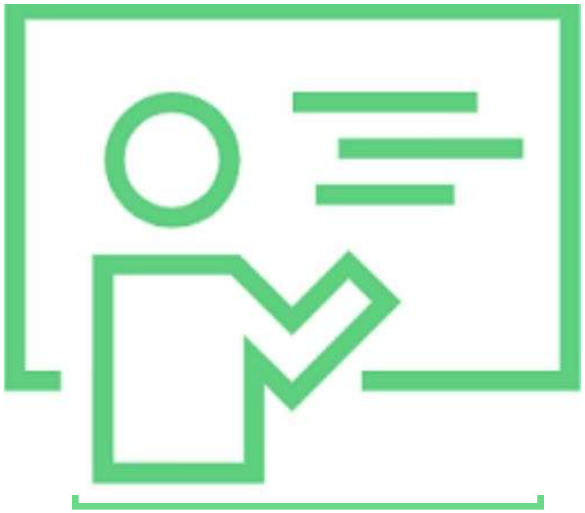


Funktioner #2

Stefan Holmberg, Systementor AB

Funktioner

Grundläggande byggsten i C

Tillåter återanvändning av kod

- Ohanterlig om man skrev hela programmet i *main()*

Ökar läsbarhet

Minskar fel

När ska jag använda funktioner?

- Tumregel: om koden gör samma sak fler gånger men med olika inputs; skapa en funktion av det
- Tumregel: om koden tar upp mer än en hel skärm, bryta ner den i fler funktioner

Mer

Funktioner *deklareras* i C med:

- Funktionsnamn
- En returtyp
  - Kan vara *void*
- Ett antal parametrar

Och *definieras* i C med:

- Ett kodblock
  - Som slutar med *return* om det finns en icke *void* returtyp

Ännu mer

Funktioner som du vill göra tillgänglig till andra .c filer kan

deklareras i en .h (header) fil och *definieras* i en .c (källkod) fil

Funktioner som du inte vill göra tillgänglig i andra filer kan

deklareras och *definieras* i en och samma sats i en .c fil

Deklarationen av en funktion måste komma före användning av den, men definitionen kan komma senare eller i en annan fil

Anropa en funktion

funktionsnamn(*arg1*, *arg2*, ...);

Eller

resultat = *funktionsnamn*(*arg1*, *arg2*, ...);

```c
int double( int x ) {
    return x * 2;
}

int main() {
   int n, result;
   scanf("%d", &n );
   result = double( n );
    printf("%d", result);

   return 0;
}
```

- Definieras och deklare
  i samma sats
- Anropas senare
- Vad gör *double()*?

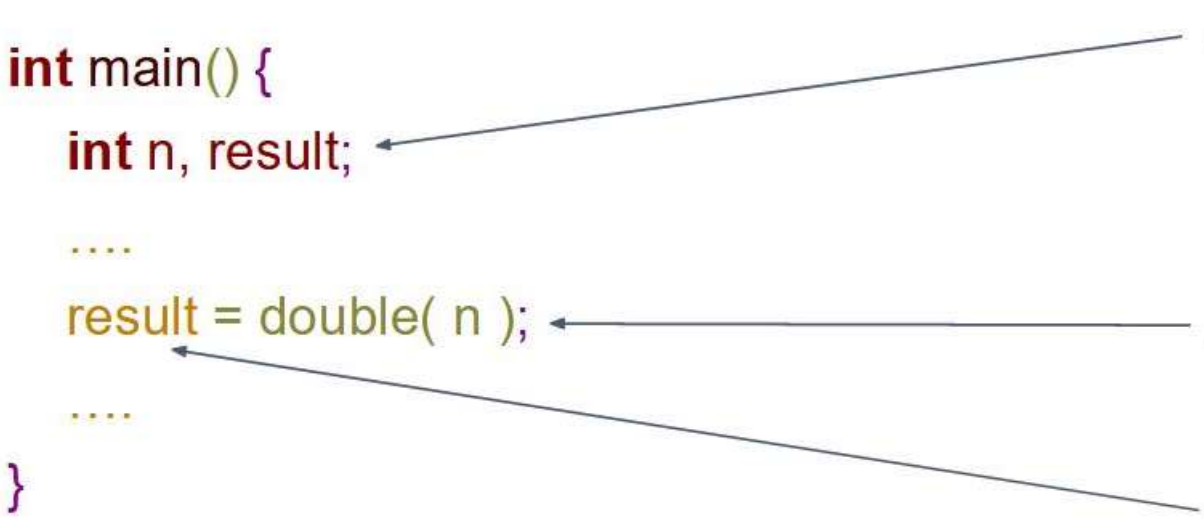
Deklaration

```
int double( int x );

int main() {
    int n, result;
    scanf("%d", &n );
    result = double( n );
     printf("%d", result);

    return 0;
}
int double( int x ) {
    return x * 2;
}
```

Begrepp



- Variabler för att hålla parametern och resul av funktionen
- Anropa *double* med parametern *n*
- Och lagra resultatet i *result*

Anropa flera gånger

```
int main() {
    int n, m, result;
    ....
    m = double( n );
    result = double( m );
    ....
}
```

- Kan anropa fler gånger
- Med olika parameter
- Och till olika variabler för resultatet
- Vad gör denna kod?

Anropa flera gånger MER

```
int main() {
    int n, result;
    ....
    result = double( double( n ) );
    ....
}
```

- Resultatet måste inte la
  en variabel
- Det går att använda
  resultatet som om att d
  vore en variabel

Värden...stack...pekare

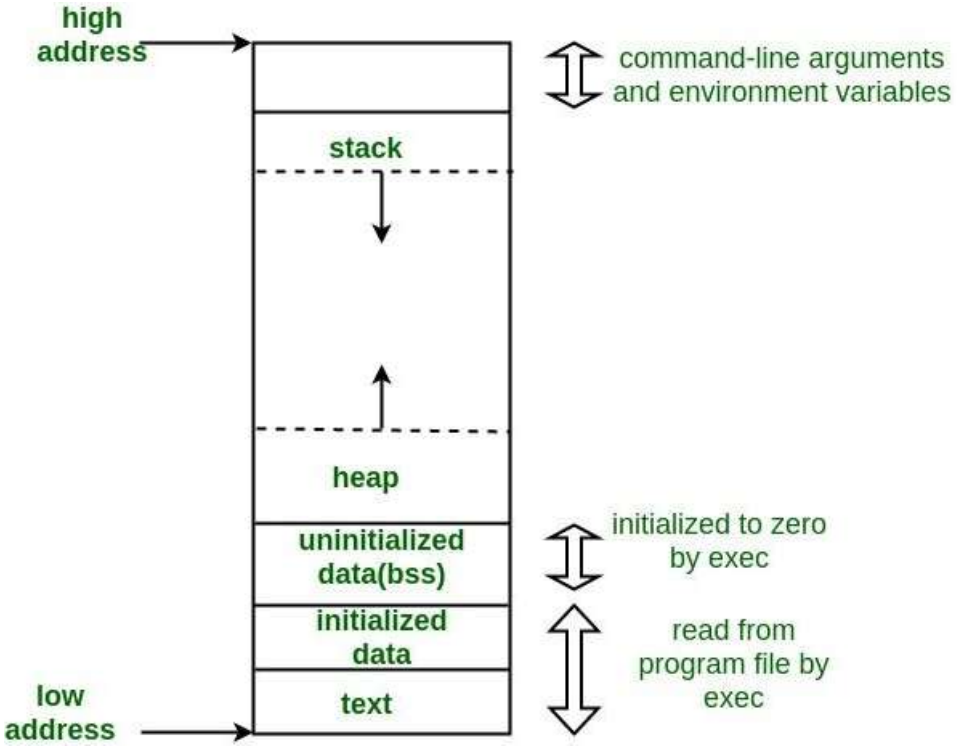
```
void addera(int c){
    int t = 123;
    char whatever[10] = "blabla";
    c = c + 12;
    printf("%d\n", c);
}

void main(){
    int b = 99;
    int c = 11;
    char namn[15] = "Stefan";
    printf("%d\n", c);
    addera(c);
    printf("%d\n", c);
}
```

Vad skriver den ut?

1. 11 23 11
2. 11 23 23
3. 11 11 11

Värden...stack...pekare



Värden...stack...pekare

```
void addera(int c){
    int t = 123;
    char whatever[10] = "blabla";
    c = c + 12;
    printf("%d\n", c);
}

void main(){
    int b = 99;
    int c = 11;
    char namn[15] = "Stefan";
    printf("%d\n", c);
    addera(c);
    printf("%d\n", c);
}
```

Vad skriver den ut?

1. 11 23 11
2. 11 23 23
3. 11 11 11

DEMOS

1. Enkel variabel → funktion
ALLTID COPY BY VALUE

2. Funktions variabel (SCOPE)

3. Return

4. Ändra på variabel i FUNKTION

Hello PEKARE (adress)