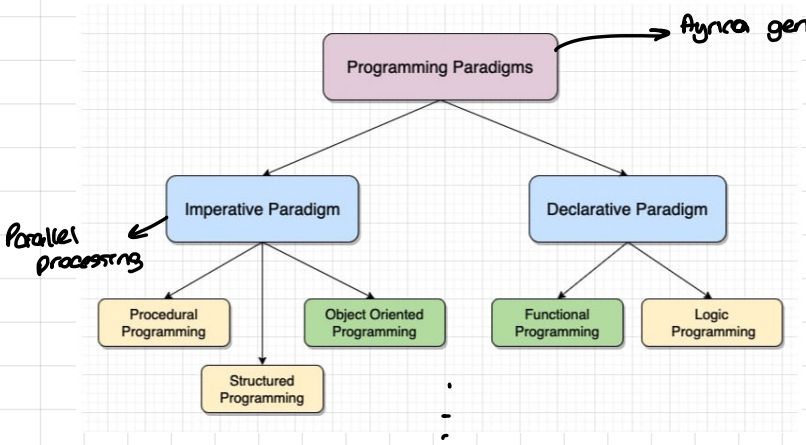
• C dili → prosedürel programlama dilidir.

→ Bu Programming Paradigm olarak geçer



Ayrıca generic paradigm da var.

Parallel processing

→ Ancak C++ multi-paradigm bir dil / OOP, Generic, Procedural ....

→ C++ verim kritik (efficiency) gereken yerlerde kullanılır. → Audio/video, Fin tech, Sistem programlama vs.... / Java C# kullanıcı rahatken C++ daha sistemle yakınlık

→ C'de olup C++'da olmayan kavramlar:

- Compand Literal:
- Variable Length Array → Kitaplardan bak.
- Designated Initializers

* Konu 1: C++ ve C Arasındaki Incompatible Durumlar:

• Implicit Int: Gizli int → C++'da geçerli değil. C'de geçerli

Kapalı int

Örtük int

• Normalde type belirtmeleri yapılması gereken yerlerde, C' dili default olarak signed int varsayar.

func(void) C'de yazılabilirken, C++ yazmayız

↳ return değerini singed int kabul ettir

• Fonksiyon Parametre Tanımı → C dilinde:

- int foo( ) → farklıdır.
- int func(void) → foo(1,2,3) .. gibi çağırılabilir. YANMA!!

→ C++ dilinde:

- int foo( ) → Aynıdır.
- int func(void)

• Bildirimler: → C dilinde:

• Scope içinde statementlar sonra declaration yapılmaz. (C 99'da yok)

→ C++ da:

• Bu yapılabilir.

• Old-style Function Definition: → C dilinde:

```
double sum (d1, d2, d3)
double d1, d2, d3
{
    return d1+d2+d3
}
```

→ Fonksiyonun ana bilgisi olmadan önce değişken tanımı mümkündü (C 99'a kadar)

→ C++ dilinde

• Bu yok.

→ Loose Type Control: C' dilinde derleme zamanında type control çok zayıftır
C++ dilinde oldukça katı

→ Aritmetik Türlerden Pointer'a Implicit Dönüşüm: C dilinde yapılabilir / `int x = 10; int * ptr = x;`
C++ dilinde syntax error

→ Type cast Çeşitleri: C++'ın kendine özgü castingleri var
- Static cast
- Const cast
- reinterpret cast
- dynamic cast

C'de türler arasında typecast var.

→ Karakter Sabitleri (character literal): C'de 'A', '\n', '\101' gibi değişkenlerin türü int
C++'da türü char.

→ String Sabiti (string literal): "kerim" → C dilinde char[6] (char arraydir)
Kullanılırken ancak, array decay olur. char* olarak işlem görür. (Array to char*)

→ C++'da const char[6]
Array const char* 'a decay olur.

Array Decay / Array to Pointer Conversion:

`int a[10] = {0};`
`&a[0]` → Dizinin ismi dizinin ilk elemanına döndü

→ Const Keyword: C dilinde const değişken init edilmek zorunda değil. Değer atanmaz zorunda değil.

C++ dilinde zorunlu !!!

→ Linkage: `int x = 10;` → external linkage
yani tüm modüllerde
link time ve run time da
aynı değişkeni ifade eder

→ eğer global olarak, static diye tanımlamadıysak internal linkage

- C++'da global const nesneler → C dilinde external
→ C++ dilinde internal

→ Initializing Different Const Qualifiers: * const int x=10 → C'de legal
int *p = &x → C++'da illegal

→ Boolean: C 99'da _Bool eklendi. Aynı bir tür. (stdbool kütüphane).

bool flag = true
_Bool flag = true 'dur aslında,

C++ dilinde bool ayrı bir tür (distinct type)

→ true ve false bir macrodur
enum olarak da tanımlanabilir.

* Karşılaştırma Operatörleri 10>5 → C'de signed int olarak işlem görür. → C'de 0
→ C'++'da bool → C++ bool

* C++'da bool'dan aritmetik türlere dönüşüm var → int x=0
bool b = x → false

int x=1
bool b= x → true

* C++ 17'ye kadar bool flag = false ++flag diyebiliyorduk. C++ 17'de kaldırıldı. Modern c++'da syntax error.

* C++'da pointer to bool da var. → Null pointer false.
→ Diğer tüm pointerlar true.
(tam tersi her zaman syntax error)

→ NULL: C dilinde NULL bir macrodur. Çoğu compiler bunu null pointer olarak kullanır. C++'da NULL kullanılmadı.
ve Null ptr conversion

C++ en başında int *p = 0 gibi kullanımı vardı → Null pointer yerine 0

Modern C++'da nullptr geldi. (nullptr_t type'ında). Artık NULL'da yok 0'a eşitlene de yok.
↓
true / false gibi bir sabit.

• int *p = nullptr; ✓
int x = nullptr; ✗

• bool x = nullptr ✗

• C dilinde int *p=0 'da null pointerdır.
int *p; diye bırakmak C dilinde de → global olarak tanımlıdaysa null pointer yoksa çöp değer