Inline funkcije

- Da bi ugradio kod *inline* funkcije na mestu poziva, prevodilac mora da ima kompletnu definiciju (tela) te funkcije. Zato su za ove funkcije dozvoljene *višestruke definicije* u istom programu, ali samo po jedna u svakoj jedinici prevođenja; po pravilu, definicije *inline* funkcija navode se u zaglavljima
- Čak i ako se nalaze u različitim jedinicama prevođenja, višestruke definicije *inline* funkcija odnose se na istu funkciju i semantika te funkcije se ne menja: ona će imati istu adresu, lokalni statički objekti su jedinstveni za sve definicije (sve iste definicije takvog objekata odnose se na jedan jedinstveni entitet) i slično
- Sve definicije iste *inline* funkcije u različitim jedinicama prevođenja moraju biti identične; ako nisu, ponašanje programa je nedefinisano
- Sa jedne strane, prevodilac ne mora ispoštovati zahtev za ugrađivanje *inline* funkcije u kod pozivaoca, već generisati kod za uobičajen poziv. Sa druge strane, ne brani se prevodiocu da kod i neke druge funkcije koja nije *inline* ugradi na mesto poziva kao optimizaciju. Zbog toga se osnovni smisao *inline* funkcija zapravo gubi, pa *inline* formalno ne znači ništa više nego to da su dozvoljene višestruke definicije iste funkcije u različitim jedinicama prevođenja u istom programu
- Zbog toga je u jezik uveden i koncept *inline* statičkih varijabli (objekata ili referenci): dozvoljene su višestruke definicije *inline* statičke varijable u različitim jedinicama prevođenja istog programa (ali samo po jedna definicija u jednoj jedinici), pod uslovom da su te definicije identične; sve one se odnose na isti entitet
- Ovo omogućava kompletne definicije biblioteka u fajlovima zaglavljima, bez potrebe za definisanje objekata u *.cpp* fajlovima

```
inline istream cout;
```

Podrazumevani argumenti

- Vrlo često je potrebno da funkcija ima nekoliko varijanti, odnosno da se može pozvati sa nekim argumentom ili bez njega, pri čemu izostvaljeni argument treba da uzme neku *podrazumevanu vrednost* (*default argument*); na primer:

```
complex::complex(double re=0.0, double im=0.0);
```

```
template <typename T>
list& list::insert (T t, int at=0);
```

- Ako se neki stvarni argument u pozivu ovakve funkcije izostavi, taj argument dobiće podrazumevanu vrednost navedenu u deklaraciji funkcije:

```
complex c1(1.,1.), c2(1.), c3;

list<complex> lst;
lst.insertAt(c1).insertAt(c2,1).insertAt(c3);
```