Parametrų perdavimo mechanizmai

Value vs. **Pointer** vs. **Reference**

1. Perdavimas pagal reikšmę (*call/pass by value*)

Tai vienintelis "tikras" C kalbos mechanizmas.

- Į funkciją perduodamos **kopijos**.
- Funkcija turi savo lokalius kintamuosius (`temp`, `x`, `y`).
- Originalūs kintamieji `main` funkcijoje **NĖRA** keičiami.

Tai saugu, bet neefektyvu dideliems objektams ir netinka, kai norime pakeisti originalą.

```
void swap_val(int x, int y) { // Gauna KOPIJAS (x=1, y=2)
    int temp = x;
    x = y;
    y = temp;
} // Čia kopijos sunaikinamos. Originalai a ir b nepakito.
```

Python Tutor

2. C rodyklėmis imituojamas perdavimas pagal nuorodą ("*call/pass by reference*")

C kalboje norėdami pakeisti originalą, turime "gudrauti" naudodami adresus.

1. Parametrai tampa rodyklėmis (`int*`).
2. Naudojame **išadresavimo** (dereference) operatorių `*`, kad pasiektume reikšmę.
3. Kviečiant funkciją, siunčiame **adresus** (`&a`).

```
void swap_ptr(int* x, int* y) { // Gauna ADRESUS
    int temp = *x; // Paimk reikšmę iš adreso x
    *x = *y;       // Įrašyk y reikšmę į x adresą
    *y = temp;     // Įrašyk temp į y adresą
}

// Kvietimas: swap_ptr(&a, &b);
```

Vizualizuoti atmintį (Python Tutor)

---

3. C++ perdavimas pagal nuorodas (*call/pass by reference*)

C++ įveda **tikrąjį** perdavimą pagal nuorodą. Nuoroda (`int&`) – tai **pseudonimas** (alias). Tai lyg antras vardas tam pačiam kintamajam.

- Sintaksė švaresnė (nereikia `*` ir `&` funkcijos viduje).
- Kompiliatorius užtikrina, kad nuoroda visada į kažką rodytų.

```
void swap_cpp(int& x, int& y) { // x yra 'a' pseudonimas
    int temp = x;
    x = y;         // Keičia patį originalą
    y = temp;
}

// Kvietimas: swap_cpp(a, b);  <- Atrodo paprastai!
```

Išbandyti kode (Godbolt)

Apibendrinimas: 3 būdai

Savybė	By Value (C/C++)	Pointer (C stilius)	Reference (C++ stilius)
Deklaracija	`void f(int x)`	`void f(int* x)`	`void f(int& x)`
Kvietimas	`f(a)`	`f(&a)`	`f(a)`
Veiksmas	Dirba su kopija	Dirba su adresu	Dirba su originalu
Sintaksė	Paprasta	Sudėtinga (*, &)	Paprasta
Verdiktas	Saugus, bet "lokalus"	Galingas, bet pavojingas	**Modernus standartas**

```
// Tik C++ leidžia rašyti taip švariai:
int main() {
    int a=1, b=2;
    swap_cpp(a, b); // a=2, b=1
}
```