Call Stack

>> Wenn Sie eine Funktion aufrufen, reserviert das System für diese Funktion Speicherplatz im Stack-Speichersegment.

Diese Speicherbereiche nennen wir häufig **Stack Frames** oder *Funktionsrahmen*.

Es können mehrere Stack Frames gleichzeitig im Speicher existieren.

Wenn main() die Funktion move() aufruft, welche dann direction() aufruft, gibt es für die drei Funktionen je einen Stack Frame.

Der Call Stack

Diese Frames liegen im Stack-Segment und werden in einer Stack-Datenstruktur gespeichert. Der Frame der zuletzt aufgerufenen Funktion liegt immer „oben“ auf dem Stack.

Wenn eine neue Funktion aufgerufen wird, wird ein neuer Frame auf den Stack **gepusht** – er wird zum aktiven Frame.

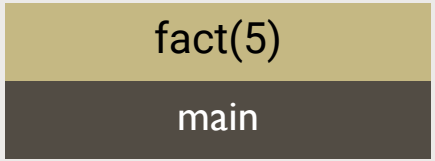
Wenn eine Funktion ihre Arbeit beendet, wird ihr Frame vom Stack **gepoppt**, und der Frame direkt darunter wird zum neuen aktiven Frame an der Spitze des Stacks.

Die vorherige Funktion setzt ihre Arbeit dann genau dort fort, wo sie durch den Funktionsaufruf unterbrochen wurde.

```c
int fact(int n)
{
    if (n == 1)
        return 1;
    else
        return n * fact(n-1);
}


int main(void)
{
    printf("%i\n", fact(5));
}
```

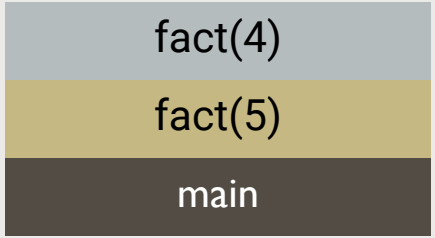
Call Stack

```c
int fact(int n)
{
    if (n == 1)
        return 1;
    else
        return n * fact(n-1);
}


int main(void)
{
    printf("%i\n", fact(5));
}
```



Call Stack

```c
int fact(int n)
{
    if (n == 1)
        return 1;
    else
        return n * fact(n-1);
}


int main(void)
{
    printf("%i\n", fact(5));
}
```

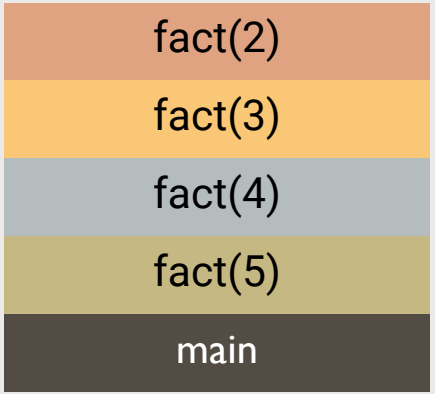


Call Stack

```c
int fact(int n)
{
    if (n == 1)
        return 1;
    else
        return n * fact(n-1);
}


int main(void)
{
    printf("%i\n", fact(5));
}
```

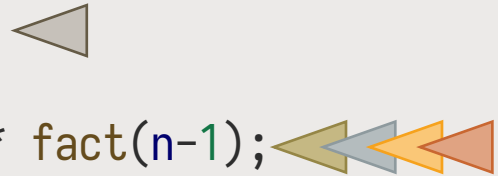
Call Stack

```c
int fact(int n)
{
    if (n == 1)
        return 1;
    else
        return n * fact(n-1);
}


int main(void)
{
    printf("%i\n", fact(5));
}
```



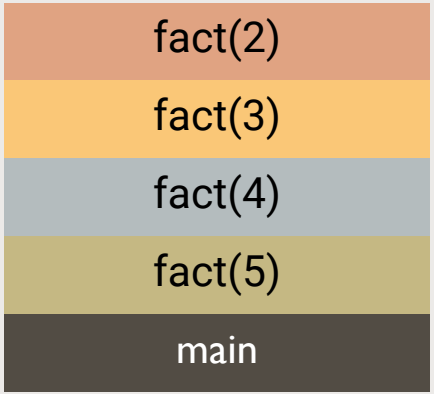
Call Stack

```c
int fact(int n)
{
    if (n == 1)
        return 1;
    else
        return n * fact(n-1);
}


int main(void)
{
    printf("%i\n", fact(5));
}
```



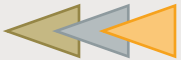
Call Stack

```c
int fact(int n)
{
    if (n == 1)
        return 1;
    else
        return n * fact(n-1);
}


int main(void)
{
    printf("%i\n", fact(5));
}
```



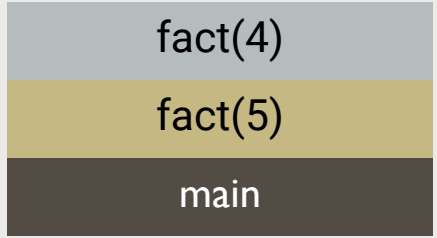
Call Stack

```c
int fact(int n)
{
    if (n == 1)
        return 1;
    else
        return n * fact(n-1);
}


int main(void)
{
    printf("%i\n", fact(5));
}
```



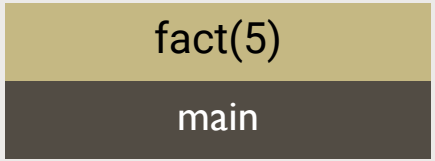
Call Stack

```c
int fact(int n)
{
    if (n == 1)
        return 1;
    else
        return n * fact(n-1);
}


int main(void)
{
    printf("%i\n", fact(5));
}
```



Call Stack

```c
int fact(int n)
{
    if (n == 1)
        return 1;
    else
        return n * fact(n-1);
}


int main(void)
{
    printf("%i\n", fact(5));
}
```

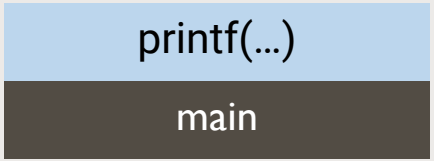


Call Stack

```c
int fact(int n)
{
    if (n == 1)
        return 1;
    else
        return n * fact(n-1);
}


int main(void)
{
    printf("%i\n", fact(5));
}
```

Call Stack

```c
int fact(int n)
{
    if (n == 1)
        return 1;
    else
        return n * fact(n-1);
}


int main(void)
{
    printf("%i\n", fact(5));
}
```



Call Stack

```c
int fact(int n)
{
    if (n == 1)
        return 1;
    else
        return n * fact(n-1);
}


int main(void)
{
    printf("%i\n", fact(5));
}
```

Call Stack

Der Call Stack speichert Frames für jede aktive Funktion im Programm.

Nur der oberste Frame auf dem Stack ist zu einem Zeitpunkt aktiv.

Jeder Frame enthält den kompletten Zustand einer Funktionsausführung.

Frames werden beim Funktionsaufruf gepusht und bei return gepoppt.

Der Stack wächst von hohen zu niedrigen Speicheradressen.

Ein rekursiver Aufruf erzeugt mehrere Frames derselben Funktion.