Шпаргалка по переводу конструкций Python на C++

для быстрого входа в курс программирования на C++

Как пользоваться. Левая колонка показывает привычную конструкцию Python, правая - близкий вариант на C++. Это не автоматический перевод программы: C++ требует явно думать о типах, времени жизни объектов, владении памятью, заголовочных файлах и стоимости операций. Примеры ориентированы на современный C++23; места C++20, C++23 и C++26 отдельно помечены.

Оглавление

1. Простейшая программа

Осенний семестр: структура файла, точка входа, вывод

Python	C++

`print("Hello, World!")`	`#include <print>`
	
	`int main() {`
	`  std::println("Hello, World!");`
	`}`

`# В Python код можно писать сразу в файле.`
`# Функция main не обязательна.`
	`#include <iostream>`
	
	`int main() {`
	`  std::cout << "Hello, World!\n";`
	`  return 0;`
	`}`

`import math`
`print(math.sqrt(9))`
	`#include <cmath>`
	`#include <print>`
	
	`int main() {`
	`  std::println("{}", std::sqrt(9.0));`
	`}`

`import sys`
`print(sys.argv)`
	`#include <print>`
	
	`int main(int argc, char* argv[]) {`
	`  for (int i = 0; i < argc; ++i) {`
	`  std::println("{}", argv[i]);`
	`  }`
	`}`

2. Комментарии и имена

Осенний семестр: комментарии, стиль именования, области видимости

Python	C++
`# Однострочный комментарий`	`// Однострочный комментарий`

`"""`
`Часто используется как docstring,`
`но это строковый литерал, а не комментарий.`
`"""`
	`/*`
	` Многострочный комментарий.`
	` Компилятор полностью игнорирует этот текст.`
	`*/`

`def distance(x, y):`
`  """Вернуть расстояние."""`
`  return (x*x + y*y) ** 0.5`
	`/// Вернуть расстояние.`
	`double distance(double x, double y) {`
	`  return std::sqrt(x * x + y * y);`
	`}`

`snake_case = 10`
`CONSTANT_NAME = 42`
	`int snake_case = 10;`
	`constexpr int constant_name = 42;`
	`// В C++ константность лучше выражать типом,`
	`// а не только соглашением об имени.`

`x = 1`
`if x > 0:`
`  y = 2`
`print(y)`
	`int x = 1;`
	`if (x > 0) {`
	`  int y = 2;`
	`}`
	`// y вне блока не существует.`

3. Типы данных и объявления

Осенний семестр: статическая типизация, auto, const, литералы

Python	C++

`x = 10`
`y = 3.14`
`name = "Ada"`
`ok = True`
	`int x = 10;`
	`double y = 3.14;`
	`std::string name = "Ada";`
	`bool ok = true;`

`x = 10 # тип хранится у объекта`
`x = "ten" # можно заменить тип`
	`int x = 10; // тип переменной фиксирован`
	`// x = "ten"; // ошибка компиляции`

`x = 10`
`y = 3.14`
	`auto x = 10; // int`
	`auto y = 3.14; // double`
	`// auto выводит тип на этапе компиляции.`

`PI = 3.14159 # только соглашение`	`constexpr double pi = 3.14159;` `const int max_count = 100;`

`big = 1_000_000`
`b = 0b101010`
`h = 0xff`
	`int big = 1'000'000;`
	`int b = 0b101010;`
	`int h = 0xff;`

`x = None`	`int* p = nullptr; // нет адреса`
	`std::optional<int> x = std::nullopt; // нет значения`
	`// Для функций без результата: void.`

`a, b = 1, 2`	`int a = 1;`
	`int b = 2;`
	
	`// или`
	`auto [x, y] = std::pair{1, 2};`

4. Арифметика и сравнения

Осенний семестр: операции, деление, остаток, преобразования

Python	C++
x + y x - y x * y x / y	x + y x - y x * y x / y // Если x и y целые, / в C++ дает целую часть.
7 / 2 # 3.5 7 // 2 # 3 -7 // 2 # -4	7.0 / 2 // 3.5 7 / 2 // 3 -7 / 2 // -3 // В C++ целое деление усекается к нулю.

x % y	x % y
	// Для отрицательных чисел знак результата
	// в Python и C++ может отличаться.

x ** 2 x ** 0.5	x * x std::pow(x, 0.5) std::sqrt(x)
x == y x != y x < y a <= x <= b	x == y x != y x < y a <= x && x <= b

abs(a - b) < 1e-9	std::abs(a - b) < 1e-9
	// Для double не сравниваем "почти равные"
	// числа через ==.

int(3.9) # 3
float("2.5") # 2.5
str(42) # "42"
	static_cast<int>(3.9) // 3
	std::stod("2.5") // double
	std::to_string(42) // string

5. Логические и побитовые операции

Осенний семестр: bool, short circuit, битовые маски

Python	C++
x and y x or y not x	x && y x || y !x

# У and/or есть ленивое вычисление. 	
if i < len(a) and a[i] > 0:
    ...
	if (i < a.size() && a[i] > 0) {
	    // a[i] не вычисляется, если i вне размера.
	}

x & y x | y x ^ y ~x x << n x >> n	x & y x | y x ^ y ~x x << n x >> n // Лучше использовать unsigned-типы.

mask = 1 << k
if value & mask:
    ...
	unsigned mask = 1u << k;
	if (value & mask) {
	    // k-й бит установлен.
	}

value |= 1 << k # установить бит
value &= ~(1 << k) # сбросить бит
value ^= 1 << k # переключить бит
	value |= 1u << k;
	value &= ~(1u << k);
	value ^= 1u << k;

6. Присваивание и обмен значений

Осенний семестр: изменение состояния переменных

Python	C++

a = 10
a += 2
a -= 2
a *= 2
a /= 2
a //= 2
a %= 2
	int a = 10;
	a += 2;
	a -= 2;
	a *= 2;
	a /= 2;
	// Для целых a /= 2 тоже целочисленно.
	a %= 2;

a, b = b, a	std::swap(a, b);

i += 1
i -= 1
	++i; // увеличить на 1
	--i; // уменьшить на 1
	i += 1; // тоже можно

x = y = 0	int x = 0;
	int y = 0;
	// Можно: x = y = 0; если обе переменные уже есть.

7. Ввод с консоли

Осенний семестр: input, split, getline, stringstream

Python	C++
n = int(input()) x = float(input())	int n; double x; std::cin >> n; std::cin >> x;
name = input()	std::string name; std::getline(std::cin, name);

Python	C++
`a, b = map(int, input().split())`	`int a, b;` `std::cin >> a >> b;`

`nums = list(map(int, input().split()))`	`std::string line;`
	`std::getline(std::cin, line);`
	
	`std::istringstream in(line);`
	`std::vector<int> nums;`
	`for (int x; in >> x; ) {`
	`  nums.push_back(x);`
	`}`

`n = int(input())`
`a = [int(input()) for _ in range(n)]`
	`int n;`
	`std::cin >> n;`
	`std::vector<int> a(n);`
	`for (int& x : a) {`
	`  std::cin >> x;`
	`}`

8. Вывод и форматирование

Осенний семестр: print, cout, std::format, std::print

Python	C++
`print(x)` `print(a, b, c)`	`std::cout << x << '\n';` `std::cout << a << ' ' << b << ' ' << c << '\n';`
`print(a, b, c, sep="")` `print(x, end=" ")`	`std::cout << a << b << c << '\n';` `std::cout << x << ' ';`

`name = "Ada"`
`print(f"Hello, {name}!")`
	`std::string name = "Ada";`
	`std::println("Hello, {}!", name); // C++23`
	
	`// или`
	`std::cout << std::format("Hello, {}!\n", name);`

`print(f"{x:.2f}")` `print(f"{n:05d}")`	`std::println("{:.2f}", x);` `std::println("{:05}", n);`

`print(*a)`
`print(*a, sep=", ")`
	`for (std::size_t i = 0; i < a.size(); ++i) {`
	`  if (i != 0) std::cout << ' ';`
	`  std::cout << a[i];`
	`}`
	`std::cout << '\n';`

9. Условный оператор

Осенний семестр: if, elif, else, блоки со скобками

Python	C++

`if x < 0:`
`  print("negative")`
	`if (x < 0) {`
	`  std::println("negative");`
	`}`

`if x < 0:`
`  print("negative")`
`elif x > 0:`
`  print("positive")`
`else:`
`  print("zero")`
	`if (x < 0) {`
	`  std::println("negative");`
	`} else if (x > 0) {`
	`  std::println("positive");`
	`} else {`
	`  std::println("zero");`
	`}`

`result = "even" if n % 2 == 0 else "odd"`	`std::string result = (n % 2 == 0) ? "even" : "odd";`

`if 1 <= day <= 5:`
`  print("workday")`
	`if (1 <= day && day <= 5) {`
	`  std::println("workday");`
	`}`

`if x:`
`  print("truthy")`
	`if (x != 0) {`
	`  std::println("non-zero");`
	`}`
	`// В C++ лучше писать условие явно.`

10. Выбор варианта

Осенний семестр: match/case, switch, цепочка if

Python	C++

`match command:`
`  case "start":`
`  run()`
`  case "stop":`
`  stop()`
`  case _:`
`  print("unknown")`
	`if (command == "start") {`
	`  run();`
	`} else if (command == "stop") {`
	`  stop();`
	`} else {`
	`  std::println("unknown");`
	`}`

Python	C++

match day:
  case 1:
  print("Mon")
  case 2:
  print("Tue")
  case _:
  print("?")
	switch (day) {
	  case 1:
	  std::println("Mon");
	  break;
	  case 2:
	  std::println("Tue");
	  break;
	  default:
	  std::println("?");
	}

match value:
  case 1 | 2 | 3:
  print("small")
	switch (value) {
	  case 1:
	  case 2:
	  case 3:
	  std::println("small");
	  break;
	}

if x in {1, 3, 5}:
  print("odd digit")
	if (x == 1 || x == 3 || x == 5) {
	  std::println("odd digit");
	}

11. Циклы for и while

Осенний семестр: range, обход контейнеров, break, continue

Python	C++

for i in range(n):
  print(i)
	for (int i = 0; i < n; ++i) {
	  std::println("{}", i);
	}

for i in range(1, n + 1):
  ...
	for (int i = 1; i <= n; ++i) {
	  // ...
	}

for i in range(n - 1, -1, -1):
  ...
	for (int i = n - 1; i >= 0; --i) {
	  // осторожно: i должен быть знаковым
	}

for x in a:
  print(x)
	for (int x : a) {
	  std::println("{}", x);
	}
	
	for (int& x : a) {
	  x *= 2; // изменить элементы
	}

for i, x in enumerate(a):
  print(i, x)
	for (std::size_t i = 0; i < a.size(); ++i) {
	  std::println("{} {}", i, a[i]);
	}

for x, y in zip(a, b):
  print(x + y)
	std::size_t n = std::min(a.size(), b.size());
	for (std::size_t i = 0; i < n; ++i) {
	  std::println("{}", a[i] + b[i]);
	}
	// std::views::zip есть в C++23, но поддержка
	// компиляторами может отличаться.

while x > 0:
  x -= 1
	while (x > 0) {
	  --x;
	}

while True:
  line = input()
  if line == "stop":
  break
  if line == "":
  continue
	while (true) {
	  std::string line;
	  std::getline(std::cin, line);
	  if (line == "stop") break;
	  if (line.empty()) continue;
	}

12. Функции

Осенний семестр: параметры, возвращаемое значение, перегрузка

Python	C++

def square(x):
  return x * x
	int square(int x) {
	  return x * x;
	}
	
	double square(double x) {
	  return x * x;
	}

def greet(name="student"):
  print(f"Hello, {name}")
	void greet(std::string_view name = "student") {
	  std::println("Hello, {}", name);
	}

def divmod2(a, b):
  return a // b, a % b
 	
q, r = divmod2(7, 3)
	std::pair<int, int> divmod2(int a, int b) {
	  return {a / b, a % b};
	}
	
	auto [q, r] = divmod2(7, 3);

def add_one(a):
  a.append(1)
	void add_one(std::vector<int>& a) {
	  a.push_back(1);
	}
	// & означает передачу по ссылке.

Python	C++

def norm(x: float, y: float) -> float:
  return (x*x + y*y) ** 0.5
	double norm(double x, double y) {
	  return std::hypot(x, y);
	}

f = lambda x: x * x	auto f = [](int x) {
	  return x * x;
	};

def fact(n):
  if n <= 1:
  return 1
  return n * fact(n - 1)
	long long fact(int n) {
	  if (n <= 1) return 1;
	  return n * fact(n - 1);
	}

13. Многофайловая программа

Осенний семестр: импорт, заголовочные файлы, единицы трансляции

Python	C++

# math_utils.py 	
def square(x):
  return x * x
 	
# main.py 	
from math_utils import square
print(square(5))
	// math_utils.h
	int square(int x);
	
	// math_utils.cpp
	#include "math_utils.h"
	int square(int x) { return x * x; }
	
	// main.cpp
	#include "math_utils.h"
	int main() { std::println("{}", square(5)); }

if __name__ == "__main__":
  main()
	int main() {
	  run_program();
	}
	// В C++ точка входа всегда main.

import math	#include <cmath> // #include вставляет объявления из заголовка.

from package import name	import my_module; // C++20 modules, если настроены
	// В учебных проектах чаще используют #include.

14. Структуры, кортежи, перечисления

Осенний семестр: группировка данных и именованные варианты

Python	C++
point = (10, 20) x, y = point	std::pair<int, int> point{10, 20}; auto [x, y] = point;
item = ("Ada", 95, True) name, score, ok = item	std::tuple<std::string, int, bool> item{"Ada", 95, true}; auto [name, score, ok] = item;

from dataclasses import dataclass
 	
@dataclass
class Student:
  name: str
  score: int
	struct Student {
	  std::string name;
	  int score;
	};

s = Student("Ada", 95) print(s.name)	Student s{"Ada", 95}; std::println("{}", s.name);

from enum import Enum
 	
class Color(Enum):
  Red = 1
  Green = 2
	enum class Color {
	  Red = 1,
	  Green = 2
	};

if color == Color.Red:
  print("red")
	if (color == Color::Red) {
	  std::println("red");
	}

15. Случайные числа

Осенний семестр: random, генератор и распределение

Python	C++

import random
x = random.randint(1, 6)
	std::random_device rd;
	std::mt19937 gen(rd());
	std::uniform_int_distribution<int> dist(1, 6);
	int x = dist(gen);

random.seed(42)
x = random.randint(1, 6)
	std::mt19937 gen(42);
	std::uniform_int_distribution<int> dist(1, 6);
	int x = dist(gen);

x = random.random() # [0.0, 1.0)
y = random.uniform(a, b)
	std::uniform_real_distribution<double> d01(0.0, 1.0);
	std::uniform_real_distribution<double> dab(a, b);
	double x = d01(gen);
	double y = dab(gen);

x = random.choice(a)	std::uniform_int_distribution<std::size_t> dist(0, a.size() - 1); auto x = a[dist(gen)];
random.shuffle(a)	std::shuffle(a.begin(), a.end(), gen);

Python	C++

import random
x = random.gauss(5, 2)
	std::normal_distribution<double> norm(5.0, 2.0);
	double x = norm(gen);
	// Второй параметр - стандартное отклонение.

# простой учебный вариант генератора	std::default_random_engine gen(rd()); 	
	// Допустимо для обзора, но конкретный алгоритм
	// стандартом не фиксируется.

16. Массивы, ссылки, указатели

Осенний семестр: память, индексы, передача массивов

Python	C++
a = [1, 2, 3] print(a[0])	std::vector<int> a{1, 2, 3}; std::println("{}", a[0]);
a = [0] * 10	std::vector<int> a(10, 0); std::array<int, 10> b{}; // размер известен при компиляции

matrix = [[0] * m for _ in range(n)]	std::vector<std::vector<int>> matrix(n, std::vector<int>(m, 0));
	
	// Иногда лучше плоский массив:
	std::vector<int> flat(n * m);
	auto at = [&](int i, int j) -> int& { return flat[i * m + j]; };

len(a) a[i]	a.size() a[i] // без проверки a.at(i) // с проверкой и исключением

def change(a):
  a[0] = 10
	void change(std::vector<int>& a) {
	  a[0] = 10;
	}

# Аналога указателей в обычном Python-коде нет.	int x = 10; 	
	int* p = &x; // адрес x
	std::println("{}", *p); // значение по адресу

x = None
if x is None:
  ...
	int* p = nullptr;
	if (p == nullptr) {
	  // указатель никуда не указывает
	}

# Низкоуровневое выделение памяти скрыто.	int* p = new int{42}; 	
	delete p;
	
	// В прикладном C++ чаще:
	auto q = std::make_unique<int>(42);

# Передать "вид" на часть данных.	void print_all(std::span<const int> a) { 	
	  for (int x : a) std::println("{}", x);
	}
	// std::span не владеет памятью.
	// std::span::at() - C++26.

17. Строки

Осенний семестр: str, std::string, string_view, C-строки

Python	C++
s = "hello" len(s) s[0]	std::string s = "hello"; s.size(); s[0];
s + t s * 3	s + t std::string result; for (int i = 0; i < 3; ++i) result += s;
s[1:4] s[:3] s[3:]	s.substr(1, 3) s.substr(0, 3) s.substr(3)

s.find("ab")
s.startswith("ab")
s.endswith("ab")
	s.find("ab") // позиция или npos
	s.starts_with("ab") // C++20
	s.ends_with("ab") // C++20

s.strip()
s.lower()
s.upper()
	// В стандартной библиотеке C++ нет одной функции
	// strip/lower/upper для всей строки.
	// Используют алгоритмы, ranges или внешние библиотеки.

parts = s.split()	std::istringstream in(s);
	std::vector<std::string> parts;
	for (std::string word; in >> word;) {
	  parts.push_back(word);
	}

", ".join(words)	std::string result;
	for (std::size_t i = 0; i < words.size(); ++i) {
	  if (i) result += ", ";
	  result += words[i];
	}

# bytes / bytearray 	
b = b"abc"
	char cstr[] = "abc"; // C-строка с '\0'
	std::string s = "abc"; // строка C++
	std::string_view sv = s; // невладеющий вид

18. Векторы, итераторы, алгоритмы

Осенний семестр: list как динамический массив, STL algorithms

Python	C++

a = []
a.append(10)
a.pop()
	std::vector<int> a;
	a.push_back(10);
	a.pop_back();

a.insert(i, x) del a[i]	a.insert(a.begin() + i, x); a.erase(a.begin() + i);

a.sort()
b = sorted(a)
	std::sort(a.begin(), a.end());
	
	auto b = a;
	std::sort(b.begin(), b.end());

a.sort(reverse=True)	std::sort(a.begin(), a.end(), std::greater<>{});

min(a), max(a), sum(a)	*std::min_element(a.begin(), a.end());
	*std::max_element(a.begin(), a.end());
	std::accumulate(a.begin(), a.end(), 0);

x in a	std::ranges::find(a, x) != a.end() // или std::find(a.begin(), a.end(), x) != a.end()
any(x > 0 for x in a) all(x > 0 for x in a)	std::ranges::any_of(a, [](int x) { return x > 0; }); std::ranges::all_of(a, [](int x) { return x > 0; });
b = [x * x for x in a] c = [x for x in a if x > 0]	std::vector<int> b; std::ranges::transform(a, std::back_inserter(b), [](int x) { return x * x; }); std::vector<int> c; std::ranges::copy_if(a, std::back_inserter(c), [](int x) { return x > 0; });
i = bisect_left(a, x)	auto it = std::lower_bound(a.begin(), a.end(), x); auto i = std::distance(a.begin(), it);

19. Исключения и ошибки

Осенний семестр: try, except, assert, optional, expected

Python	C++

try:
  x = int(s)
except ValueError:
  print("bad number")
	try {
	  int x = std::stoi(s);
	} catch (const std::invalid_argument&) {
	  std::println("bad number");
	}

try:
  f()
finally:
  close_resource()
	{
	  Resource r; // RAII: деструктор освободит ресурс
	  f();
	}

raise RuntimeError("bad state")	throw std::runtime_error("bad state");
assert x >= 0	assert(x >= 0);

def find(a, x):
  return i or None
	std::optional<std::size_t> find_index(
	  const std::vector<int>& a, int x
	);

def parse(s): return value_or_error	std::expected<int, std::string> parse(std::string_view s); // std::expected - C++23.

20. Файлы и приведение типов

Осенний семестр: text/binary files, paths, casts

Python	C++

with open("data.txt", encoding="utf-8") as f:
  text = f.read()
	std::ifstream file("data.txt");
	std::string text(
	  std::istreambuf_iterator<char>(file),
	  std::istreambuf_iterator<char>()
	);

with open("out.txt", "w", encoding="utf-8") as f: f.write("hello\n")	std::ofstream file("out.txt"); file << "hello\n";

with open("data.txt") as f:
  for line in f:
  print(line.rstrip())
	std::ifstream file("data.txt");
	for (std::string line; std::getline(file, line);) {
	  std::println("{}", line);
	}

with open("log.txt", "a") as f: f.write("new line\n")	std::ofstream file("log.txt", std::ios::app); file << "new line\n";

with open("data.bin", "rb") as f:
  data = f.read()
	std::ifstream file("data.bin", std::ios::binary);
	std::vector<char> data(
	  std::istreambuf_iterator<char>(file),
	  std::istreambuf_iterator<char>()
	);

from pathlib import Path p = Path("data") / "input.txt"	std::filesystem::path p = std::filesystem::path("data") / "input.txt";
int(x) float(x) str(x)	static_cast<int>(x) static_cast<double>(x) std::to_string(x)
# bytes объекта как набор байтов	// reinterpret_cast нужен редко и опасен: auto* bytes = reinterpret_cast<const std::byte*>(&obj);

21. Собственные структуры данных

Осенний семестр: список, стек, очередь, дерево

Python	C++

`stack = []`
`stack.append(x)`
`x = stack.pop()`
	`std::stack<int> stack;`
	`stack.push(x);`
	`int x = stack.top();`
	`stack.pop();`

`from collections import deque`
`q = deque()`
`q.append(x)`
`x = q.popleft()`
	`std::queue<int> q;`
	`q.push(x);`
	`int x = q.front();`
	`q.pop();`

`d = deque()`
`d.appendleft(x)`
`d.append(y)`
`d.popleft()`
`d.pop()`
	`std::deque<int> d;`
	`d.push_front(x);`
	`d.push_back(y);`
	`d.pop_front();`
	`d.pop_back();`

`heapq.heappush(h, x)`
`x = heapq.heappop(h)`
	`std::priority_queue<int> h;`
	`h.push(x);`
	`int x = h.top();`
	`h.pop();`
	`// По умолчанию max-heap.`

`# Связный список обычно не пишут вручную.`	`struct Node {`
	`  int value;`
	`  Node* next = nullptr;`
	`};`

`# Дерево как вложенные объекты.`	`struct Node {`
	`  int value;`
	`  std::unique_ptr<Node> left;`
	`  std::unique_ptr<Node> right;`
	`};`

`# Обход в ширину`	`std::queue<Node*> q;`
	`q.push(root);`
	`while (!q.empty()) {`
	`  Node* v = q.front();`
	`  q.pop();`
	`  // обработать v`
	`}`

22. Классы и объекты

Весенний семестр: инкапсуляция, конструкторы, методы

Python	C++

`class Point:`
`  def __init__(self, x, y):`
`  self.x = x`
`  self.y = y`
	`class Point {`
	`public:`
	`  Point(double x, double y) : x_(x), y_(y) {}`
	
	`private:`
	`  double x_;`
	`  double y_;`
	`};`

`p = Point(1, 2)`	`Point p{1, 2};` `auto q = std::make_unique<Point>(1, 2);`

`class Counter:`
`  def inc(self):`
`  self.value += 1`
	`class Counter {`
	`public:`
	`  void inc() { ++value_; }`
	
	`private:`
	`  int value_ = 0;`
	`};`

`@property`
`def value(self):`
`  return self._value`
	`int value() const {`
	`  return value_;`
	`}`

`def __str__(self):`
`  return f"({self.x}, {self.y})"`
	`friend std::ostream& operator<<(std::ostream& out,`
	`  const Point& p) {`
	`  return out << '(' << p.x_ << ", " << p.y_ << ')';`
	`}`

`def __eq__(self, other):` `  return self.x == other.x and self.y == other.y`	`bool operator==(const Point& other) const = default;`

`# Сборщик мусора освобождает объект позже.`	`~Point() = default;`
	`// Деструктор вызывается детерминированно:`
	`// при выходе объекта из области видимости.`

23. Наследование и полиморфизм

Весенний семестр: base class, virtual, override

Python	C++

class Animal:
  def speak(self):
  raise NotImplementedError
 	
class Dog(Animal):
  def speak(self):
  return "woof"
	class Animal {
	public:
	  virtual std::string speak() const = 0;
	  virtual ~Animal() = default;
	};
	
	class Dog : public Animal {
	public:
	  std::string speak() const override {
	  return "woof";
	  }
	};

animals = [Dog(), Cat()]
for a in animals:
  print(a.speak())
	std::vector<std::unique_ptr<Animal>> animals;
	animals.push_back(std::make_unique<Dog>());
	
	for (const auto& a : animals) {
	  std::println("{}", a->speak());
	}

isinstance(obj, Dog)	if (auto* dog = dynamic_cast<Dog*>(obj)) {
	  // obj действительно указывает на Dog
	}

super().__init__()	Derived() : Base() {
	  // тело конструктора Derived
	}

24. Шаблоны, концепты, лямбды

Весенний семестр: обобщенное программирование

Python	C++

def max2(a, b):
  return a if a > b else b
	template <class T>
	T max2(T a, T b) {
	  return (a > b) ? a : b;
	}

def print_all(*args):
  print(*args)
	template <class... Args>
	void print_all(const Args&... args) {
	  (std::print("{} ", args), ...);
	  std::println("");
	}

def add(a: int, b: int) -> int:
  return a + b
	template <class T>
	  requires std::integral<T>
	T add(T a, T b) {
	  return a + b;
	}

key=lambda x: x.score	[](const Student& x) {
	  return x.score;
	}

threshold = 10
f = lambda x: x > threshold
	int threshold = 10;
	auto f = [threshold](int x) {
	  return x > threshold;
	};

from typing import Protocol	template <class T>
	concept Printable = requires(T x) {
	  std::cout << x;
	};

25. Контейнеры STL

Весенний семестр: выбор контейнера по операции

Python	C++
list[int]	std::vector<int> // динамический массив std::array<int, 10> // фиксированный размер std::deque<int> // быстрые оба конца std::list<int> // двусвязный список

set()
x in s
s.add(x)
	std::set<int> s; // упорядочено
	std::unordered_set<int> us; // хеш-таблица
	s.contains(x); // C++20
	s.insert(x);

d = {}
d[key] = value
value = d[key]
	std::map<Key, Value> d;
	d[key] = value;
	auto value = d.at(key);
	
	std::unordered_map<Key, Value> fast;

for key, value in d.items():
  print(key, value)
	for (const auto& [key, value] : d) {
	  std::println("{} {}", key, value);
	}

from collections import Counter
cnt = Counter(a)
	std::unordered_map<int, int> cnt;
	for (int x : a) {
	  ++cnt[x];
	}

from collections import defaultdict g = defaultdict(list) g[u].append(v)	std::unordered_map<int, std::vector<int>> g; g[u].push_back(v);

Python	C++
`a[begin:end]`	`std::span<int> view(a.data() + begin, end - begin);` `// Невладеющий диапазон.`

26. Ranges и функциональный стиль

Весенний семестр: filter, map, views, pipelines

Python	C++

`squares = map(lambda x: x*x, a)`	`auto squares = a | std::views::transform(`
	`  [](int x) { return x * x; }`
	`);`

`positives = filter(lambda x: x > 0, a)`	`auto positives = a | std::views::filter(`
	`  [](int x) { return x > 0; }`
	`);`

`for x in reversed(a):`
`  ...`
	`for (int x : a | std::views::reverse) {`
	`  // ...`
	`}`

`for i in range(1, n):`
`  ...`
	`for (int i : std::views::iota(1, n)) {`
	`  // ...`
	`}`

`sum(x*x for x in a if x > 0)`	`auto values = a`
	`  | std::views::filter([](int x) { return x > 0; })`
	`  | std::views::transform([](int x) { return x * x; });`
	
	`int sum = std::accumulate(values.begin(), values.end(), 0);`

27. RAII и умные указатели

Весенний семестр: владение ресурсом вместо ручного delete

Python	C++
`obj = SomeClass()`	`SomeClass obj; // автоматическое время жизни`
`obj = SomeClass()` `# ссылка на объект хранится в переменной`	`auto obj = std::make_unique<SomeClass>();` `// единственный владелец объекта`

`a = b`	`a = b; // копирование, если тип копируемый`
	`auto p = std::move(q); // передача владения`

`# Объект нужен из нескольких мест.`	`auto p = std::make_shared<Node>();` `std::weak_ptr<Node> parent; // слабая ссылка без владения`

`with open(path) as f:`
`  use(f)`
	`{`
	`  std::ifstream f(path);`
	`  use(f);`
	`} // файл закрывается автоматически`

28. Модули, пространства имен, тесты

Весенний семестр и инструменты курса

Python	C++
`import package.module`	`#include "module.h" // классический путь` `import module; // C++20 modules, если настроены`

`# package/name.py`	`namespace course {`
	`  int f(int x);`
	`}`

`assert f(2) == 4`	`assert(f(2) == 4);`

`def test_square():`
`  assert square(3) == 9`
	`TEST_CASE("square") {`
	`  CHECK(square(3) == 9);`
	`}`
	`// Пример в стиле doctest/Catch2.`

`python main.py`	`g++ -std=c++23 main.cpp -o main` `./main`
`pip install package`	`// В C++ зависимости обычно подключают через` `// систему сборки: CMake, vcpkg, Conan и т.п.`

Источники для сверки

- Python documentation: Tutorial and Library Reference, https://docs.python.org/3/
- cppreference: C++ language and standard library reference, https://en.cppreference.com/w/cpp
- C++ Core Guidelines, https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines
- Bjarne Stroustrup, Programming: Principles and Practice Using C++, 3rd edition, 2024.
- Bjarne Stroustrup, A Tour of C++, 3rd edition, 2022.