulo
  - Sobrecarga

```
nullptr_t
```

Tipos fundamentais

- O tipo vazio
  - Ausência de valor

```
void
```

- Lógico
  - `true` ou `false`

```
bool
```

- Caracter
  - Símbolo

```
char
```

- Inteiro
  - Arquitetura alvo

```
short   int   long     long long
```

- Real
  - IEEE-754

```
float     double
```

- Tipo ponteiro nulo
  - Sobrecarga

```
nullptr_t
```

Tipos fundamentais

- O tipo vazio
  - Ausência de valor
- Lógico
  - `true` ou `false`
- Caracter
  - Símbolo
- Inteiro
  - Arquitetura alvo
- Real
  - IEEE-754
- Tipo ponteiro nulo
  - Sobrecarga

```
void

bool

char

short   int   long     long long

float     double

nullptr_t
```

Tipos fundamentais

- O tipo vazio
  - Ausência de valor

```
void
```

- Lógico
  - `true` ou `false`

```
bool
```

- Caracter
  - Símbolo

```
char
```

- Inteiro
  - Arquitetura alvo

```
short   int   long     long long
```

- Real
  - IEEE-754

```
float     double
```

- Tipo ponteiro nulo
  - Sobrecarga

```
nullptr_t
```

Tipos fundamentais

- O tipo vazio
  - Ausência de valor

`void`

- Lógico
  - `true` ou `false`

`bool`

- Caracter
  - Símbolo

`char`

- Inteiro
  - Arquitetura alvo

`short` `int` `long` `long long`

- Real
  - IEEE-754

`float` `double`

- Tipo ponteiro nulo
  - Sobrecarga

`nullptr_t`

Tipos fundamentais

- O tipo vazio
  - Ausência de valor
- Lógico
  - `true` ou `false`
- Caracter
  - Símbolo
- Inteiro
  - Arquitetura alvo
- Real
  - IEEE-754
- Tipo ponteiro nulo
  - Sobrecarga

```
void

bool

char    char16_t    char32_t

short   int   long    long long

float    double

nullptr_t
```

Tipos fundamentais

- O tipo vazio
  - Ausência de valor
- Lógico
  - `true` ou `false`
- Caracter
  - Símbolo
- Inteiro
  - Arquitetura alvo
- Real
  - IEEE-754
- Tipo ponteiro nulo
  - Sobrecarga

```
void

bool

char    char16_t    char32_t

short   int   long    long long

float    double

nullptr_t
```

Tipos fundamentais

- O tipo vazio
  - Ausência de valor
- Lógico
  - `true` ou `false`
- Caracter
  - Símbolo
- Inteiro
  - Arquitetura alvo
- Real
  - IEEE-754
- Tipo ponteiro nulo
  - Sobrecarga

```
void

bool

char    char16_t    char32_t

short   int   long    long long

float    double

nullptr_t
```

Tipos fundamentais

- O tipo vazio
  - Ausência de valor
- Lógico
  - `true` ou `false`
- Caracter
  - Símbolo
- Inteiro
  - Arquitetura alvo
- Real
  - IEEE-754
- Tipo ponteiro nulo
  - Sobrecarga

```
void

bool

char    char16_t    char32_t

short   int   long    long long

float    double

nullptr_t
```

Tipos fundamentais

- O tipo vazio
  - Ausência de valor
- Lógico
  - `true` ou `false`
- Caracter
  - Símbolo
- Inteiro
  - Arquitetura alvo
- Real
  - IEEE-754
- Tipo ponteiro nulo
  - Sobrecarga

```
void

bool

char    char16_t    char32_t

short   int   long    long long

float    double

nullptr_t
```

Tipos fundamentais

- O tipo vazio
  - Ausência de valor
- Lógico
  - `true` ou `false`
- Caracter
  - Símbolo
- Inteiro
  - Arquitetura alvo
- Real
  - IEEE-754
- Tipo ponteiro nulo
  - Sobrecarga

```
void

bool

char    char16_t    char32_t

short   int   long    long long

float    double

nullptr_t
```

Tipos fundamentais

- O tipo vazio
  - Ausência de valor
- Lógico
  - `true` ou `false`
- Caracter
  - Símbolo
- Inteiro
  - Arquitetura alvo
- Real
  - IEEE-754
- Tipo ponteiro nulo
  - Sobrecarga

```
void

bool

char    char16_t    char32_t

short   int   long    long long

float    double

nullptr_t
```

Tipos fundamentais

- O tipo vazio
  - Ausência de valor
- Lógico
  - `true` ou `false`
- Caracter
  - Símbolo
- Inteiro
  - Arquitetura alvo
- Real
  - IEEE-754
- Tipo ponteiro nulo
  - Sobrecarga

```
void

bool

char    char16_t    char32_t

short   int   long    long long

float    double

nullptr_t
```

Tipos fundamentais

- O tipo vazio
  - Ausência de valor

```
void
```

- Lógico
  - `true` ou `false`

```
bool
```

- Caracter
  - Símbolo

```
char    char16_t    char32_t
```

- Inteiro
  - Arquitetura alvo

```
short   int   long     long long
```

- Real
  - IEEE-754

```
float     double
```

- Tipo ponteiro nulo
  - Sobrecarga

```
nullptr_t
```

Tipos fundamentais

- O tipo vazio
  - Ausência de valor
- Lógico
  - `true` ou `false`
- Caracter
  - Símbolo
- Inteiro
  - Arquitetura alvo
- Real
  - IEEE-754
- Tipo ponteiro nulo
  - Sobrecarga

`void`

`bool`

`char` `char16_t` `char32_t`

`short` `int` `long` `long long`

`float` `double`

`nullptr_t`

Pode ser signed ou unsigned

Tipos fundamentais

- O tipo vazio
  - Ausência de valor
- Lógico
  - `true` ou `false`
- Caracter
  - Símbolo
- Inteiro
  - Arquitetura alvo
- Real
  - IEEE-754
- Tipo ponteiro nulo
  - Sobrecarga

```
void

bool

char    char16_t    char32_t

short   int   long    long long

float    double

nullptr_t
```

Tipos fundamentais

```cpp
#include <iostream>


int main(){
    int a,b;
    unsigned short s1;
    long long l;
    float f;
    double d;
    /* ... */
    return 0;
}
```

Tipos fundamentais

```
#include <iostream>



int main(){
   int a,b;
   unsigned short s1;
   long long l;
   float f;
   double d;
   /* ... */
   return 0;
}
```

32 bits com sinal

Tipos fundamentais

```cpp
#include <iostream>


int main(){
    int a,b;
    unsigned short s1;
    long long l;
    float f;
    double d;
    /* ... */
    return 0;
}
```

Tipos fundamentais

```
#include <iostream>


int main(){
    int a,b;
    unsigned short s1;
    long long l;
    float f;
    double d;
    /* ... */
    return 0;
}
```

16 bits sem sinal

Tipos fundamentais

```cpp
#include <iostream>


int main(){
    int a,b;
    unsigned short s1;
    long long l;
    float f;
    double d;
    /* ... */
    return 0;
}
```

Tipos fundamentais

```
#include <iostream>


int main(){
   int a,b;
   unsigned short s1;
   long long l;
   float f;
   double d;
   /* ... */
   return 0;
}
```

64 bits com sinal

Tipos fundamentais

```cpp
#include <iostream>


int main(){
    int a,b;
    unsigned short s1;
    long long l;
    float f;
    double d;
    /* ... */
    return 0;
}
```

Tipos fundamentais

```
#include <iostream>


int main(){
   int a,b;
   unsigned short s1;
   long long l;
   float f;
   double d;
   /* ... */
   return 0;
}
```

32 bits IEEE-754

Tipos fundamentais

```cpp
#include <iostream>


int main(){
    int a,b;
    unsigned short s1;
    long long l;
    float f;
    double d;
    /* ... */
    return 0;
}
```

Tipos fundamentais

```
#include <iostream>


int main(){
   int a,b;
   unsigned short s1;
   long long l;
   float f;
   double d;
   /* ... */
   return 0;
}
```

64 bits IEEE-754

Tipos fundamentais

```cpp
#include <iostream>


int main(){
    int a,b;
    unsigned short s1;
    long long l;
    float f;
    double d;
    /* ... */
    return 0;
}
```

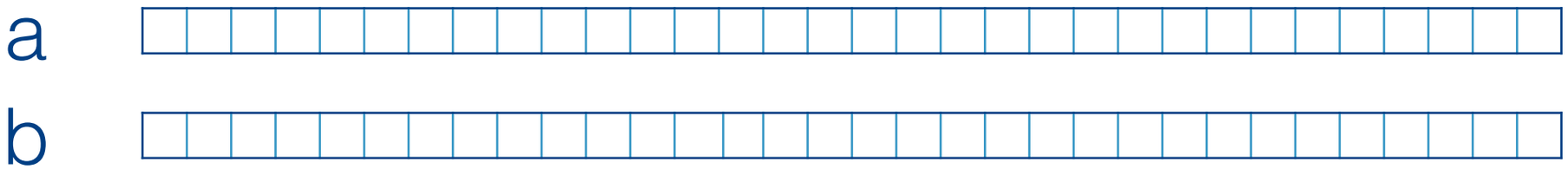
Conversão implícita

```cpp
int main(){
  int a,b;
  short c;
  std::cin >> a >> b;
  c = a+b;
  std::cout << a << " + " << b << " = " << c << std::endl;
  return 0;
}
```

Conversão implícita

```cpp
int main(){
  int a,b;
  short c;
  std::cin >> a >> b;
  c = a+b;
  std::cout << a << " + " << b << " = " << c << std::endl;
  return 0;
}
```

Conversão implícita




```cpp
int main(){
  int a,b;
  short c;
  std::cin >> a >> b;
  c = a+b;
  std::cout << a << " + " << b << " = " << c << std::endl;
  return 0;
}
```

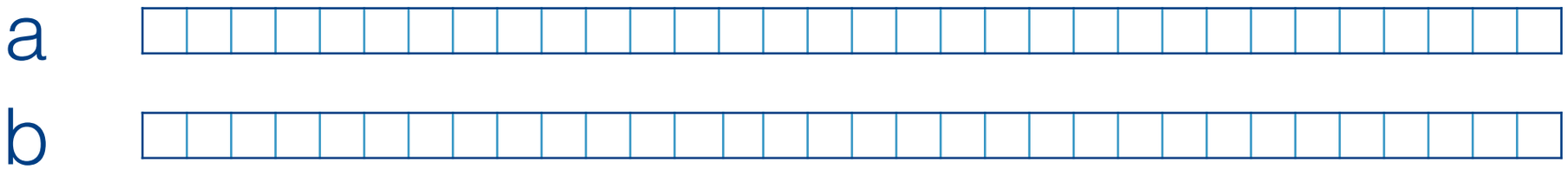
Conversão implícita

a 

b 

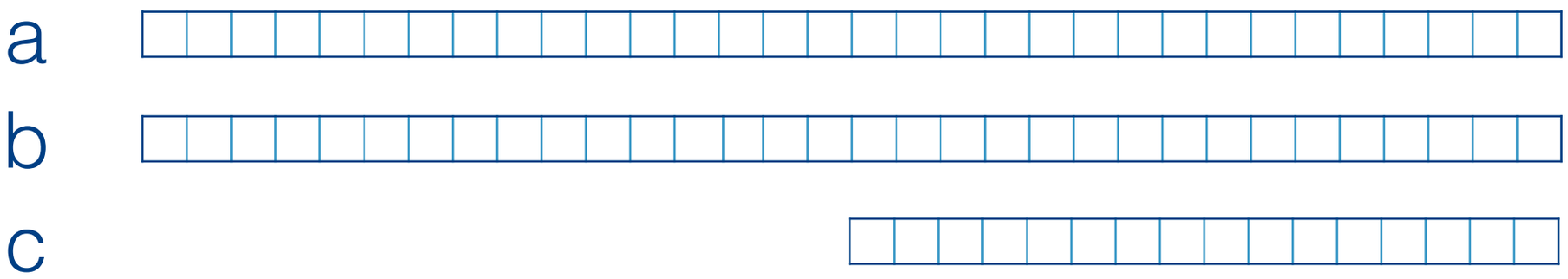
```cpp
int main(){
  int a,b;
  short c;
  std::cin >> a >> b;
  c = a+b;
  std::cout << a << " + " << b << " = " << c << std::endl;
  return 0;
}
```

Conversão implícita



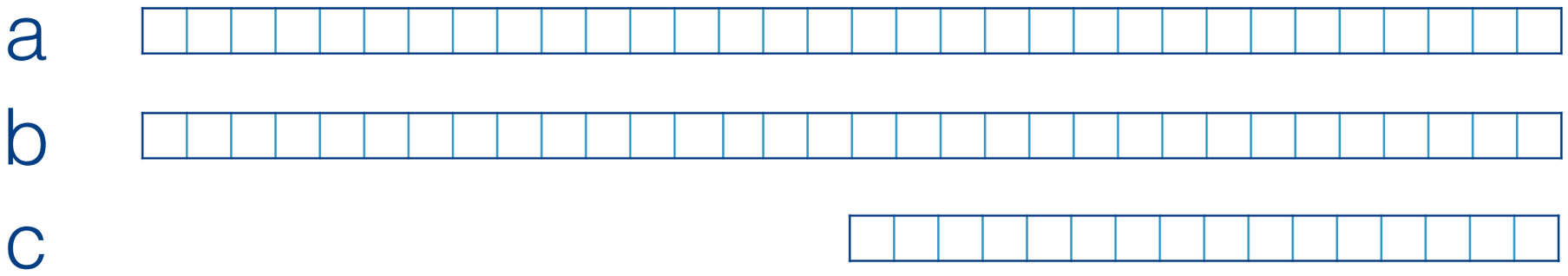

```cpp
int main(){
  int a,b;
  short c;
  std::cin >> a >> b;
  c = a+b;
  std::cout << a << " + " << b << " = " << c << std::endl;
  return 0;
}
```

Conversão implícita



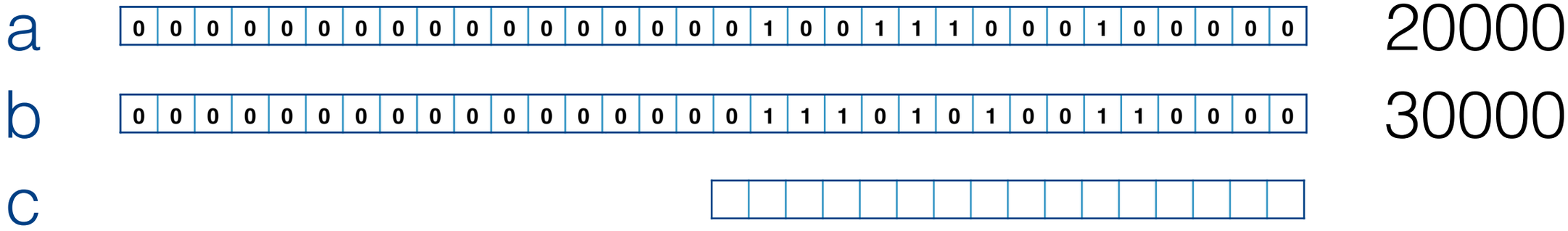

```cpp
int main(){
  int a,b;
  short c;
  std::cin >> a >> b;
  c = a+b;
  std::cout << a << " + " << b << " = " << c << std::endl;
  return 0;
}
```

Conversão implícita



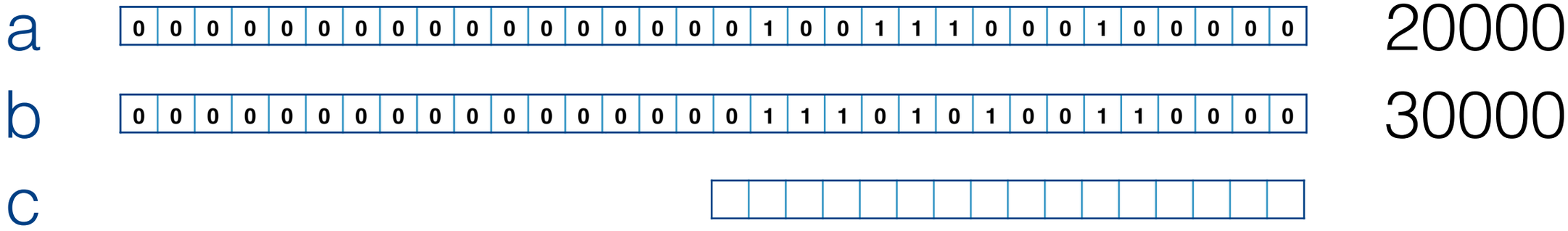

```cpp
int main(){
  int a,b;
  short c;
  std::cin >> a >> b;
  c = a+b;
  std::cout << a << " + " << b << " = " << c << std::endl;
  return 0;
}
```

Conversão implícita



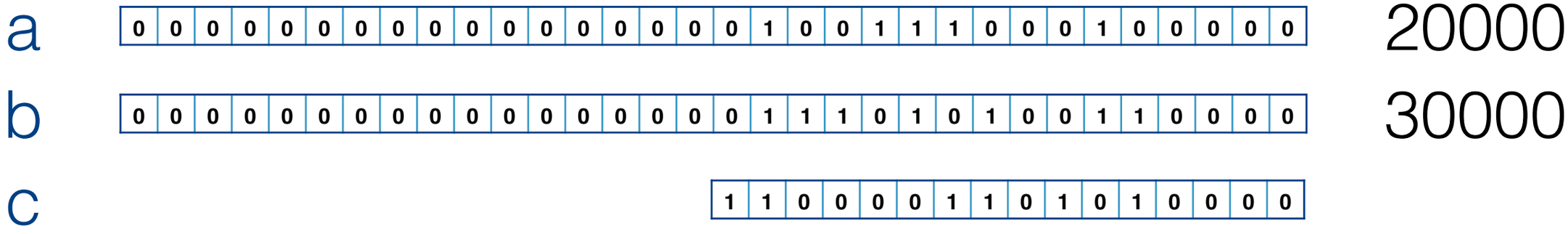

```cpp
int main(){
  int a,b;
  short c;
  std::cin >> a >> b;
  c = a+b;
  std::cout << a << " + " << b << " = " << c << std::endl;
  return 0;
}
```

Conversão implícita



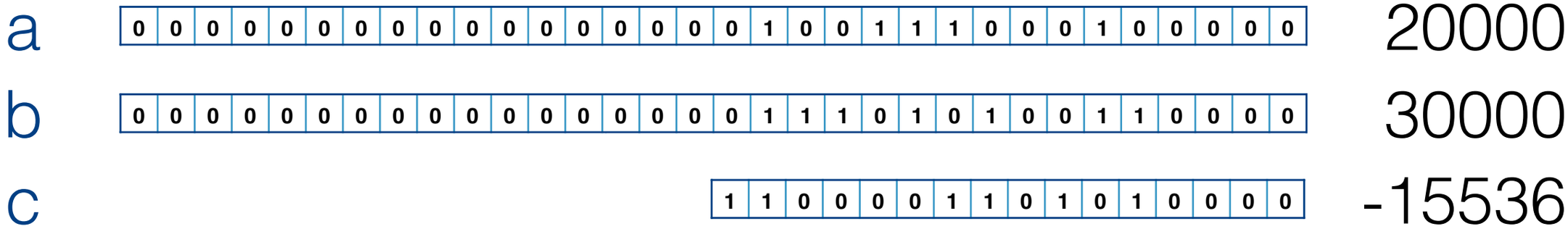

```cpp
int main(){
  int a,b;
  short c;
  std::cin >> a >> b;
  c = a+b;
  std::cout << a << " + " << b << " = " << c << std::endl;
  return 0;
}
```

Conversão implícita




```cpp
int main(){
  int a,b;
  short c;
  std::cin >> a >> b;
  c = a+b;
  std::cout << a << " + " << b << " = " << c << std::endl;
  return 0;
}
```

Conversão implícita




```cpp
int main(){
  int a,b;
  short c;
  std::cin >> a >> b;
  c = a+b;
  std::cout << a << " + " << b << " = " << c << std::endl;
  return 0;
}
```

Conversão implícita

```
int main(){
  int x, y;
  double z;
  x = 10;
  y = 3;
  z = x / y;
  std::cout << x << " / " << y << " = " << z << std::endl;
}
```

Conversão implícita

```
int main(){
  int x, y;
  double z;
  x = 10;
  y = 3;
  z = x / y;
  std::cout << x << " / " << y << " = " << z << std::endl;
}
```

z=3

Conversão implícita

```
int main(){
  int x, y;
  double z;
  x = 10;
  y = 3;
  z = x / y;
  std::cout << x << " / " << y << " = " << z << std::endl;
}
```

z=3

```
int main(){
  int x;
  double y, z;
  x = 10;
  y = 3;
  z = x / y;
  std::cout << x << " / " << y << " = " << z << std::endl;
}
```

Conversão implícita

```
int main(){
  int x, y;
  double z;
  x = 10;
  y = 3;
  z = x / y;
  std::cout << x << " / " << y << " = " << z << std::endl;
}
```

z=3

```
int main(){
  int x;
  double y, z;
  x = 10;
  y = 3;
  z = x / y;
  std::cout << x << " / " << y << " = " << z << std::endl;
}
```

z=3.3333334

Conversão explícita

```cpp
#include <iostream>

int main(){
  int a,b;
  double c;
  cin >> a >> b;
  c=a/b;
  std::cout << a << " / " << b << " = " << c << std::endl;
  return 0;
}
```

Conversão explícita

```
#include <iostream>


int main(){
   int a,b;
   double c;
   cin >> a >> b;
   c=a/b;
   std::cout << a << " / " << b << " = " << c << std::endl;
   return 0;
}
```

Conversão explícita

```
#include <iostream>

int main(){
   int a,b;
   double c;
   cin >> a >> b;
   c=a/b;
   std::cout << a << " / " << b << " = " << c << std::endl;
   return 0;
}
```

int/int -> int

Conversão explícita

```cpp
#include <iostream>

int main(){
  int a,b;
  double c;
  cin >> a >> b;
  c=a/b;
  std::cout << a << " / " << b << " = " << c << std::endl;
  return 0;
}
```

Conversão explícita

```
#include <iostream>


int main(){
  int a,b;
  double c;
  cin >> a >> b;
  c=(double)a/(double)b;
  std::cout << a << " / " << b << " = " << c << std::endl;
  return 0;
}
```

Operadores aritméticos

- Precedência:
  - Esquerda para a direita
  - Multiplicação, divisão e resto da divisão
  - Adição e subtração

Nome	Operador	Exemplo
Adição	+	a+b
Subtração	–	a–b
Multiplicação	*	a*b
Divisão	/	a/b
Resto da divisão	%	a%b

Prática

Programação

C++

Aula 01

Jorgiano Vidal