Datavetenskap

Opponent:

Martin Landälv

Respondenter:

Reza Javanbakhti

Jimmy Pesola

Statisk detektering av minneshanteringsfel i C/C++

Oppositionsrapport, C/D-nivå

2006:13

1 Sammanfattat omdöme av examensarbetet

Jag tycker att examensarbetet i stort ser mycket bra ut. Det är ett intressant ämne som, om man läser deras uppsats, även har gett dem nya och djupare kunskaper, inom bl.a. användning av Flex och Bison.

Uppsatsen är väl skriven och språket håller hög kvalitet. Det finns vissa saker att anmärka på men i det stora hela är det ett mycket väl genomfört arbete.

2 Synpunkter på uppsatsen knuten till examensarbetet

Uppsatsen beskriver hur problem och fördelar med statisk kontra dynamisk detektering av minnesläckage samt hur ni gått tillväga för att implementera er egen applikation för statisk detektering av minneshanteringsfel. Den röda tråden får man i första kapitlet och den följer sedan med till slutet.

2.1 Titel

Titeln är ”Statisk detektering av minneshanteringsfel i C/C++”, vilket är en lämplig titel eftersom det är just det uppsatsen handlar om.

2.2 Uppsatsens disposition

Dispositionen är generellt sett mycket bra. Kapitlen avlöser varandra vilket gör att det är lätt att följa den röda tråden, men vissa kapitel är onödigt lite uppdelade; det är långa stycken och även i vissa fall onödigt mycket kod som man behöver sätta sig in i och förstå. Jag tycker också att ni med fördel kan byta plats på kapitel två och tre. En analys innan man ser kravspecifikationen kan få läsaren att förstå kraven lite djupare.

Styckeuppdelningen är svår att följa på vissa ställen, eftersom ni då varken använt indentering eller blankrad.

2.3 Begreppsapparat

Uppsatsen är skriven på ett relativt lättförståeligt språk, men vissa konstiga ord finns med som inte förklaras. Man bör komma ihåg att uppsatsen är skriven för personer som har kunskap inom datavetenskap, men vissa uttryck kan ändå anses vara okända.

2.4 Argumentering och slutsatsdragning

Argumenteringen är bra genomförd, och övertygar för det mesta. Kraven är väl specificerade och ni beskriver också väl varför kraven och begränsningarna finns, ett undantag finns att läsa i avsnitt 3.4.

2.5 Sammanfattningen

Sammanfattningen är bra och tydlig. Det är bra att ni nämner att ni tagit idén på arbetet från företaget Saab Aerotech, men att ni att även haft egna idéer och att det är ett eget arbete.

Ni skriver också om behovet att kunna analysera kod efter eventuella läckage och påpekar vinsterna av att kunna göra det statiskt, men nämner även problemen som finns och jämför med dynamisk detektion.

Ni beskriver hur er prototyp fungerar, att ni använder hjälpverktygen Flex och Bison, vilka språk ert program stöder och att det kan detektera minnesläckage, felaktiga avallokeringar av minne, dangling pointers samt läsning och skrivning till ogiltiga minnesområden.

2.6 Språkbehandling

Språket håller relativt hög klass. Det förekommer – ganska många – fall där ni valt att använda semikolon istället för kolon; den första förekomsten finns i tredje stycket av den svenska sammanfattningen. Jag föreslår att ni kollar i boken Svenska Skrivregler om i vilka sammanhang kolon respektive semikolon ska användas.

Jag hittade bara en förekomst av talspråk och det var användningen av ”ju” på sida nio.

Jag hittade inte en enda särskrivning, vilket var väldigt trevligt! Däremot hittade jag två fall där ni utelämnat bindestreck för underförstådd orddel.

2.7 Referat och källförteckning

Ni har hitta många källor till ert arbete. De flesta är dock webbreferenser, vilket inte är så konstigt med tanke på ämnet. Några referenser är dock otydliga, bland annat referens 16 som endast hänvisar till google.com.

2.8 Övriga kommentarer

Ni skriver i sammanfattningen och även i avsnitt 1.2 (Mål för projektet) att er prototyp kan detektera felaktig avallokering av minne samt läsning från och skrivning till ogiltiga minnesområden, detta visas dock inte i något av era testfall.

3 Genomgång av uppsatsen kapitelvis

3.1 Inledning

Inledningen beskriver på ett bra sätt varför ni valt arbetsuppgiften, vilka alternativa program som finns och för- och nackdelar med statiskt detektering jämfört med dynamisk.

Ni beskriver även målet med projektet: vad ni önskar att er applikation ska kunna detektera för sorters fel när den är färdigutvecklad.

I avsnitt 1.1 har ni gjort ert vanligaste fel – felaktig styckeindelning. Boken Svenska Skrivregler tar upp de två sätten som finns i det svenska språket för styckeindelning: indrag och blankrad. I uppsatsen används ofta enbart radbrytning, men även blankrad.

3.2 Kravspecifikation

Kravspecifikationen är väl utformat och beskriver användbarhetskrav liksom funktionella krav.

I kapitlets inledning förekommer styckeindelningsfelet.

Jag tycker att ni kan skippa att dela in avsnitt 2.1 i ytterliggare underkapitel. Alla underkapitel beskriver användarkraven och kan med fördel placeras under samma avsnitt, då slipper man dessutom små kapitel på tre till fyra rader. Det finns trots allt en mer detaljerad beskrivning av användargränssnittet som bilaga.

Kapitel två och tre kan byta plats tycker jag. Då hinner läsaren läsa er analys och förstår då bättre varför ni har vissa av kraven.

3.3 Analys av projektet

I detta kapitel analyseras projektet. Ni berättar om befintliga system för minneshanteringsfel, berättar varför ni valt att göra konsolbaserat program istället för att använda ett GUI.

I avsnitt 3.1 tycker jag att det bör skapas underrubriker för varje verktyg som testas därför att avsnittet är långt och det blir svårt att snabbt se vilket stycke som motsvarar vilket verktyg. I avsnittet använder ni blankrad för att åtskilja styckena, men radbrytning förekommer i några stycken – detta bör åtgärdas!

”Run-Time Checking” upprepas för olika verktyg. Jag tycker det är onödigt att överhuvudtaget nämna den engelska termen när det är beskrivet att det ”sker under under körningen av kompilerad kod”. Upprepningen förekommer i beskrivningen av Visual Leak Detector och där används dessutom en annan term och denna gång med små bokstäver initialt – båda förekomsterna kan tas bort utan att det påverkar läsaren.

I beskrivningen av YAMD står det att en ”nackdel är att det [YAMD] kräver stora mängder virtuellt minne […]”, men det står inte i förhållande till vad de stora mängderna är.

På sida 9, rad 3, är ett kommatecken utplacerat före ”och”. Detta lilla ”fel” förekommer på fler ställen i uppsatsen och bör rättas till. Det är inte ett speciellt allvarligt fel, men användning av kommatecken före och är mycket ovanlig i svensk skrift, däremot ofta förekommande i engelsk.

I beskrivningen av verktyget CMemLeak står det att verktyget är avsett för ”applikationer skrivna i C och ska fungera i de flesta utvecklingsmiljöer […]”. När man läser ordet *ska* får man känslan av att det inte är säkert i vilka miljöer verktyget fungerar. Har ni inte testat eller utgår det inte av verktygets beskrivning?

På sida 13 förekommer ordet ”annotations” utan beskrivning. Annotations känns som en konstig form att använda; varför inte använda anteckningar, anmärkningar eller notiser? Om inte, ge en förklaring. I ert sammanhang representerar ”annotations” den engelska termen; annotationer kan användas om inga av mina ovanstående förslag duger.

På sida 13 förekommer även felaktig placering av punkt (näst sista raden). I de fall en parentes utgör en egen mening ska punkten placeras innanför, men i fall då parentesen är en del av en annan mening ska den placeras utanför.

3.4 Konstruktionslösning

Kapitlet beskriver hur ni gått tillväga för att konstruera ert program.

”En statisk analys […]” som börjar på femte raden i avsnitt 4.1 borde inleda ett nytt stycke, likaså ”Eftersom minne kan allokeras vid […]” lite längre ned.

På sida 17 nämns för första gången ”lexemes”, vilket är det engelska namnet för lexem. Eftersom det finns en motsvarighet i svenska bör den användas.

I avsnitt 4.4 där antaganden för konstruktionslösningen nämns finns de olika antaganden samt förklaringar; det gäller i alla punkter förutom 5:an, där en förklaring skulle vara önskvärd.

I avsnitt 4.5 står det att ni inte lyckades få Flex och Bison att fungera med C++-kod utan därför var tvungna att generera lexern och parsern i C-kod. Vad innebär det i praktiken? Hade implementationen skett på något annat sätt om det hade fungerat med C++-kod?

3.5 Implementation

I kapitlet beskrivs hur verktyget är implementerat, i vilken miljö och hur ni använder er av hjälpverktygen Flex och Bison. Även teoretiska lösningar som ni inte hunnit implementera beskrivs.

I andra raden på sida 24 har ett bindestreck utelämnats; ”C- eller C++-kompilator” ska det stå. På fjärde raden på samma sida står det ”C/C++-kompilatorer”, det ska vara ”C-/C++-kompilatorer” eftersom ”/”-tecknet i detta fall utläses som ”och”.

På sida 25 förekommer återigen det engelska ordet *lexemes*. Även ”(rules)” på rad två kan tas bort eftersom det svenska uttrycket regler står innan.

Det finns ganska många kodbitar med i detta kapitel och indenteringen på dem behöver kollas över. I vissa fall är även kodexemplen onödiga, exempelvis hur tokens returneras till parsern (börjar längst ned på sida 25); det räcker att ta med något exempel och sen förklara lite bättre hur variablerna yylval och yytext hänger ihop.

I fjärde stycket på sida 27 står det ”[…] kastas bort i Flex-filen.”, det ska vara Flex-filen (tror jag).

Generellt för hela kapitlet är att det är onödigt mycket kod. Koden är bra för att man verkligen ska förstå hur de olika delarna är uppbyggda men jag hade föredragit mer pseudokod, om det verkligen inte är nödvändigt att visa det exakta koden. Jag har väldigt svårt att förstå hur det är uppbyggt när jag ser koden på sida 28 när jag inte vet hur

a_add_item_by_details-funktionen fungerar exempelvis – jag är nog inte den enda. Mindre kod – mer pseudokod!

I avsnitt 5.6 i det nedersta stycket kan orden ”än” och ”här” strykas.

3.6 Tester

Kapitlet beskriver olika testfall och resultatet som fås vid användning av prototypen.

Testfallen är bra och beskriver tydligt hur väl prototypen fungerar. Det saknas dock testfall för kontroll av felaktiga avallokeringar samt skrivning och läsning till ogiltiga minnesområden.

Storleken på bl.a. heltal varierar beroende på arkitektur vilket är något som ni inte tagit upp. Ni förutsätter att ett heltal är 4 bytes.

Det skulle med fördel vara med radnummer på koden som visas i testexemplen, eftersom resultaten hänvisar till radnummer.

I testfall 3 ger prototypen varning om eventuellt minnesläckage i if-sats. I kapitel tre (sida 19) står det att varningar kommer om det inte går att utvärdera if-satser så varför kan inte prototypen utvärdera if-satsen i det exemplet då ni använder ”0” och ”size”, som är satt till 10, som parametrar till allocateMemory? Om inte värden för variabler lagras, då det går, ger prototypen alltid varningar och då borde det som står på sida 19 ändras.

Testfall 4 innehåller onödigt mycket kod för att inte vara som bilaga. Om det inte går att utföra samma test med mindre kod tycker jag det ska placeras i bilagan och att ni sen hänvisar dit.

3.7 Resultat och rekommendationer

Kapitlet beskriver hur slutprodukten blev i jämförelse med era förväntningar. Även problem som uppstått och rekommendationer nämns.

I sista stycket på sida 59 tycker jag att ni med fördel kan nämna de sju listorna igen. Sju olika listor är inte lätt för någon som läser igenom rapporten en gång att komma ihåg. Antingen formulera om den meningen eller skriv vilka listorna är igen.

3.8 Summering av projektet

Det här kapitlet handlar – eller borde handla – om summering av projektet. Första stycket och halva andra är helt okej, men sen ”Vid analys av källkod som finns […]” börjar ni beskriva begränsningar och problem, vilket inte hör hemma i summeringen av projektet.

Längst ned på sida 61 förekommer återigen felaktig styckeindelning. Jag har inte rapporterat varenda förekomst av felaktig styckeindelning eftersom de förekommer på olika ställen i hela uppsatsen.

3.9 Övriga kommentarer

Jag tycker uppsatsen borde gås över för att åtgärda felaktig styckeindelning och även kontrollera användning av komma, semikolon och kolon.

Jag tycker också att kapitel två och tre med fördel skulle byta plats. En analys av arbetet är att föredra innan man specificerar sina krav eftersom läsaren då kan ha större förståelse för kraven som nämns.

4 Slutliga kommentarer

I det stora hela är uppsatsen väl skriven. Språket håller hög kvalitet men det förekommer en del fel, främst felaktig styckeindelning.

Personligen tycker jag att det har varit ett mycket intressant ämne där man fått bra insikt i olika problem som kan uppstå – och lösningar. Man förstår också hur stort och omfattande ämne det är efter man läst uppsatsen. Det är bra att ni jämfört olika existerande verktyg för detektion av felaktig minneshantering. Man har på så sätt fått veta vilka som finns, vad som är bra med dem och hur ni tänkt plocka de bra bitarna från varje verktyg för att implementera i ert eget.