Laborationsdata

Namn: Thomas Nordenmark
LoginID: thno1200
Lab nr: 3
Datum: 2013-12-02
Utvecklingsmiljö: Arch Linux x64, QT Creator IDE 2.8.1, GCC 4.8.2

Filer

lab3_upg1.cpp
lab3_upg2.cpp

Slutsatser och kommentarer

De två första programmen var inga större problem förutom att jag körde fast lite innan jag kom på att man kan dela upp strängen med namnen, vända på den och sortera på efternamn → förnamn den vägen.

Det tredje programmet med rot13 och rot7 får jag inte att fungera efter 2 dagars läsande och programmerande. I uppgiftsbeskrivningen står det att man ska hantera tecknen 32-255 vilket inte är meningen med ett Caesarchiffer. Anledningen till att rot13 ska skifta tecknen just 13 bokstäver är för att det amerikanska alfabetet innehåller 26 tecken. Chiffret kan alltså krypteras upp genom att köra samma algoritm igen utan att ens vända på det (vilket i sin tur blir hiskeligt fult då man inte får använda funktioner). När det gäller olika nycklar var femte tecken tänkte jag att man testar mot modulus 5 och byter nyckel om det inte är någon rest och det inte är första varvet i loopen. Men får inte detta att fungera åt andra hållet.

Jag fick ett program att fungera med en enda sträng utan mellanslag och endast fram till gemenen u, sen spårade ASCII-tabellen ur då jag inte får den att rotera från början. Vilket är mycket märkligt då u har ASCII-värde 117. Försökte lösa det genom att kolla det sammanlagda värdet på strängen samt nyckeln, ta modulus av strängen + nyckeln och addera 32 men i vanlig ordning håller man på att bli skogstokig på extended ASCII när man använder ett system med UTF-8. Det är ett olidligt strulande med teckentabeller, fonter och terminaler och mest bara frustrerande så man tappar lusten helt. Helt omöjligt att försöka felsöka och räkna ASCII-värden för att utröna vad som går fel också. Det känns mest som om alla dessa ASCII-uppgifter är till för folk som fortfarande kör Code Page 437 eller någon annan gammal DOS-teckentabell som ingen längre använder. Därför abdikerar jag på den här uppgiften och kastar ut den genom fönstret.

Strunta i ASCII eller håll er till USASCII 32-127 (som är samma i UTF), för det här tramset är bara onödigt när UTF varit standard sen 1990-talet (och medför helt andra problem).