Lektion 6 – Time Is Flying Never to Run

Time Is Just a Prison

En viktig del i programmering är *tidskomplexitet*. Vanliga datorer kan utföra ungefär $100000000 = 10^8$ operationer per sekund och i många problem är det lätt att råka överskrida den gränsen om man inte tänkt till ordentligt.

För att ta reda på hur många operationer ett program gör är det enklaste sättet att kolla hur många gånger ens loopar körs.

Ett exempel där man måste tänka på tidskomplexiteten är i följande spel: Din kompis tänker på ett tal mellan 1 och 10^{10} som du ska komma på med så få antal gissning som möjligt, om du gissar fel får du veta ifall talet du gissade på var för litet eller för stort.

Du skulle kunna gissa på $1, 2, 3, ...$ upp till 10^{10}, men om din kompis tänker på 10^{10} måste du gissa 10^{10} gånger! Så många gånger hinner inte ens en dator gissa på en sekund. Istället kan man gör något som kallas binärsökning. Det går ut på att man börjar med att gissa på $\frac{10^{10}}{2}$, om detta är för stort vet man att talet är mellan 1 och $\frac{10^{10}}{2}$. Nu vi igen gissa i mitten på detta intervallet, också vidare. Det visar sig att man med den här metoden kan komma på vilket tal ens kompis tänker på med max 35 gissningar.

Heltalsdivision

Om du har två variabler av typen `int` och delar dem med varandra kommer du i programmet alltid få en annan `int`, fast att talen inte är delbara med varandra. Exempelvis kommer `cout << 8/3 << endl;` skriva ut 2 fast att $\frac{8}{3} = 2,666...$ När du delar med något som gör att det inte går jämnt ut avrundar datorn neråt till närmaste heltal. Detta är samma sak som så många gånger hela det man delar med får plats i talet man delar.