Deklaracije

- Veoma često je potrebno deklarisati varijablu koja se inicijalizuje vrednošću nekog izraza, uključujući i poziv funkcije. Relativno često, a posebno u slučaju korišćenja šablonskih klasa iz standardne biblioteke i njihovih operacija, tip takve povratne vrednosti je veoma složen i uključuje parametrizovane šablonske klase, cv-kvalifikatore i slično; slična situacija je i u definisanju samih metoda složenih šablonskih klasa, gde tip neke varijable zavisi od parametara šablona
- U takvim situacijama programeru može biti teško i nepraktično da obavezno precizira taj tip u deklaraciji varijable, ili bi time program postao teže čitljiv. Sa druge strane, prevodilac tip te povratne vrednosti izraza ili funkcije sasvim pouzdano zna za vreme prevođenja
- Za ovakve potrebe, moguće je deklarisati varijablu bez eksplicitnog navođenja tipa, uz ključnu reč *auto*. Time varijabla ima i dalje statički definisan tip, ali taj tip određuje sam prevodilac na osnovu tipa inicijalizatora te varijable, i dalje ga koristi kao da je on eksplicitno naveden u deklaraciji
- Ovo se može koristiti u bilo kojoj deklaraciji koja definiše varijablu sa inicijalizatorom, što može da olakša pisanje koda i učini ga kompaktnijim. Na primer:

```
template<typename T>
T sum (const list<T>& array) {
  T sum = 0;
  for (auto it=array.cbegin(); it!=array.cend(); it++)
    sum += *it;
  return sum;
}
```

Ova funkcija vraća objekat - *iterator* koji je u stanju da iterira kroz kolekciju (ovde listu) sve do kraja i obezbeđuje pristup do tekućeg elementa kolekcije

Tačan tip ovog objekta nije ni bitan, već su bitne samo operacije koje on obezbeđuje (prekolopljeni operatori ++ za pomeranje na sledeći element i * za pristup tekućem). Tip ovog objekta prevodilac zna - to je tip povratne vrednosti ove funkcije

Deklaracije

- Ista stvar može se iskoristiti za lakše pisanje koda bez obaveznog određivanja tipa rezultata izraza, koji može zavisiti od tipova operanada; na primer:

```
auto z = x*exp(y);
auto u = static_cast<decltype(v)>(z);
```

- U nekim situacijama, posebno kod šablonskih funkcija, povratni tip funkcije je teško ili nemoguće odrediti. Tada se povratni tip funkcije može odrediti na osnovu tipa izraza i specifikatora *decltype* navedenog iza znaka ->:

```
template<typename T, typename U>
auto add(T t, U u) -> decltype(t + u) {
    return t+u;
}
```

- Funkcija može imati povratni tip koji nije eksplicitno naveden, već se takođe zaključuje, ali na osnovu tipa izraza iza naredbe *return*:

```
template<typename T, typename U>
auto add(T t, U u) {
    return t+u;
}
```