CHALMERS

Institutionen för Data och informationsteknik.
Roger Johansson
VT 2022

Maskinorienterad programmering

Inledande uppgifter där du:

- Bekantar dig med utvecklingsmiljön
- Lär dig redigera, kompilera och testa ett enkelt C-program

Uppgifterna löses med *CodeLite/GCC* för värddatorn.

1. Inledningsvis skapar du ett *arbetsutrymme* (Workspace), och lägger till ett *projekt* i arbetsutrymmet.
2. Därefter *bygger* du ett enkelt `Hello World` –program.
3. Du provar att *köra* ditt program.
4. Du övergår därefter till att undersöka hur man kan generera och skriva ut en följd av *Fibonaccital*. Fibonaccital är tal som ingår i en heltalsföljd, Fibonaccis talföljd, där varje tal är summan av de två föregående Fibonaccitalen; de två första talen är 0 och 1.

Fler lämpliga programexempel av varierande svårighetsgrad kan du hitta här:

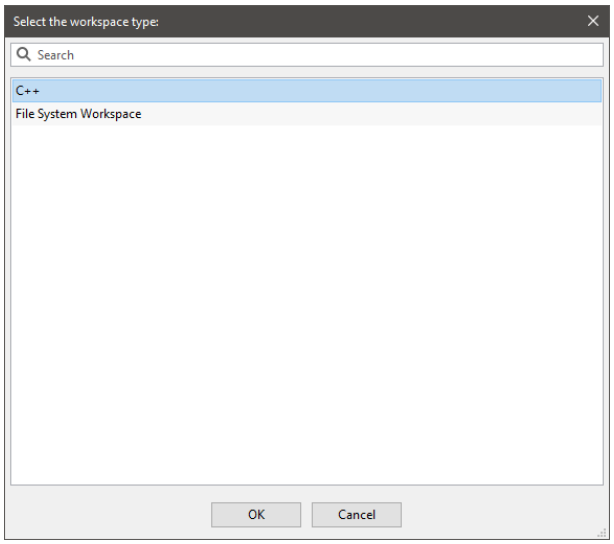
`https://www.programmingsimplified.com/c-program-examples`

Skapa ett nytt arbetsutrymme

Skapa ett nytt arbetsutrymme, välj från menyn:

Workspace | New Workspace

Du får nu välja vilken typ av applikation du avser att arbeta med, välj här C++.



Beroende på din Codelite installation kan det finnas fler alternativ. Vi behandlar inte dessa här.

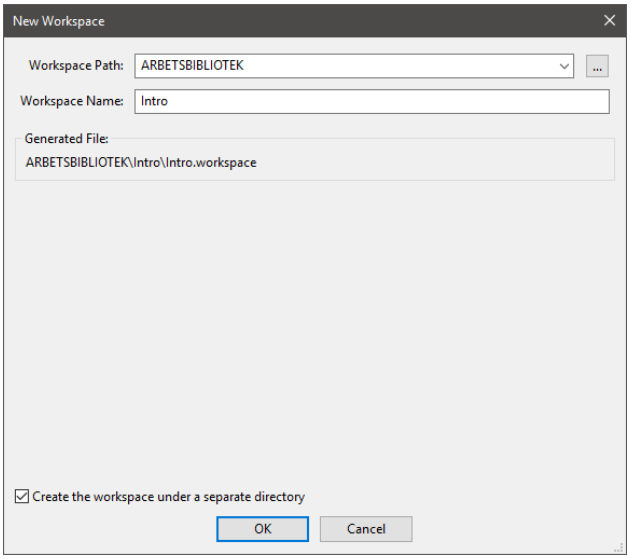
Klicka nu OK för att gå vidare.

Tänk redan från början igenom hur du vill organisera dina filer. Ett enkelt sätt är att välja sökvägen (Workspace Path) som en *rotkatalog* och därefter lägga till projekt och filer under denna rotkatalog. Det är dock inte nödvändigt men man bör tänka på att praktiskt taget alla filer anges med relativa sökvägar vilket kan ha betydelse om man senare försöker flytta runt projekt eller filer.

Undvik att sprida ut arbetsutrymmet över flera olika diskar, det brukar ofta leda till olika typer av svårförståeliga felsysmptom.

Undvik också blanktecken och nationella tecken, exvis å,ä,ö i sökvägar och filnamn eftersom detta kan skapa problem med olika verktyg som används av *CodeLite*.

Välj en lämplig plats (*Workspace* Path) och ett lämpligt namn (Workspace Name) för ett första arbetsutrymme.



Klicka därefter på OK för att skapa arbetsutrymmet. Namnet får automatiskt filnamnstillägget `.workspace`

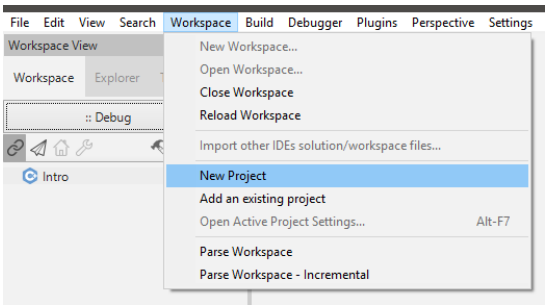
Att tänka på: projekt kan ingå i flera arbetsutrymmen. Du kan därför senare skapa ett nytt arbetsutrymme där du lägger till tidigare projekt. På så sätt kan samma projekt nås från flera olika arbetsutrymmen.

Skapa ett nytt projekt

Du kan skapa ett nytt projekt på flera olika sätt:

välj från huvudmenyn:

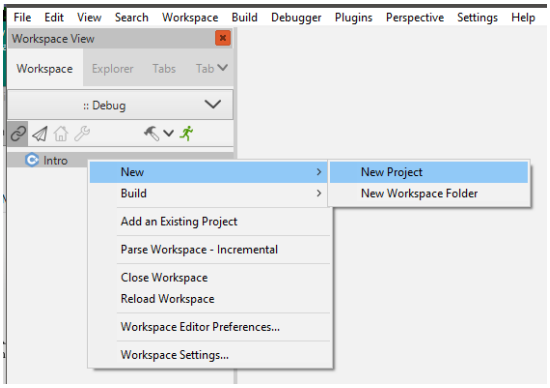
Workspace | New Project



eller

Högerklicka på det nya arbetsutrymmets ikon i *Workspace*-fliken, välj:

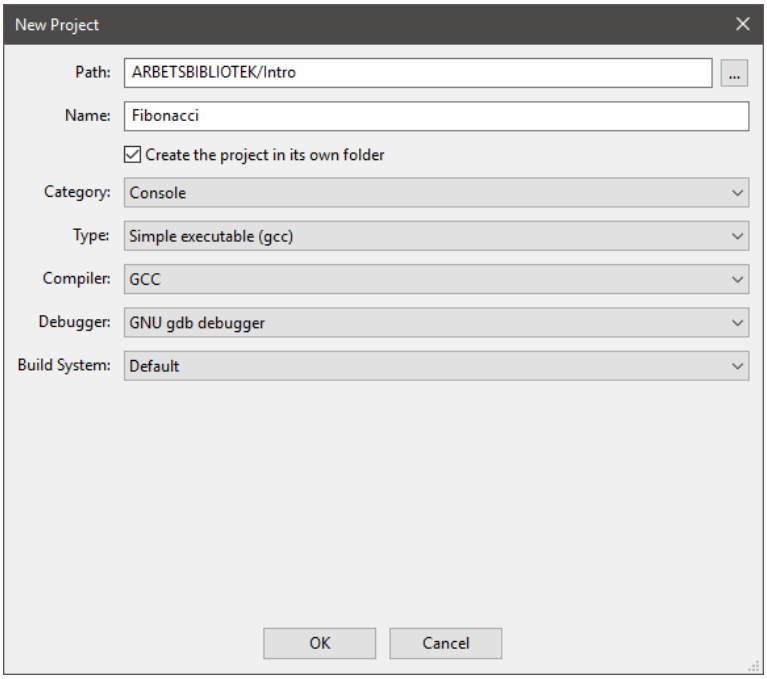
New | New Project



New Project-dialogen öppnas:

Kontrollera att inställningarna anger rätt typ av projekt och kompilator, jämför med figuren.

Obs: under MacOS kan du bara välja debugger LLDB här.



Välj

Console | Simple executable (gcc)

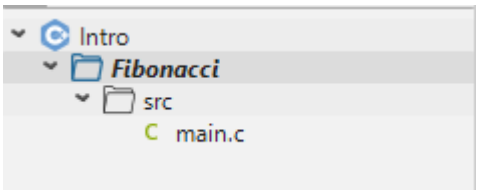
Ge projektet ett namn, som exempel

`Fibonacci`

Klicka OK.

Projektet `Fibonacci` skapas...

...expandera dess ikon i Workspace-fliken:



Här har du nu fått en ny fil, main.c att börja ditt projekt med.

```
#include <stdio.h>
int main(int argc, char **argv)
{
	printf("hello world\n");
	return 0;
}
```

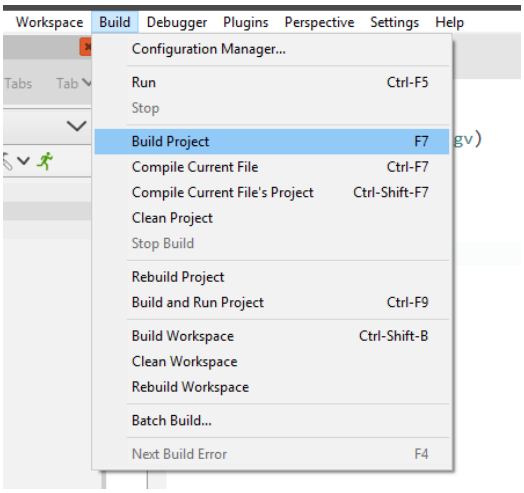
Vi börjar med att prova denna innan vi övergår till Fibonacci...

Kompilera och länka

Du kan "bygga" projektet, dvs. skapa en exekverbar fil, på flera olika sätt:

välj från huvudmenyn:

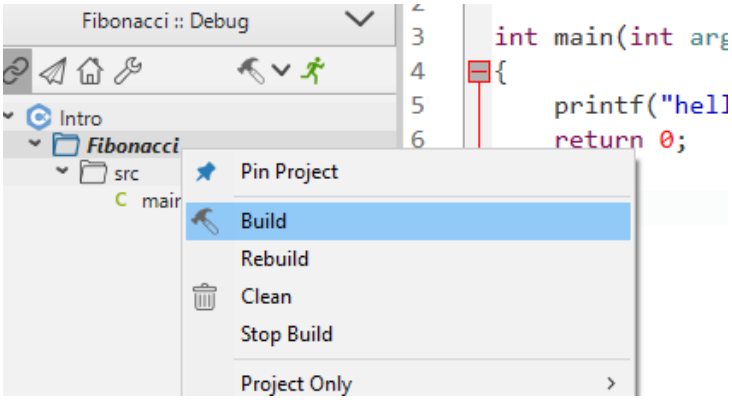
Build | Build Project



eller med snabbvalstangent (F7)

eller genom att högerklicka på projektet och välja:

Build



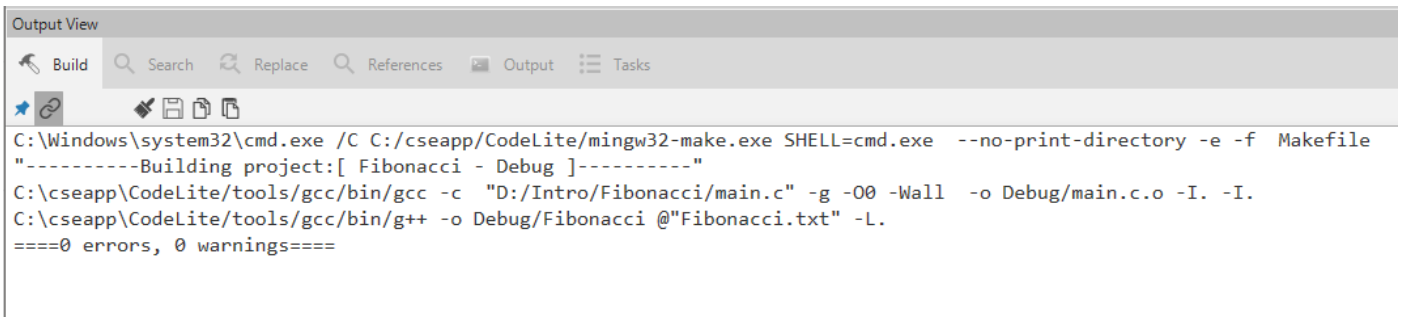
Alternativet Build används alltså för att kompilera nödvändiga filer i projektet till objektfiler (.o) för att avslutningsvis länka samman objektfiler med standardbibliotek till en exekverbar fil.

Via denna popup-meny kan du göra flera olika saker, som att lägga till ytterligare filer i projektet, strukturera vyn med hjälp av virtuella foldrar (påverkar ej filernas placering på disken...) etc.

Alternativet Clean tar bort samtliga objektfiler och Rebuild kompilerar om samtliga källtextfiler innan objektfilerna länkas samman.

Du kan själv undersöka andra operationer som kan utföras via denna meny.

Vid Build aktiveras också motsvarande meddelandefönster längst ned, och visar resultatet:



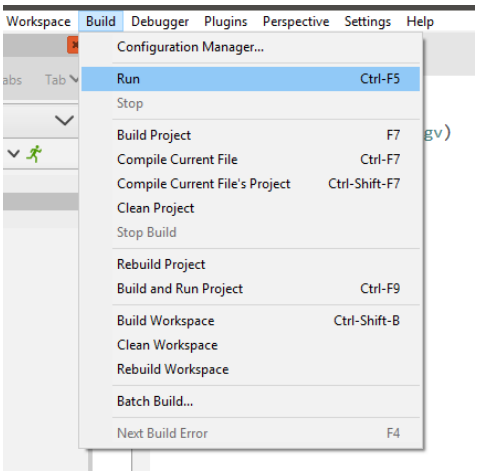
Här från Windows, det ser något annorlunda ut under Linux eller MacOS.

Provkör programmet

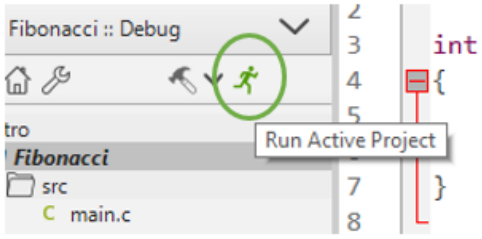
Du kan starta programmet på flera olika sätt, från huvudmenyn:

Build | Run

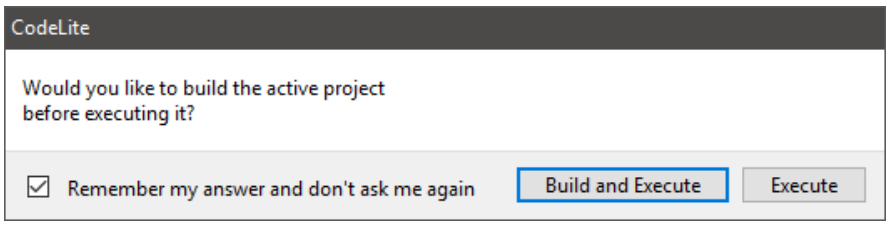
eller med snabbval (Ctrl-F5)



eller via ikonen Run Active Project

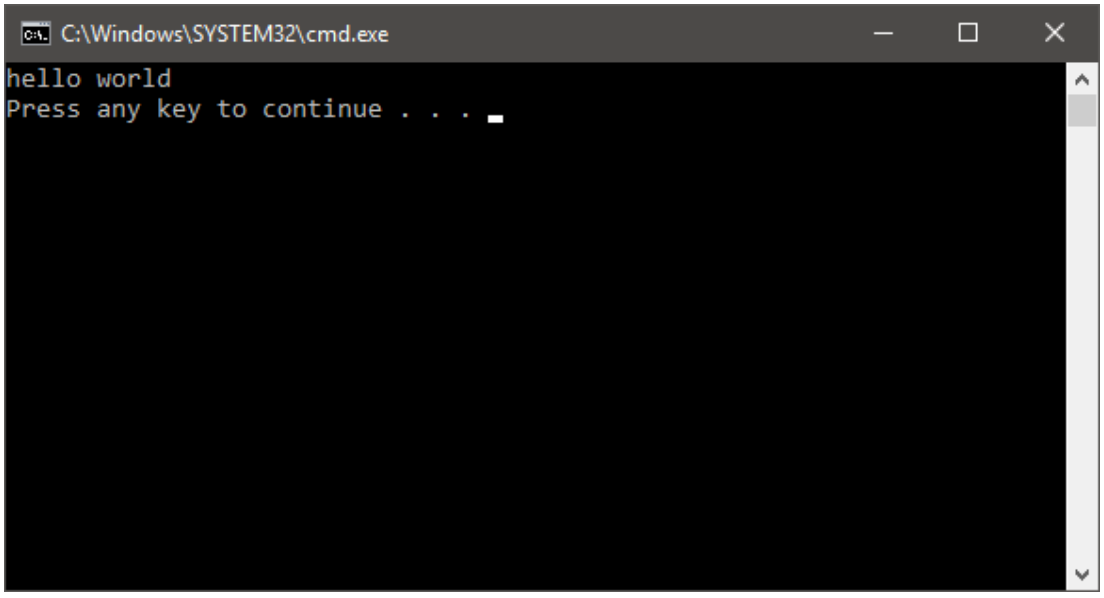


Du får nu frågan:



Kryssa i "kom-ihåg"-boxen så slipper du svara på detta varje gång. Välj sedan Execute.

Ett konsolfönster skapas automatiskt (vi valde ju denna projekttyp) och programmets utskrift skickas till detta fönster...



Test, felsökning och felavhjälpning

Vi fortsätter nu med kod hämtad från:

`http://www.programmingsimplified.com/c-program-generate-fibonacci-series`

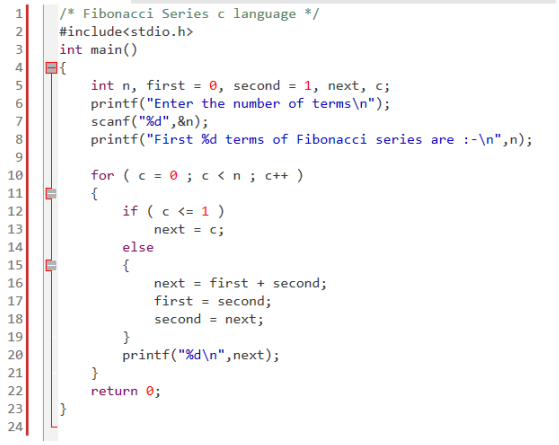
Du kan också kopiera koden härifrån:

Ersätt `Hello World`-programmet i filen `main.c` med Fibonacci...

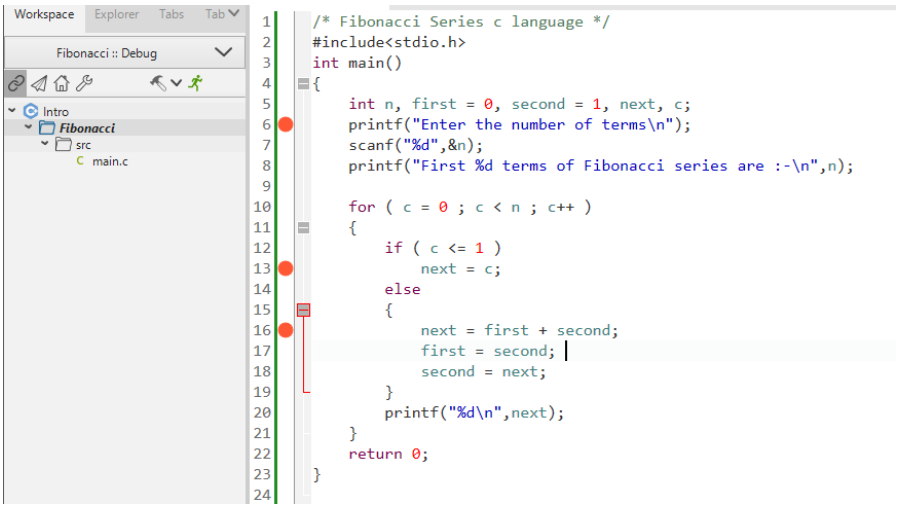
```
/* Fibonacci Series c language */
#include<stdio.h>
int main()
{
  int n, first = 0, second = 1, next, c;
  printf("Enter the number of terms\n");
  scanf("%d",&n);
  printf("First %d terms of Fibonacci series are :-\n",n);

  for ( c = 0 ; c < n ; c++ )
  {
    if ( c <= 1 )
      next = c;
    else
    {
      next = first + second;
      first = second;
      second = next;
    }
    printf("%d\n",next);
  }
  return 0;
}
```

Att sätta en brytpunkt:

Placera markören i något av fälten med vit bakgrund i texteditorns vänstra kant och högerklicka, välj AddBreakpoint för att sätta ut en brytpunkt på raden.



Följande bild visar hur vi satt brytpunkter på tre strategiska ställen i programmet:

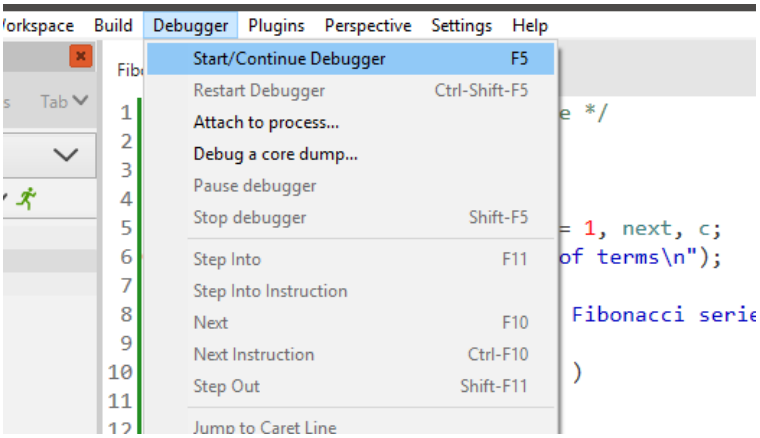


Kompilera och länka projektet Fibonacci. Om du får felmeddelanden, rätta till och kompilera igen.

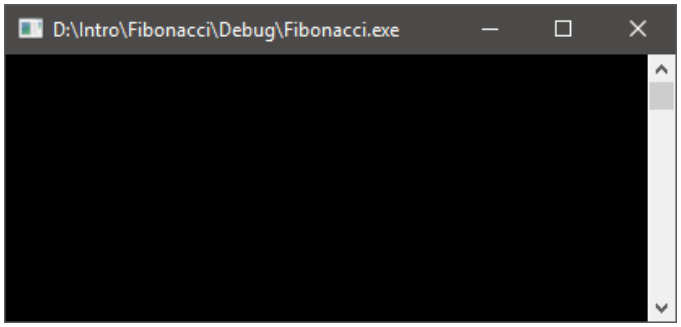
Starta sedan debuggern från huvudmenyn:

Debugger | Start/Continue Debugger

eller med snabbval (F5)



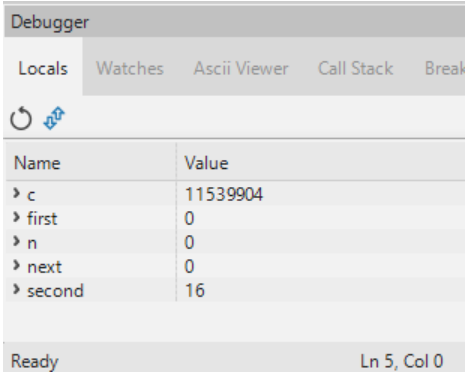
Ett *konsollfönster* skapas, det är detta fönstret vi använder för in- och utmatning med vårt testprogram



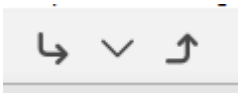
Exekveringspunkten, dvs. den rad i programmet, som står i tur att utföras, märks upp, som här med grön bakgrund, eller en grön pil i vänster marginal. Här visas nu programmets första exekverbara sats:

```
2   #include<stdio.h>
3   int main()
4   {
5       int n, first = 0, second = 1, next, c;
6       printf("Enter the number of terms\n");
7       scanf("%d",&n);
8       printf("First %d terms of Fibonacci series are :-\n",n);
9
10      for ( c = 0 ; c < n ; c++ )
11      {
12          if ( c <= 1 )
13              next = c;
14          else
15          {
16              next = first + second;
17              first = second;
```

Observera hur de lokala variablerna (Locals) har slumpmässiga värden det kan vara helt andra värden på din skärm, variablerna är än så länge oinitierade, detta görs av programmets första sats.



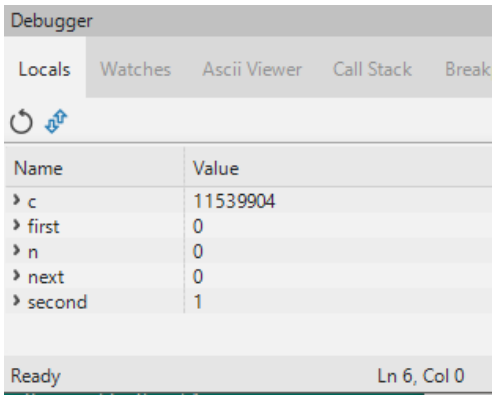
De tre ikonerna StepIn, Next och StepOut används för kontrollera stegvis exekvering av programmet:att utföra delar av programmet.



- StepIn: Om nästa exekveringspunkt är en funktion så följ programmet in i funktionen. (Använd Next om du vill utföra hela funktionen).
- Next: Utför hela exekveringspunktens rad.
- StepOut, utför hela funktionen

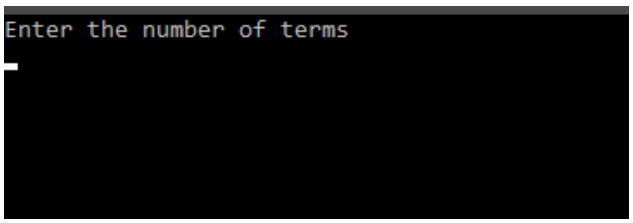
Här är det lämpligt att stega med Next:

Den första raden, dvs. tilldelningarna av `first` och `second` utförs, exekveringspunkten flyttas till nästa rad. Observera hur fönstret med variabler uppdateras.

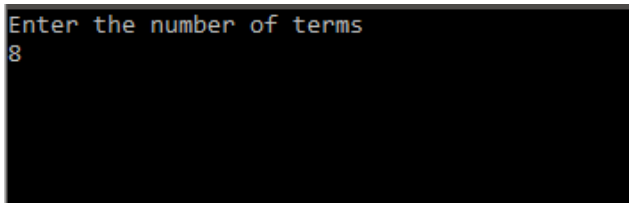


Utför nu hela `printf`-satsen, Växla till konsollfönstret och observera programmets utskrift.

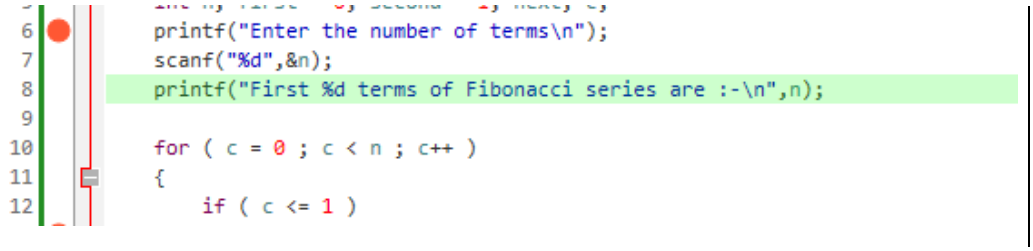
Exekveringspunkten flyttas till nästa rad, som är en inmatningssats (`scanf`).

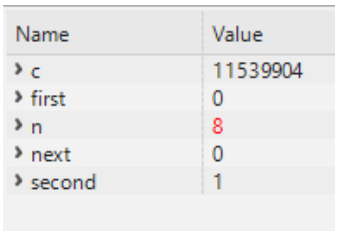


Klicka nu ytterligare en gång på Next-ikonen, för att utföra raden med `scanf`-satsen, observera att exekveringspunktens markering försvinner (programmet exekveras), det beror på att programmet väntar på inmatning, ge ett heltal, exempelvis 8 och tryck därefter <Enter> för att avsluta inmatningen.



Debuggern tar nu tillbaks kontrollen, exekveringspunkten placeras på nästa `printf`-sats. Observera också hur variabeln `n`, som tilldelas i `scanf`, uppdateras.



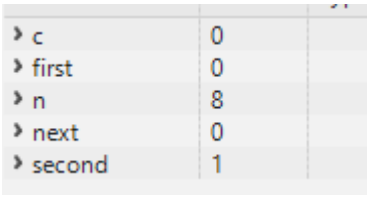


Som nästa steg kan du prova att exekvera programmet till nästa brytpunkt, klicka på den gröna startikonen (Continue)

Programmet stoppas vid nästa brytpunkt...

```
10      for ( c = 0 ; c < n ; c++ )
11      {
12          if ( c <= 1 )
13              next = c;
14          else
15          {
16              next = first + second;
17              first = second;
18              second = next;
19          }
```

Inspektera de lokala variablernas värden



Ta bort brytpunkterna i programmet genom att vänsterklicka på den orange markeringen (markeringen försvinner)

Sätt ut en ny brytpunkt på programmets sista rad (`return 0`)

```
20          printf("%d\n",next);
21      }
22      return 0;
23  }
24
```

Exekvera programmet (Continue) till sista brytpunkten:

```
20          printf("%d\n",next);
21      }
22      return 0;
23  }
24
```

Observera texten som matats ut i konsollfönstret av programmet:

```
First 8 terms of Fibonacci series are :-
0
1
1
2
3
5
8
13
```

Avsluta debuggern genom att klicka på den röda ikonen: