UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

Programação e Desenvolvimento de Software 2

Introdução à linguagem C++

Prof. Luiz Chaimowicz
(slides adaptados do Prof. Douglas Macharet)

DCC
DEPARTAMENTO DE
CIÊNCIA DA COMPUTAÇÃO

Introdução

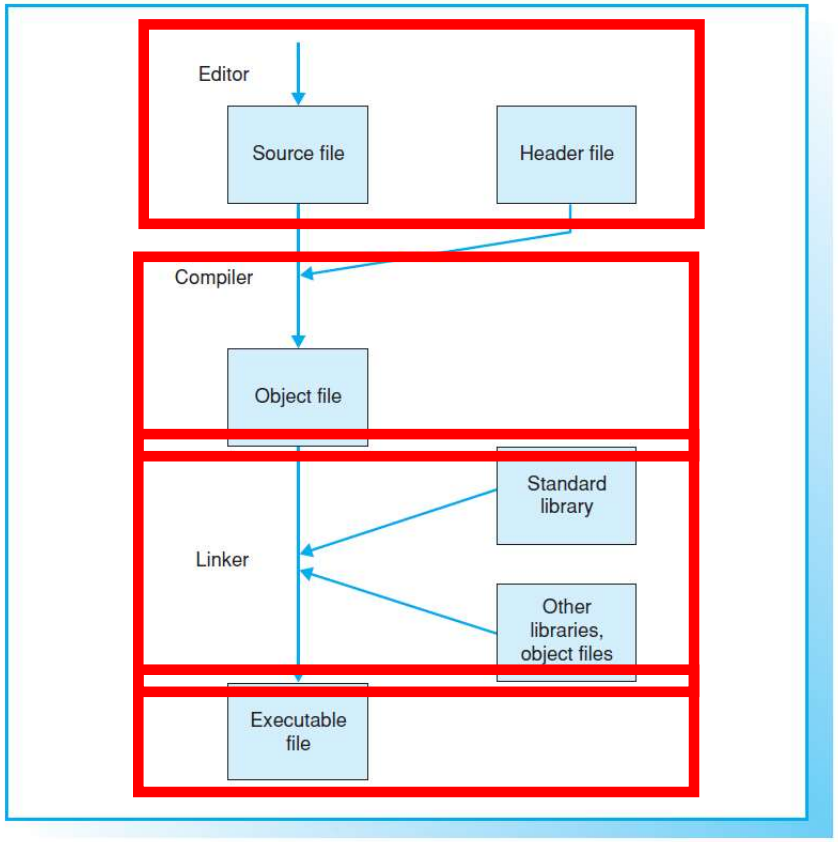
- Introdução breve, incompleta e não-definitiva!
  - Anos para dominar a linguagem, não é nosso objetivo
- Assume-se alguma experiência prévia com C
  - Foco mais na sintaxe e não em lógica de programação
- Exemplos com os recursos e princípios básicos
  - Alguns conceitos mais detalhados ao longo da disciplina

Introdução
C++

- Extensão da linguagem C
  - Características de alto e baixo nível
  - Não pode ser considerado um super conjunto
    - Não implementa o C completamente!*
- Unidades básicas: Funções → Classes
  - *C with Classes* → C++
- Multiparadigma: Procedural, OO, Funcional
- Sintaxe é fonte de inspiração para Java, C#, ...

*https://en.wikipedia.org/wiki/Compatibility_of_C_and_C++

C++



Fonte: *A Complete Guide to Programming in C++*

C vs. C++
Exemplo básico

Não precisamos da extensão para bibliotecas externas.

C

```c
#include <stdio.h>

int main() {
  printf("Hello World\n");

  return 0;
}
```

```
$ gcc hello.c -o hello
$ ./hello
"Hello world!"
```

Acesso a um escopo (namespace) específico

Stream da saída padrão

C++

```cpp
#include <iostream>

using namespace std;

int main() {
  cout << "Hello world!" << endl;

  return 0;
}
```

Quebra de linha

```
$ g++ hello.cpp -o hello
$ ./hello
"Hello world!"
```

C++
Versões

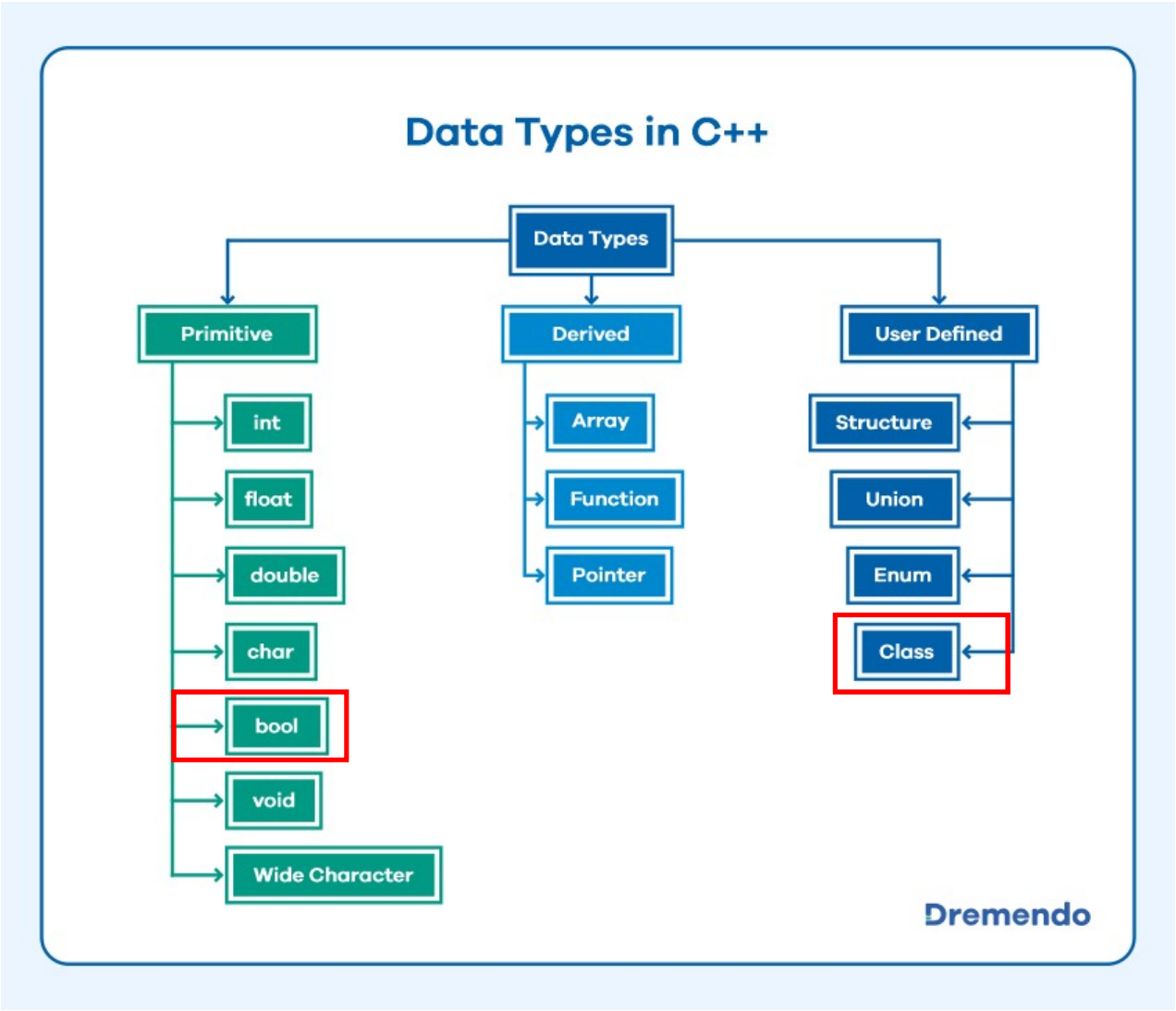
- Nesta disciplina utilizaremos **C++11**

C++98	C++11	C++14	C++17	C++20
○ Templates ○ Exceptions ○ `iostream`-API ○ `std`-library string, containers, algorithms	○ Rvalue references with move semantics ○ Lambdas ○ Variadic templates ○ Uniform initialization ○ Type inference (`auto`) ○ Range-based `for` loop ○ `constexpr` ○ `std`-library APIs support move semantics, smart pointers, concurrency, hash-based containers, `atomic<>`	○ Binary literals ○ Generalized return type deduction ○ Generalized lambda captures ○ Generic lambdas ○ Relaxed `constexpr` restrictions ○ Heterogeneous lookup in associative containers ○ `std`-library `make_unique()`, transformation `_t` alias "shortcuts"	○ Structured bindings ○ `if` and `switch` with initialization ○ Compile-time static `if constexpr` ○ Aggregate extensions ○ Fold expressions ○ Mandatory copy elision ○ Class template argument deduction ○ `std`-library `optional<>`, `variant<>`, `any<>`, `byte`, `string_view` ○ File system library ○ Parallel STL algorithms	○ …

```
$ g++ -std=c++11 -Wall hello.cpp -o hello
```

https://nohau.eu/courses/c-advanced-whatss-new-in-c-a-ka-modern-c/

C++
Tipos



https://www.dremendo.com/cpp-programming-tutorial/cpp-data-types-and-modifiers

C++

Cadeias de caracteres

Entrada e Saída

Manipulação da memória

C++

Cadeias de caracteres

- String: tipo de dado muito útil (biblioteca padrão)
- Usado para guardar uma sequência de caracteres

Include da biblioteca

Declaração e inicialização das variáveis

Para acessar um escopo/módulo (namespace) específico

Método auxiliar

```
#include <iostream>
#include <string>

int main() {
  std::string curto = "Hello World!";
  std::string longo = "Essa é uma string grande para o exemplo!";

  std::cout << curto << std::endl;
  std::cout << longo << std::endl;

  int tamanho = longo.length();
  std::cout << tamanho << std::endl;

  return 0;
}
```

https://www.cplusplus.com/reference/string/string/

C++

Cadeias de caracteres

Ao fazer isso não precisamos informar o escopo antes das variáveis

```cpp
#include <iostream>
#include <string>

using namespace std;

int main() {
  string a;
  a = "123456";
  string b;
  b = "123456";

  bool igual = (a == b);
  cout << igual << endl;

  cout << a[0] << endl;
  cout << b[5] << endl;

  a[0] = '0';
  cout << a << endl;

  string c = a + b;
  cout << c << endl;

  return 0;
}
```

Podemos guardar numerais, mas isso é uma 'palavra' e não um 'número'

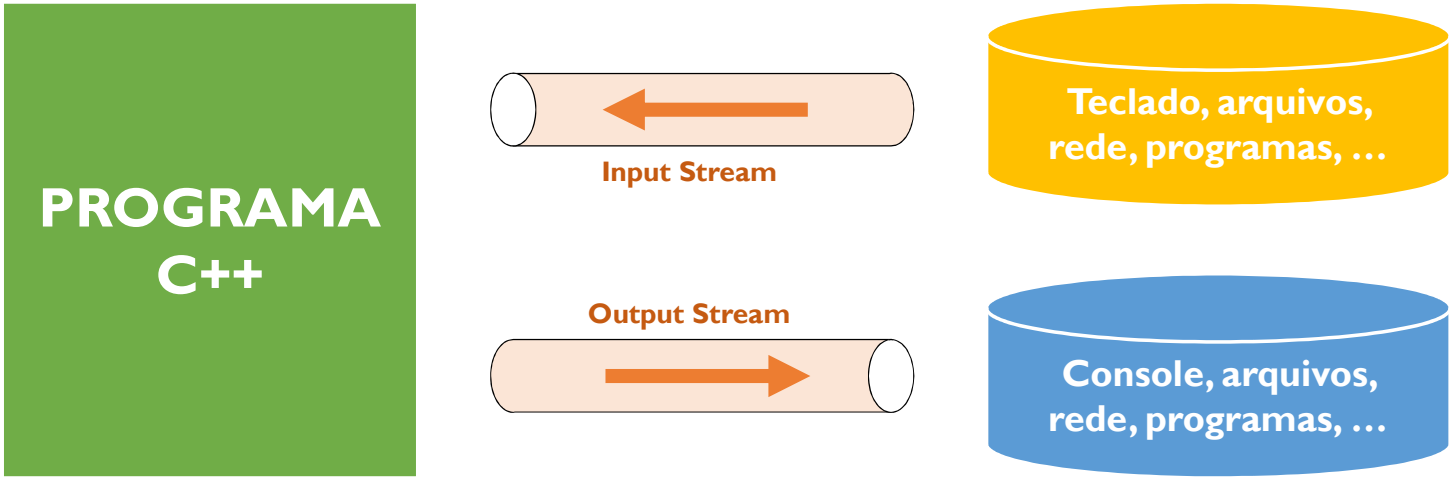
Podemos compará-las e salvar o retorno em uma variável Booleana

Acesso/manipulação de cada caractere similar a um array

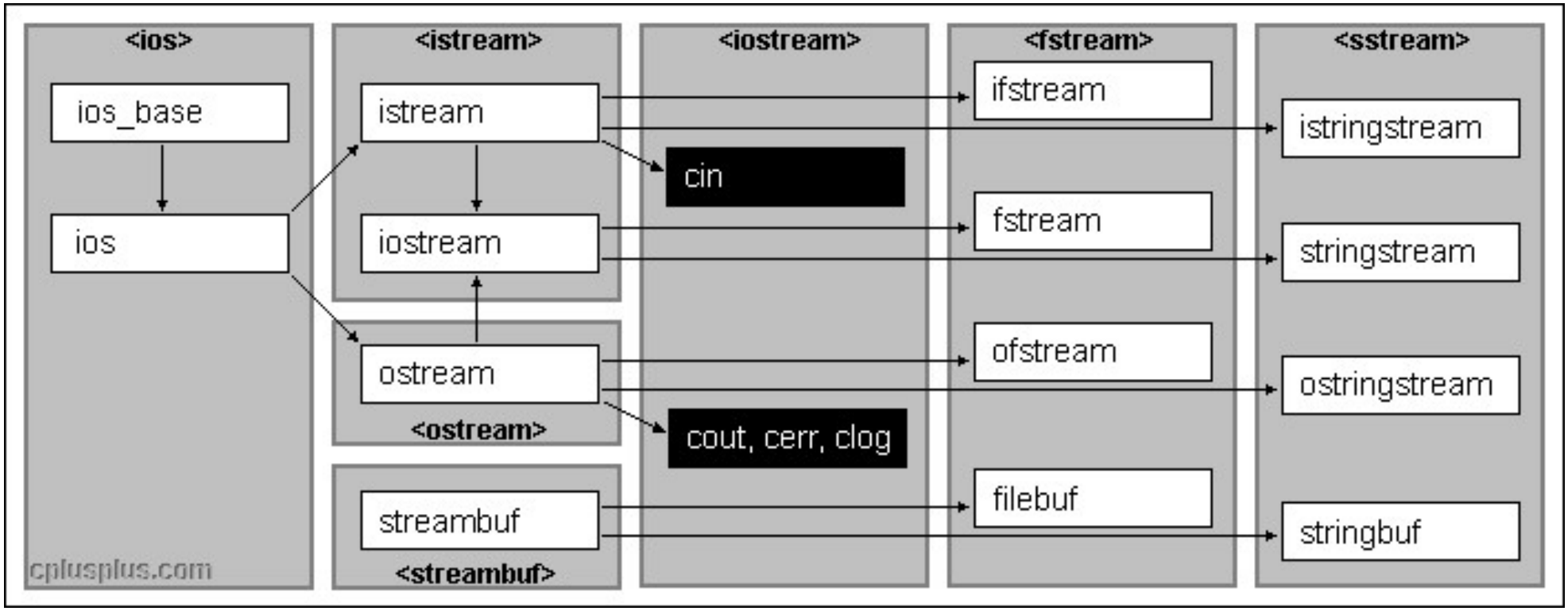
Concatenação

C++
Entrada e Saída

- Streams ("fluxos") são utilizados para comunicação
- Podemos usar *printf* também (mas vamos evitar!)



C++
Entrada e Saída



https://www.cplusplus.com/reference/iolibrary/

C++

Entrada e Saída (console)

```
#include <iostream>
#include <string>
using namespace std;

int main() {
   string nome;
   int idade;

   cout << "Digite o seu nome: ";
   cin >> nome;

   cout << "Digite sua idade: ";
   cin >> idade;

   cout << endl;
   cout << nome << " tem " << idade << " anos." << endl;

   return 0;
}
```

Include da biblioteca

Operador de **inserção** no stream

Saída padrão

Entrada padrão

Operador de **extração** do stream

Independente do tipo de entrada

```
scanf("%d", &idade);
```

Saída "encadeada"

Cuidado: leitura é feita até encontrar <espaço>, <tab> ou <final de linha>

C++

Entrada e Saída (strings)

Leitura de uma linha inteira de entrada

```
#include <iostream>
#include <string>

using namespace std;

int main() {

  string nome_completo;

  cout <<"Digite o seu nome completo: ";
  getline (cin, nome_completo);

  cout << "Ola " << nome_completo << endl;

  return 0;
}
```

http://www.cplusplus.com/reference/string/string/getline/

C++
Entrada e Saída (console)

Continue lendo os valores de **cin** em **x**, enquanto um valor **válido** puder ser lido

Assim que for lido um valor que não seja um int, ou quando cin for fechado, o loop termina. Ou seja, o loop só será executado enquanto x for válido.

```
#include <iostream>

using namespace std;

int main() {

  int x;

  while(cin >> x) {
    cout << "[" << x << "]" << endl;
  }

  return 0;
}
```

C++

Entrada e Saída (strings)

Biblioteca usada para manipular strings como se fossem streams

Criando um stream (e/s) local a partir da linha lida

Fazendo leituras no stream considerando um delimitador

```
#include <iostream>
#include <string>
#include <sstream>

using namespace std;

int main() {
  string line;
  while (getline(cin, line)) {

    stringstream info(line);

    string aux;
    while (getline(info, aux, ';')) {
        cout << aux << endl;
    }
  }
  return 0;
}
```

http://www.cplusplus.com/reference/sstream/stringstream/

C++
Entrada e Saída (strings)

Utilizando o operador de extração como se fosse o cin

```
#include <iostream>
#include <string>
#include <sstream>

using namespace std;

int main() {

  string frase = "Lorem ipsum dolor sit amet";
  string palavra;

  stringstream s(frase);

  while (s >> palavra)
    cout << palavra << endl;

  return 0;
}
```

C++

Entrada e Saída (arquivos)

Biblioteca para E/S em arquivos

Stream (arquivo) de entrada

Stream (arquivo) de saída

Escrevendo em dois streams diferesnte

```
#include <iostream>
#include <fstream>
#include <string>

using namespace std;

int main() {

  ifstream in("entrada.txt", fstream::in);

  ofstream out("saida.txt", fstream::out);

  string line;
  while (getline(in, line)) {
    cout << "*" << line << "*" << endl;
    out << "[" << line << "]" << endl;
  }

  return 0;
}
```

Nome

Modo de operação
(poderia ser ocultado nesse caso)

Console

Arquivo

http://www.cplusplus.com/reference/fstream/
http://www.cplusplus.com/reference/fstream/ifstream/ifstream/

C++

Entrada e Saída (arquivos)

Boas práticas de programação!

Tb uma boa prática, mas não é estritamente necessário

```
#include <fstream>
#include <iostream>
#include <string>
using namespace std;
 int main() {
  ifstream in("entrada.txt", fstream::in);
  if (!in.is_open())
    return 1;

  ofstream out("saida.txt", fstream::out);
  if (!out.is_open())
    return 1;

  string line;
  while (getline(in, line)) {
    cout << "*" << line << "*" << endl;
    out << "[" << line << "]" << endl;
  }
  in.close();
  out.close();
  return 0;
}
```

C++
Manipulação da memória

C

- Alocação: **malloc**
- Liberação: **free**

C++

- Alocação: **new**
- Liberação: **delete**

C++

Manipulação da memória

C

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

int main() {
  int *p = (int*) malloc(sizeof(int)*5);
  int i;

  for(i=0; i<5; i++)
    p[i] = i;

  for(i=0; i<5; i++)
    printf("%i\n", p[i]);

  free(p);
  return 0;
}
```

C++

```cpp
#include <iostream>

using namespace std;

int main() {
  int *p = new int[5];

  for(int i=0; i<5; i++)
    p[i] = i;

  for(int i=0; i<5; i++)
    cout << p[i] << endl;

  delete p;
  return 0;
}
```

Considerações finais

- Os exemplos vistos são apenas motivadores iniciais
- Existem diversas outras diferenças
  - Declaração de tipos, referências, namespaces, exceções, ...
- Não é um curso apenas da linguagem!
  - Não devemos focar em detalhes muito específicos
  - C++ deve ser vista mais como uma ferramenta
- Praticar, praticar, praticar!