Linköpings Universitet
Institutionen för datavetenskap (IDA)
UPP-gruppen 2023-08-30

Instuderingsfrågor

Föreläsning 1

Frågor kopplade till föreläsning 1 delar av föreläsning 2. För att hitta svaren behöver du olika strategier:

- Titta igenom föreläsningsbilder
- Leta på kurshemsidan
- Skriv kod och prova kompilera och köra
- Använda förkunskaper alt. tänka till om vad som helt enkelt verkar vara en förnuftig strategi
- Sök informationen på nätet
- Skumma igenom hela dokumentet innan du börjar

Kontakta examinator om du hittar fel, otydligheter eller har andra frågor!

Kursinformation

1. På vilka ställen kan du hitta information om kursen och dina resultat?
2. Hur mycket egen tid räknar vi med att det krävs för att du ska lära dig det du behöver för att lösa laborationerna?
3. Vad är minimikraven för slutbetyg i kursen?
4. När är sista chansen inom årets kursomgång att lämna in en uppgift i kursen?
5. Hur får du högre betyg i kursen?
6. Vad gör du om det är något som inte fungerar som du förväntar i kursen?
7. Hur vet du vad du senast lämnade in via Sendlab?
8. Du frågar {en kompis, Google, ChatGPT} hur man löser felet `X`. Du får svaret "skriv Y". Om frågan och svaret bidrar positivt till din inlärning och förståelse vet du nu varför felet X uppstod och kan bedöma om "skriv Y" är en bra eller dålig lösning. För att göra exemplet konkret: Sätt X = "error: passing 'const Foo' as 'this' argument discards qualifiers [-fpermissive]" och Y = "Ta bort `const` eller kompilera med `-fpermissive`". Vet du utifrån Y varför felet uppstod? Var frågan rätt ställd och var svaret bra eller dåligt för din inlärning?
9. Vad är viktigt att tänka på när du ber om hjälp och när du ger hjälp till andra?
10. Vad är målet med laborationer och projekt: lämna in en korrekt lösning eller kunskaperna du behöver inhämta för att kunna lämna in en korrekt lösning?
11. Du märker att din labkompis löser problem nästan innan du ens märkt att det är ett problem och ännu mindre varför. Det går fort när hen skriver kod och du får kämpa för att hänga med och tänker ibland att det där får jag fundera mer på hemma. Din labkompis kanske känns otålig och ibland nästan irriterad när du ställer frågor och när det är din tur att skriva kod. Och hen vill göra extrauppgifter medan du tycker det är nog med grunduppgiften. Vad ska du göra i någon av dessa situationer?

12. Du märker att din labkompis sällan bidrar till problemlösningen och när hen gör det är det ofta enklare påpekanden om stavfel, saknade semikolon och liknande. Du får ofta stanna upp och förklara. Kanske har din labkompis även svårt komma i tid eller ens komma alls till labpass. Hen känns inte riktigt engagerad. Inför redovisningar måste ni gå igenom all kod tillsammans så du kan förklara hur du gjort. Vad ska du göra i någon av dessa situationer?

Kodstil

1. Hur påverkar olika indenteringsdjup kodens betydelse för C++ kompilator?
2. Hur påverkar olika indenteringsdjup kodens betydelse för en van programmerare?
3. Spelar det någon roll för kompilatorn om en variabel heter `average_salary` eller `as`?
4. Du träffar på variabelnamnet `as` när du ombeds felsöka koden en tidigare anställd skrivit. Hur tar du reda på vad variabeln används till?
5. Varför är variabelnamn (och datatyper och andra taggningar och formateringar av koden) viktigt trots att kompilatorn inte bryr sig vad du väljer?
6. Hur avslutas en komplett sats i C++?
7. Vilka alternativa benämningar finns för {klammerparenteser}?
8. Hur ska klammerparenteser placeras i koden för att följa kursens enkla tydliga stil?
9. När bör du använda “using namespace std” och när är det bättre att använda prefixet “std::”?

Utmatning och manipulatorer

1. Vad heter utströmmen som skriver ut i terminalfönstret?
2. Hur ser utmatningsoperatorn ut?
3. Vad menas med “utmatningsbuffer”?
4. Vad är skillnaden på `'\n'` och `std::endl`?
5. Vilken inkludering behövs för att använda utmatningsmanipulatorn `std::setw`?
6. Vad behövs för att ett månadsnummer i en heltalsvariabel ska skrivas ut med två siffror (7 blir 07, 12 blir 12)?
7. Hur gör du en innehållsförteckning med “Kapitel 1.....12” där både rubriken och sidnumret kan variera i antal tecken, men där helheten alltid är lika många tecken utfyllt med punkter?

Inmatning

1. Vilken inkludering behövs för läsa in från `std::cin`?
2. Vad menas med “inmatnigsbuffer”?
3. I inmatningsbuffern finns ’`3.14`’. Vad finns kvar efter inläsning av en `int`?
4. I inmatningsbuffern finns ’`12 34f 56`’. Vad finns kvar efter inläsning av tre `int`?
5. I inmatningsbuffern finns ’`"hej på dig"`’. Vad finns kvar efter inläsning av en `std::string`?

6. Inläsningen av ett tal har just misslyckats. Vad behöver du göra innan du kan läsa in fler värden?
7. Inmatningsbuffern innehåller '`1234567890`' när satsen `cin.ignoer(20, ':')` körs. Vad händer?
8. Inmatningsbuffern innehåller '`1234567890`' när satsen `getline(cin, str, '7')` körs. Vad kommer finnas i `str` och vad finns kvar i inmatningsbuffern?

Variabler och datatyper

1. Vilka tre uppgifter behövs för att skapa en variabel?
2. Vilken datatyp har uttrycket `4711`?
3. Vilken datatyp har uttrycket `3.14f`?
4. Vilken datatyp har uttrycket `9/4`?
5. Vilken datatyp har uttrycket `'hejsan'`?
6. Vilken datatyp har uttrycket `"C"`?
7. Vilken datatyp har uttrycket `'\n'`?
8. Vad innebär det att C++ är ett typat språk?
9. Vilka slags fel och misstag hanteras bättre av ett typat språk än ett otypat?

Kompilering

1. Vilken rad är den första som körs i ett C++-program?
2. Vad innebär det att C++ är ett kompilerat språk?
3. Kompilatorn ger en lista på flera "error", vad ska du göra?
4. Kompilatorn ger en lista på flera "warning", vad ska du göra?
5. Vad får du för felmeddelande om du glömmer en relevant `#include` och/eller `std::`?
6. Kompilatorn skriver "undefined reference to", vad betyder det?
7. Vad betyder det att kompileringen inte ger någon utdata, bara en ny terminalprompt?
8. Hur startar du ett nykompilerat program?
9. Hur kontrollerar du när en fil senast ändrades?
10. Vilka kompileringsflaggor ska du använda i kursen?
11. Vad är ett "alias" i terminalen?
12. Vilken rad avslutar funktionen main (och därmed hela programmet)?

Verktyg

1. Emacs, Visual Studio Code och Vim är vanliga att använda för att skriva program. Vilket verktyg är effektivast för att redigera kod?
2. Du hamnar i en situation där du inser att arbetsprocessen är omständlig, jobbig och/eller tråkig, t.ex. att du behöver gå igenom all din kod och göra små repetitiva snarlika ändringar på många ställen. Vad gör du?
3. Vi antar att du kommer ägna 80 timmar åt att redigera kod under kursens gång. Vi antar att det finns ett kortkommando som gör att du kan spara 5 sekunder på en uppgift du gör var 5:e minut. Hur mycket tid kan du spendera på att lära dig kortkommandot innan lärotiden överstiger tidsvinsten?

Villkor

1. Du inser att du just skrivit din if-sats med Python-syntax. Vad behöver du justera för att få C++-syntax?
2. Vad testar if-sats-villkoret `( cin >> n )`?
3. Du skapar en variabel när du kommit in i "sant"-delen av en if-sats. När du kommit ur if-satsen får du kompileringsfel när du försöker använda variabeln. Varför?
4. Hur många "else if" går det att hänga på en if-sats?
5. Hur många "else" går det att hänga på en if-sats?
6. Du upptäcker att du för att lösa en uppgift behöver skriva flera if-satser för att lösa problemet, och att det kommer att behövas otäckt många jämförelser när problemet ökar i omfattning. (T.ex. behövs 1 jämförelse för att sortera 2 tal, 3 jämförelser för att sortera 3 tal, 22 jämförelser för 10 tal och 525 jämförelser för att klara 100 tal...) Vad gör du?

Svar - Föreläsning 1 (och lite 2)

Kursinformation

1. Kurshemsidan och Webreg
2. 3.5hp * 40h/1.5hp - 40h schemalagd tid. Dvs ca 53h.
3. G på alla laborationer, G på projekt och 241p i OpenDSA.
4. 15/12
5. Samla 20 bonuspoäng för betyg 4 och 40 för betyg 5.
6. Kontakta kursledningen direkt!
7. Det vet du inte om du inte själv sparat undan koden du skickade in.
8. Nej, svaret Y säger inget om varför. Frågan var fel ställd och svaret därefter. Risken är att jag tror jag löst problemet och jag tror jag lärt mig något eftersom båda alternativen i Y får koden att kompilera. Men i själva verket har jag inte förstått. När jag förstått problemet förstår jag även att rätt lösning är att *lägga till* const. Detta illusterar en vanlig fälla med inlärning: den att det är lätt att det känns som att jag lär mig när jag läser en lösning och ser att den fungerar.
9. Ställ dig hela tiden frågan "varför" och applicera frågan "varför" rekursivt på svaret när du ber om hjälp. När du ger hjälp, hjälp frågeställaren komma fram till vilka "varför"-frågor hen bör ställa sig för att komma till förståelse. Ställ frågor där svaren sammantaget leder till lösning. Ge aldrig raka svaret "gör så här" - då är det ju du som löst problemet och inte frågeställaren.
10. Vi hoppas du är ute efter kunskaperna. Även tankesättet "jag har ju lämnat in en fungerade lösning alltså borde jag bli godkänd" är en vanlig missuppfattning. Det behöver inte finnas någon koppling alls mellan en fungerande lösning och de kunskaper du får poäng för att inhämta i kursen.
11. Det verkar som att ni inte är på samma nivå och det kan finnas många anledningar till det. Den bästa lösningen för båda är troligen att byta labkompis. Det är något ni måste prata om sinsemellan och ta hjälp av kursens personal. Det här är känsligt och det är lätt att det blir fel när ni pratar. Ett sätt att prata om det är att säga "När du gör X så känner jag Y". Prata alltså om vad *du* känner och hur *du* påverkas och uppmuntra din kompis att prata om vad hen känner och hur hen påverkas. Försök *inte* förklara på vilka sätt den andre är "fel" eller hur den andre borde göra annorlunda (vilket blir samma sak som att säga hen är "fel"). Ingen av er känner till hela bakgrunden till varför den andre känner och agerar på ett visst sätt och vilka möjligheter hen har att ändra beteende. Var ödmjuk och försök se inåt på vad du själv gör och känner och hur det påverkar andra. Du kan ändra på hur du agerar, men du kan inte ändra på andra.
    Exempel: När du kommer sent till labpasset kan jag antingen vänta på dig eller sätta igång på egen hand. Om jag väntar blir jag sur och irriterad och jag fattar att jag inte är rolig att vara med då, så jag sätter hellre igång på egen hand, men då känner jag mig taskig som bara kör på utan dig. Jag vet inte riktigt hur jag ska agera i den situationen, vad tycker du?
    Exempel: När du har löst halva labben redan innan vi träffas för att jobba på den känner jag mig utanför och som att jag inte har något att bidra med. Dessutom får jag svårt att hänga med på resten när jag inte har koll på det du redan gjort. Och om jag ber dig gå igenom det du gjort känns det som att vi spiller tid och att jag håller dig tillbaka.

12. Se föregående.

Kodstil

1. Inte alls.
2. En van programmerare tolkar koden utifrån indentering även när måsvingar säger något annat. Felaktig indentering kan bli gravt missvisande och leda till svårfångade fel.
3. Nej.
4. Du studerar mödosamt alla satser som använder variabeln för att lista ut vad koden egentligen gör med den.
5. Tydlig namngivning och väl vald datatyp sparar dig mycket tid (så länge den stämmer överens med vad koden faktiskt gör).
6. Med semikolon.
7. https://sv.wikipedia.org/wiki/Parentes
8. På egen rad med all kod inom klamrarna indenterad ett steg djupare.
9. Så här tidigt i kursen finns ingen anledning att undvika "using namespace std". Senare måste du använda prefixet "std::" för all kod i inkluderingsfiler. Generellt kan du behöva detaljkontroll över vad som ska hämtas från "std::" om du har någon namnkonflikt. Det är inte troligt du kommer råka ut för att du använder samma namngivning som standardbiblioteket och i så fall kan du antingen byta namn eller lägga din kod i din egen namnrymd för att komma runt problemet.
10. Varför kan man inte bara inkludera allt från standardbiblioteket så det finns när man behöver det?

Utmatning och manipulatorer

1. std::cout
2. `<<`
3. Mellanlagring av utskrifter för att samla ihop mer data att skicka på en gång till utenheten.
4. `'\n'` ger en ny rad i terminalen, och `std::endl` ger även en `std::flush`.
5. Se `https://en.cppreference.com/w/cpp/io/manip/setw`, `#include <iomanip>`
6. Sätt utskriftsbredden till 2 och utfyllnadstecknet till 0.
7. Sätt utfyllnadstecknet till punkt. Bryt upp problemet i två. En utskriftsbredd för kapitelrubriker och en utskriftsbredd för sidnummer. Låt de båda summera till den önskade bredden. Antalet tecken för sidnummer kan enkelt begränsas till 5[1] vilket lämnar största möjliga bredd till kapitelrubriker.

[1] https://www.smithsonianmag.com/smart-news/longest-book-in-the-world-impossible-to-read-180980814/

Inmatning

1. Se `https://en.cppreference.com/w/cpp/io/cin`, `#include <iostream>`
2. Den mellanlagringsbuffer där användarinmatning lagras så fort användaren bekräftat med "Enter". Programmet läser all indata från buffern och stoppar för att vänta på mer när buffern är tom.
3. '`.14`'
4. '`f 56`'. Tredje inläsningen misslyckas eftersom "f" inte är ett heltal, så inget läses då.
5. '` på dig"`'.
6. `cin.clear()` och lämplig borttagning av felaktig data (`cin.ignore(..., ...)` eller inläsning till sträng)
7. Inmatningsbuffern töms på 10 tecken och programmet väntar på att ytterligare 10 tecken ska läsas in så de kan ignoreras (så länge inget `:` påträffas).
8. `str="123456"`. Inmatningsbuffern har kvar `890`.

Variabler och datatyper

1. Datatyp, namn och initialvärde.
2. `int`
3. `float`
4. `int`. Eftersom båda talen i divisionen är heltal blir det en heltalsdivision.
5. Enkelcitattecken kan bara innehålla ett tecken. Inte en sträng. Det blir kompileringsfel.
6. `std::string` (Puristen skulle säga `const char*`. För att få en faktisk `std::string` skriver man `"C"s`. Det finns situationer när skillnaden kan orsaka huvudbry. `"Hello"s` är ett säkrare skrivsätt än `"Hello"` men kräver "using namespace std::literals".)
7. `char`. Bakåtstrecket ingår inte som tecken utan varierar enbart betydelsen av tecknet efter. Detta kallas "escapesekvens" och gör det möjligt att skriva specialtecken med vanliga tecken.
8. Att alla variabler och uttryck har en datatyp och kompilatorn kontrollerar att alla operationer är väl definierade för datatypen ifråga.
9. Misstag där oavsiktlig skillnad i datatyp leder till logiska fel (koden fungerar men ger fel resultat) eller misstag där fel datatyp leder till latenta fel (som visar sig först långt senare när ett undantagsfall som testningen förbigått inträffar).

Kompilering

1. Första raden i blocket under `int main()`.
2. Att koden kontrolleras och översätts till binärkod innan programkörning kan ske. Det innebär att det arbetet bara behöver genomföras en gång för alla efterföljande programkörningar (tills koden ändras).
3. Leta upp det första längst upp i terminalen och bena ut vad det betyder.

4. Samma procedur som för “error”.
5. Prova! Det finns tre varianter. Glöm inkludera, glöm “std::” eller glöm båda. Kompilatorn ger lite olika fel och tips.
6. Det betyder att en programkomponent (oftast en funktion) saknas. Kontrollera att du har en definition med i någon av filerna som kompileras.
7. Success! Det är här du ska göra en liten segerdans och känna dig tillfälligt mycket nöjd med tillvaron.
8. `./a.out`
9. `ls -l`
10. Se kurshemsidan.
11. Det låter dig döpa om ett terminalkommando. Används oftast för att sätta ett kortare namn på ett långt kommando med många flaggor.
12. `return 0;`

Verktyg

1. Emacs för mig. Men för dig kan det vara något annat. Det beror helt och hållet på hur väl du kan ditt verktyg.
2. Du tänker “det måste finnas ett bättre sätt” och du har rätt. Många har redan råkat ut för problemet och någon har utarbetat en lösning. Du behöver bara leta upp den.
3. 80 minuter. Det finns mycket att tjäna på att identifiera uppgifter som borde kunna utföras snabbare och enklare och lägga lite tid på att lära sig hur. Speciellt om du räknar in tiden du sparar i ett större perspektiv än bara kursen (hela arbetslivet?). Ett enkelt exempel är kompilering i terminalen. Att gång på gång skriva kommandot med alla flaggor tar lång tid och behöver göras ofta. Det finns flera sätt att bli mycket effektivare med terminalkommandon. (Fråga din assistent, kursledare eller examinator!)
   Fråga dig även om det finns ett sätt att helt undvika att den mödosamma uppgiften uppstår. Det finns antagligen ett bättre sätt att strukturera som undviker problemet.

Villkor

1. Villkoret behöver omges av parenteser, kolonet ska bort och den indenterade koden ska omges av klammerparenteser.
2. Om inläsningen lyckades.
3. Variabler finns enbart inom de klammerparenteser de deklarerats i. Du behöver deklarera variabeln i samma block som if-satsen om du ska komma åt den efter.
4. Det finns ingen övre gräns.
5. Noll eller en.

6. Du tänker "det måste finnas ett bättre sätt" och du har rätt. Lägg tid på att lära dig en metod att undvika situationen och om det inte går (det går!) ett sätt att automatisera skapandet av alla villkor. Så fort något blir mödosamt finns det enligt min erfarenhet ett bättre sätt. Du måste bara ta reda på hur.

Föreläsning 2

Alternativa strömmar

1. Du vill använda strömoperatorn » för att läsa ut heltal från en sträng. Vilken typ av ström ska du använda?
2. Du har en sammansatt datatyp du vill formatera med de vanliga ström-manipulatorerna men du vill ha resultatet som en sträng i programmet, inte i terminalen. Hur går du tillväga?
3. Hur deklarerar du en ström som enbart läser från filen `data.txt`?
4. Du har en fil med tre kolumner. Varje kolumn innehåller ett heltal, men ibland är tredje kolumnen tom (en del rader har alltså bara två tal). Du är bara intresserad av data i de två första kolumnerna. Hur löser du inläsningen?

Operatorer

1. Du vill testa `a != b` men kompilatorn säger att operatorn `!=` inte finns för datatypen. Däremot finns operatorn `==`. Hur skriver du om ditt villkor så det fungerar?
2. Du vill testa `a >= b` men kompilatorn säger att operatorn `>=` inte finns för datatypen. Däremot finns operatorn `<`. Hur skriver du om ditt villkor så det fungerar?
3. Du skriver uttrycket `2^3` och väntar dig svaret 8. Istället blir svaret 1. Vad gör operatorn `^`?
4. Hur gör du för att en division med två heltal ska utföras som en flyttalsdivision?
5. Vad skiljer preinkrement från postinkrement?
6. Du har skrivit villkoret `a < b && b < c || a*b != c` men är nu osäker på i vilken ordning de olika operatorerna faktiskt utförs. Vad gör du?
7. Du har skrivit villkoret `while ( (x < 0) || !(cin >> x) )` för att läsa in `x` igen så länge x är mindre än noll eller ny inläsning misslyckats, men programmet verkar istället fastna utan att göra en ny inläsning. Vad är problemet?

Scope och funktioner

1. Du har satt `x=4` i funktionen main. Nu använder du `{` för att öppna ett nytt block(scope) där du gör en ny variabel `int x{11}`. Vad behöver du göra för att kunna komma åt variabeln x som finns i main med värdet 4?
2. Vad betyder datatypen som står först i en funktionsdeklaration?
3. Hur många typer kan en funktion returnera?

4. Går det att returnera en sammansatt typ?
5. Varför rekommenderas att använda `const&` för funktionsparametrar av sammansatt typ?
6. Du upptäcker att din funktion behöver uppdatera tre saker åt den som anropar. Hur ska du designa din funktion?
7. En funktion har designats att beräkna både division och modulus. Som bieffekt skrivs dessutom ett felmeddelande ut när nämnaren är noll. Vilka problem finns med denna design?
8. Kan en funktion ha flera `return`-satser?
9. Hur många gånger kan du returnera från en funktion?
10. Vad händer om du glömmer returnera från en funktion?
11. Du har en stor vektor i ditt program och vill att funktionen `remove_duplicates` ska modifiera din vektor direkt istället för att returnera en kopia utan dubbletterna. Hur designar du funktionen?
12. Du har en viktig vektor i ditt program som du inte vill ska råka ändras. Hur gör du det tydligt att funktionen `get_three_largest(vector<int>& v)` inte råkar sortera den vektor som skickas som argument innan de tre största värdena hämtas ut?

Vektorer

1. Hur deklarerar du en vektor som innehåller komplexa tal (datatypen std::complex)?
2. Går det att göra en vektor som kan innehålla olika datatyper på olika platser?
3. Vilken funktion används för att lägga till något sist i en vector?
4. Du har definierat en vektor `v` med 5 heltal. Nu hämtar du ut det sista heltalet med `v[5]`. Vad blir fel och vilka två ändringar bör du göra?

Upprepning

1. Du vill skriva ut en tabell från -20 till 50 i steg om .5. Vilken upprepningssats är bäst lämpad?
2. Du vill läsa in ett värde tills inläst värde är i intervallet [2-7]. Vilken upprepningssats är bäst lämpad?
3. Du vill räkna hur många gånger det går att utföra N = N / 2 innan N == 0. Varför är “while” bäst lämpad?
4. På ditt företag måste du använda ett strikt subset av C++. En av reglerna är att du enbart får använda `for`-loopen. Hur gör du om en `while` till en `for`?
5. Du vill skriva ut en lista med månadsnummer och månadsnamn på varje rad. Hur gör du?
6. Du vill kontrollera om en inmatning stämmer överens med någon av 7 olika strängar. Hur gör du?

Aggregat

1. Hur skapas en sammansatt datatyp månader? En månad har ett namn, ett nummer och en årstid.
2. Hur gör du en variabel av ovan datatyp?
3. Hur kommer du åt månadsnumret från ovan variabel?
4. Hur deklarerar du en funktion som skriver ut en månad av ovan typ?

Svar - Föreläsning 2

Alternativa strömmar

1. `istringstream`
2. Jag gör en `ostringstream oss`, skriver ut till `oss`, och plockar fram resultatet med `oss.str()`
3. `ifstream ifs{"data.txt"}`
4. Använd mönstret från föreläsningsbild 10 (forts från bild 9).

Operatorer

1. `!(a == b)`
2. `!(a < b)`
3. Den gör bitvis XOR. Bitoperatorerna ingår inte i det du behöver lära dig, men är lätta att förväxla med annat.
4. Konvertera antingen täljaren eller nämnaren till ett flyttal före divisionen utförs.
5. Postinkrement sparar undan och ger tillbaka originalvärdet.
6. Studera `https://en.cppreference.com/w/cpp/language/operator_precedence` och lägg till parenteser i ditt villkor tills det är uppenbart i vilken ordning allt sker.
7. Inläsningen kommer aldrig att utföras när x är mindre än noll eftersom `||` är kortslutande när vänstersidan räcker för att avgöra helheten.

Scope och funktioner

1. Antingen avslutar du blocket(scopet) med `}`. Väl tillbaka i scopet av main kommer du åt dess x. Eller så ser du till att inte använda samma variabelnamn.
2. Det anger typen för hela funktionens resultat.
3. En.
4. Ja.
5. För att både undvika dyr kopiering och göra det tydligt och kontrollerat att argumentet inte råkar ändras.

6. Det är en svår fråga som avgörs från fall till fall. Prata med någon av kursens personal. Du kan använda referensparametrar för att "returnera" saker via. Du kan skapa en sammansatt datatyp för det som returneras. Du kan dela upp funktionen i tre olika, en för respektive del. Om något är en felkod kan du använda undantag till den.
7. En funktion ska ha ett syfte och inga bieffekter. Dela upp den i två och hantera felet med undantag. Undantag kommer senare i kursen.
8. Ja. Så många du vill. De bör vara tydligt strukturerade så det blir uppenbart när funktionen avbryts i förtid.
9. En gång.
10. Om du har rätt kompileringsflaggor får d en varning för det felet. Annars är det odefinierat, vilket betyder att returvärdet kan bli vad som helst eller till och med krascha programmet.
11. `remove_duplicates(vector<int>& v)`
12. `get_three_largest(vector<int> const& v)`

Vektorer

1. `std::vector<std::complex> v`
2. Nej. Vi ser senare i kursen hur det går att komma runt den begränsningen.
3. `push_back`
4. Giltiga index är 0-4. Index 5 finns inte. Operatorn `[]` har inte indexkontroll. Använd `v.at(4)` för att lösa båda problemen.

Upprepning

1. `for`
2. `do while`
3. Upprepningen behöver inte utföras alls om N är noll. `while` är förtestad och utför upprepningen noll eller flera gånger.
4. `while ( villkor ) { satser; }` blir `for ( ; villkor; ) { satser; }`
5. Jag lägger in månadsnamnen i en vektor och itererar över den. Då har jag nytta av vektorn i andra sammanhang när månadsnamnen behövs.
6. Jag lägger in de 7 strängarna i en vektor och itererar över den. Det gör tillägg av fler strängar att testa mycket enkelt.

Aggregat

1. `struct Month { std::string name; int number; string season; };`
2. `Month second{ "Feb", 2, "Winter"};`
3. `second.number`
4. `void print(Month const& m);`