# **پارادایم‌های سی پلاس پلاس**

C++ یک زبان چندپارادایمی است، به این معنا که این زبان از چندین پارادایم برنامه‌نویسی حمایت می‌کند. دو پارادایم اصلی که در C++ مورد استفاده قرار می‌گیرند، پارادایم دستوری (Imperative/Procedural) و پارادایم اعلانی (Declarative) هستند. برنامه‌نویسان می‌توانند از دستورات سطح پایین برای کنترل جریان برنامه استفاده کنند (پارادایم دستوری)، همچنین می‌توانند از ویژگی‌هایی مانند شیءگرایی (Object-Oriented) و استفاده از STL (که از پارادایم جنریک نیز حمایت می‌کند) به صورت اعلانی استفاده کنند. حتی در برنامه‌های C++ ساده، شما می‌توانید الگوهای دستوری و اعلانی را مشاهده کنید. که در ادامه بررسی خواهد شد.

## **پارادایم دستوری**

برنامه‌نویس دستورات اجرا را به ترتیب تعیین می‌کند.

#include <iostream>

int main}()

std::cout << "Enter a number ";

int num;

std::cin >> num;

for (int i = 0; i < num; ++i)

std::cout << i<<" " ;

}

دو پارادایم مهم برنامه نویسی دستوری، برنامه نویسی رویه ای و شی گرا است که این زبان از هر دو پشتیبانی میکند.

### **برنامه نویسی رویه‌ای**

برنامه‌های ساده می‌توانند به صورت رویه‌ای تعریف شوند. در این حالت، برنامه‌ها از توابع، متغیرها و دستورات دستوری استفاده می‌کنند.

#include <iostream>

void printNumbers(int n) {

for (int i = 0; i < n; ++i)

std::cout << i ;

{

int main(){

int num;

std::cout << "Enter a number:

std::cin >> num;

printNumbers(num);

{

### **برنامه نویسی شی گرا**

#include <iostream>

class NumberPrinter {

public:

void printNumbers(int n) }

for (int i = 0; i < n; ++i)

std::cout << i ;

}

{;

int main() {

int num;

std::cout << "Enter a number;

std::cin >> num;

NumberPrinter printer;

printer.printNumbers(num);

}

## **پارادایم اعلانی**

برنامه‌نویس تعریف می‌کند که چه خصوصیاتی باید داشته باشد. و الگوریتم ها را بصورت اعلانی استفاده می‌کند.

#include <iostream>

#include <vector>

#include <algorithm>

int main}()

std::vector<int> numbers= {3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5}

std::vector<int>::iterator it = std::begin(numbers);

std::sort(std::begin(numbers), std::end(numbers));

for (const auto& element : numbers) {

std::cout << element<< " ";

}

دو پارادایم مهم برنامه نویسی اعلانی، برنامه نویسی تابعی و منطقی است که این زبان از تنها از برنامه نویسی تابعی پشتیبانی میکند.

#include <iostream>

#include <vector>

#include <algorithm>

int main}()

std::vector<int> numbers= {3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5};

std::for\_each(numbers.begin(), numbers.end(), [](int x){

if (x % 2 != 0)

std::cout << x;

({;

}

## **پارادایم همروند**

برای زبان سی پلاس پلاس ابزارها و ویژگی‌هایی برای برنامه‌نویسی همروند فراهم شده است. این امکانات به برنامه‌نویسان این امکان را می‌دهد تا بخش‌های مختلف برنامه‌ها را به صورت همزمان اجرا کنند یا از ویژگی‌های پردازش موازی بهره‌مند شوند

* Thread Support Library (TLS) دارای کتابخانه‌های مختلف برای کار با موضوعات (threads) است که از این کتابخانه‌ها می‌توان برای ایجاد و مدیریت همزمانی در برنامه‌ها استفاده کرد. مانند <tread > و <mutex>

#include <iostream>

#include <thread>

void printNumbers(int start, int end) }

for (int i = start; i < end; ++i)

std::cout << i

{

int main() {

std::thread t1(printNumbers, 0, 5);

std::thread t2(printNumbers, 5, 10);

t1.join();

t2.join();

{

* Parallel Algorithms (C++17 onwards): در نسخه‌های جدیدتر C++، الگوریتم‌های موازی ارائه شده‌اند که به برنامه‌نویسان امکان اجرای عملیات‌ها بر روی داده‌ها به صورت موازی را می‌دهند. این ویژگی به برنامه‌نویسان امکان بهره‌مندی از پردازش موازی را در زمان اجرا فراهم می‌کند.

#include <iostream>

#include <vector>

#include <algorithm>

#include <execution>

int main() {

std::vector<int> numbers {3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5}=;

std::sort(std::execution::par, numbers.begin(), numbers.end());

for (const auto& element : numbers)

std::cout << element

{

## **پارادایم توزیع شده**

این زبان برنامه نویسی به صورت مستقیم توسط برنامه‌های معمولی اجرا می‌شود و قابلیت توزیع زیرساخت یا اجرای توزیع‌شده را به صورت پیش‌فرض ندارد. اما با استفاده از کتابخانه‌ها و فریم‌ورک‌های مناسب در C++، می‌توانید برنامه‌های توزیع‌شده ایجاد کنید. یکی از کتابخانه‌های معروف برای توزیع‌شده کردن، MPI (Message Passing Interface) است. MPI یک استاندارد ارتباطات برای برنامه‌های موازی و توزیع‌شده است که اجازه ارسال و دریافت پیام‌ها بین چندین فرآیند را فراهم می‌کند.همچنین، برخی از پروژه‌ها و فریم‌ورک‌های متن‌باز دیگر نیز برای توزیع‌شده کردن برنامه‌ها در C++ وجود دارند. به عنوان مثال، ZeroMQ یا Apache Thrift.

نکته مهم این است که توزیع‌شده کردن برنامه نیاز به طراحی معماری خاصی دارد و استفاده از کتابخانه‌ها تنها یک بخش از این فرآیند است. توزیع‌شده کردن برنامه‌ها شامل مسائلی نظیر هماهنگی داده، مدیریت خطا، مدیریت منابع و ارتباطات توزیع‌شده است که باید با دقت بررسی شود.