Freitagsrunde C-Kurs 2011

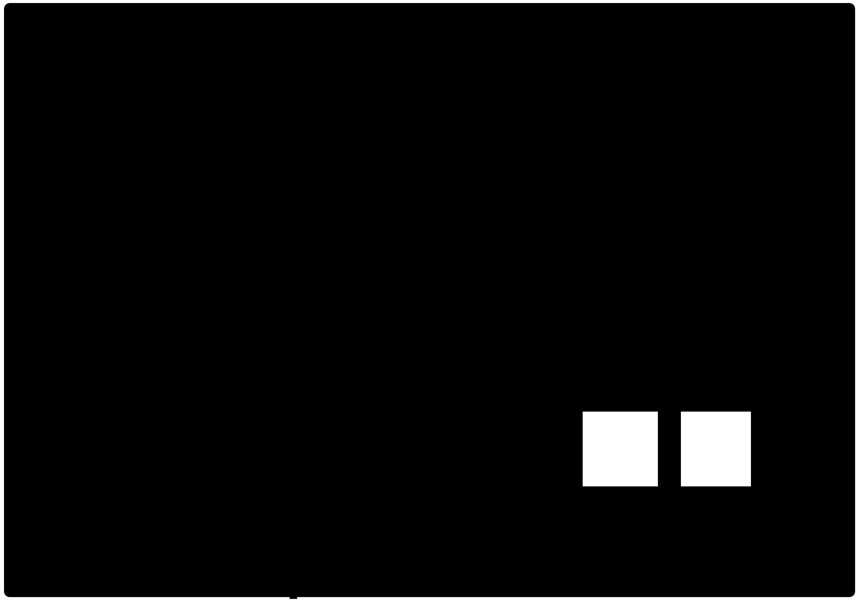
Speicherverwaltung in c

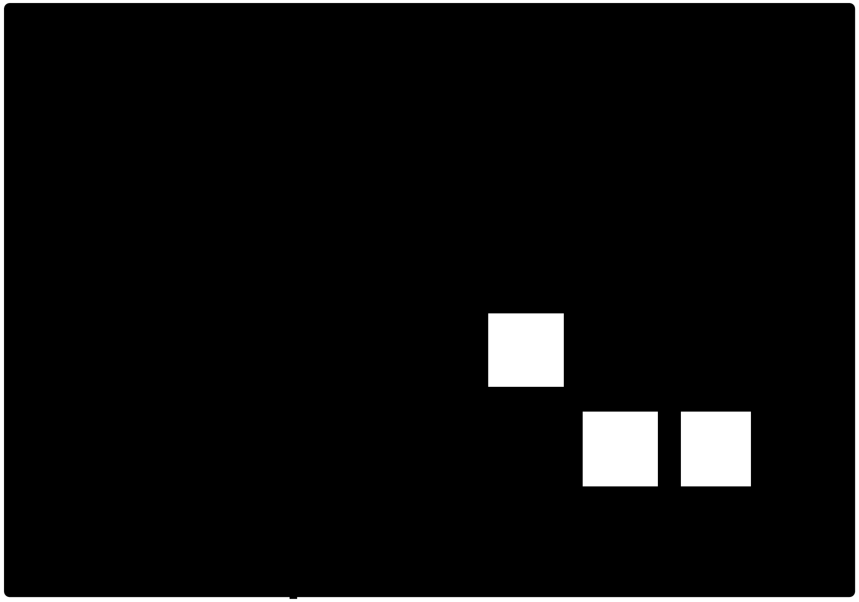
Vorlesung 2 – 14.09.2011

Katrin Lang
katrin.lang@tu-berlin.de

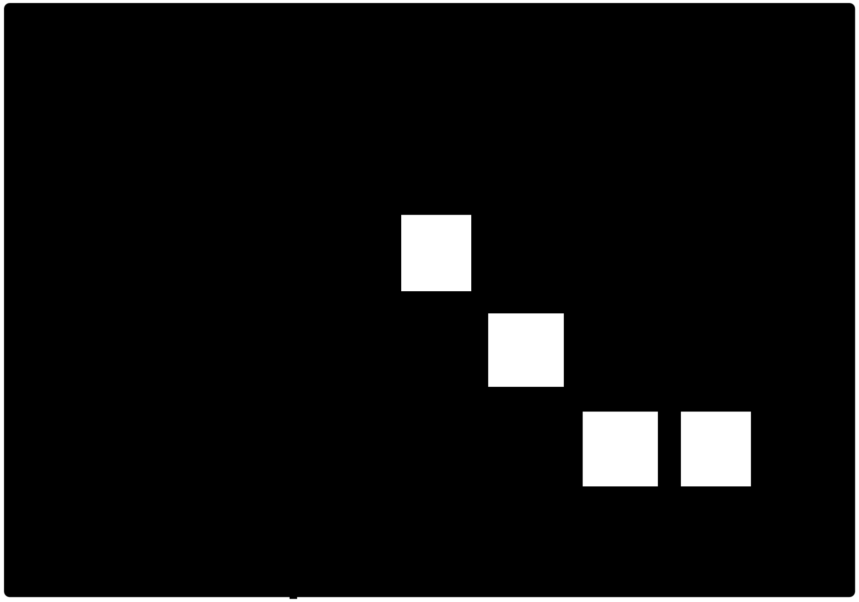
[redacted]

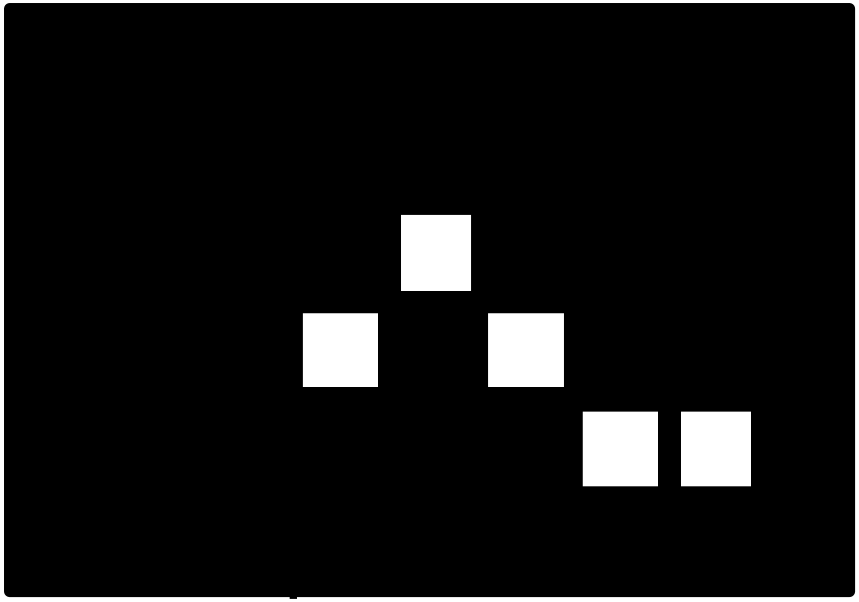
Simulation:
Spielfeld: 9x6 Kästchen
Länge der Schlange: 10

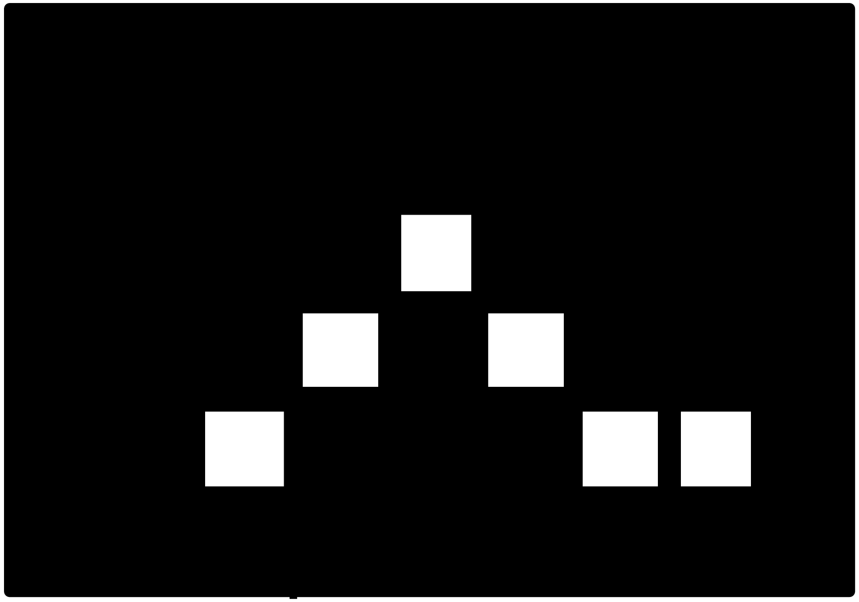


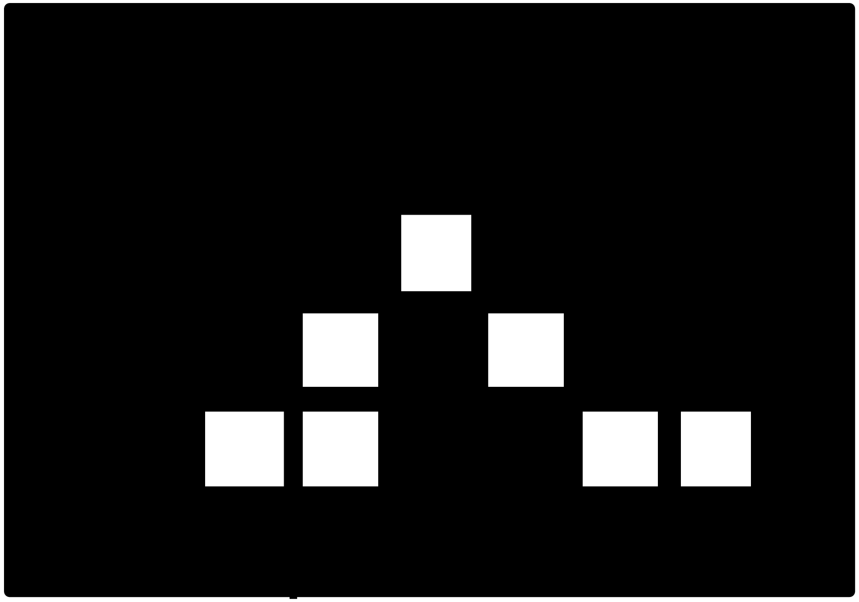


Datenstrukturen für den Screensaver

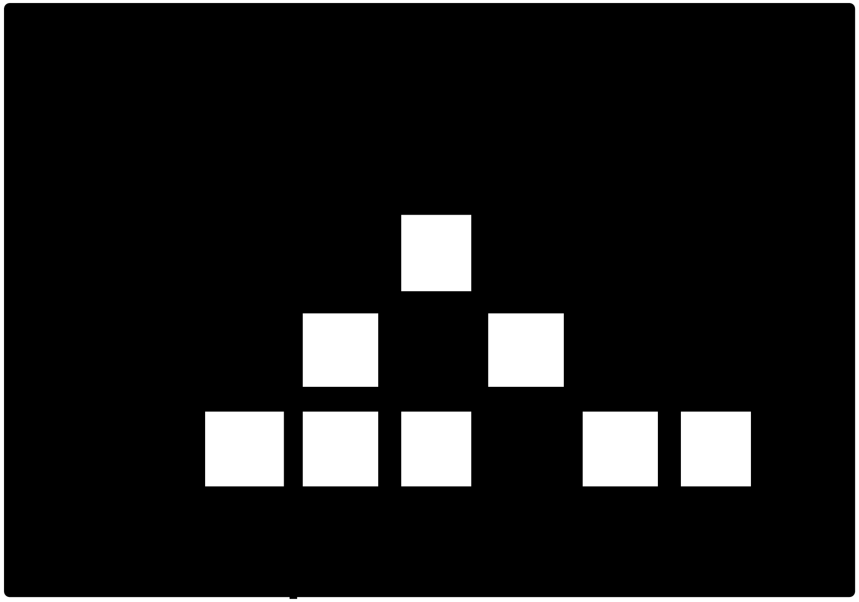


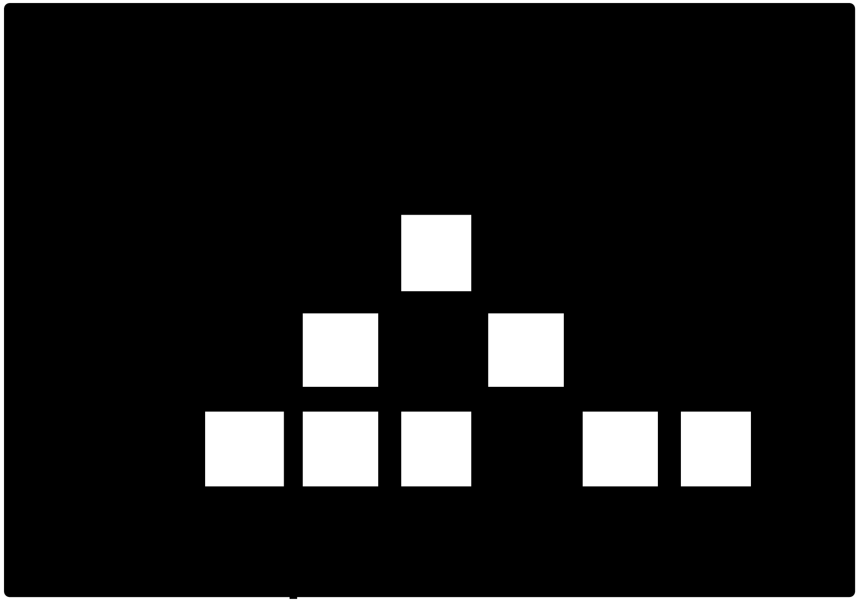


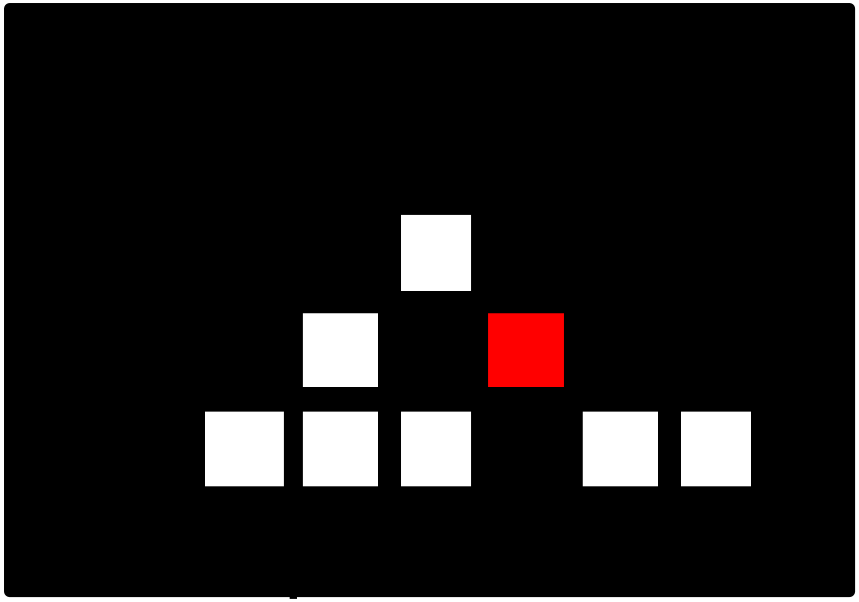


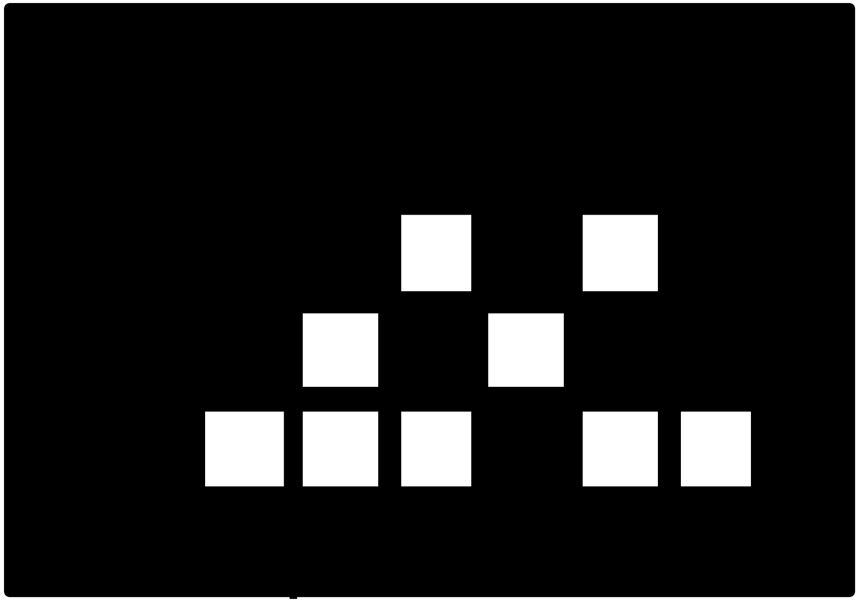


Datenstrukturen für den Screensaver

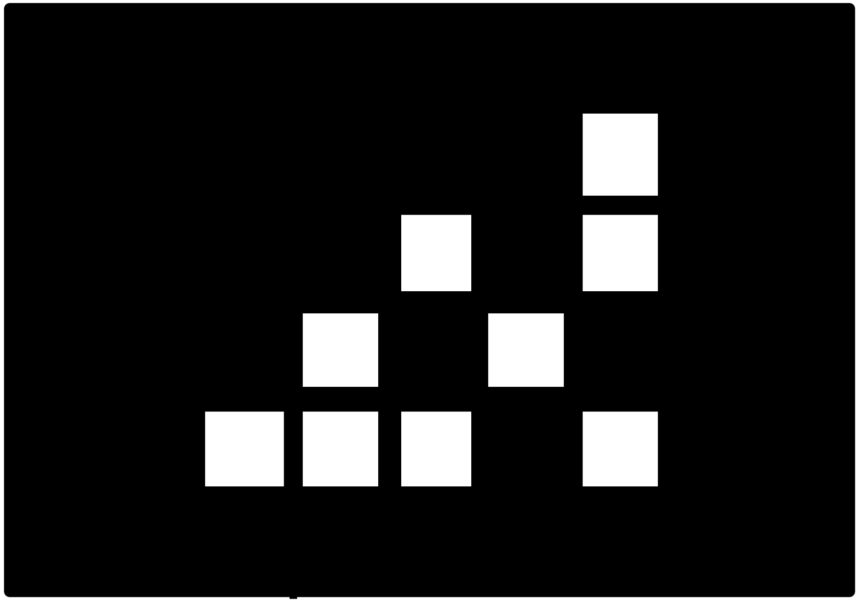


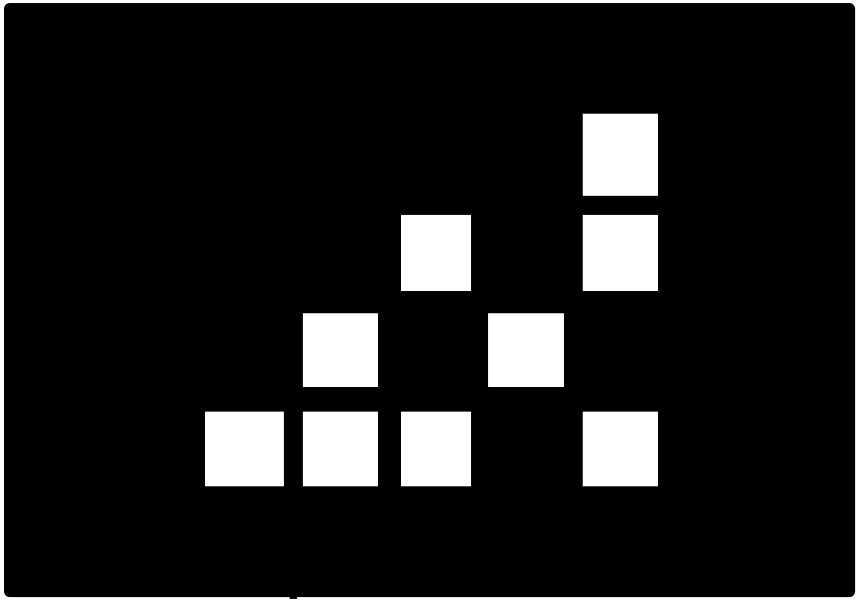






Datenstrukturen für den Screensaver





Meine Implementierung:

- Ein 2D-Array für das Spielfeld
- Eine doppelt verkettete Liste für die Schlange

Der Compiler kennt den Speicherbedarf unseres Spiels nicht, denn:

- Bildschirmgröße variiert
- Verkleinern/Vergrößern des Fensters möglich

Können wir den Speicherbedarf statisch ermitteln?

Der Compiler kennt den Speicherbedarf unseres Spiels nicht, denn:

- Bildschirmgröße variiert
- Verkleinern/Vergrößern des Fensters möglich

➜ Größe des Spielfelds muss zur Laufzeit vom Betriebssystem erfragt werden

Können wir den Speicherbedarf statisch ermitteln?

Der Compiler kennt den Speicherbedarf unseres Spiels nicht, denn:

- Bildschirmgröße variiert
- Verkleinern/Vergrößern des Fensters möglich

  ➜ Größe des Spielfelds muss zur Laufzeit vom Betriebssystem erfragt werden
- Die verkettete Liste erfordert ständiges Neuanlegen bzw. Wegwerfen von Listenelementen

Können wir den Speicherbedarf statisch ermitteln?

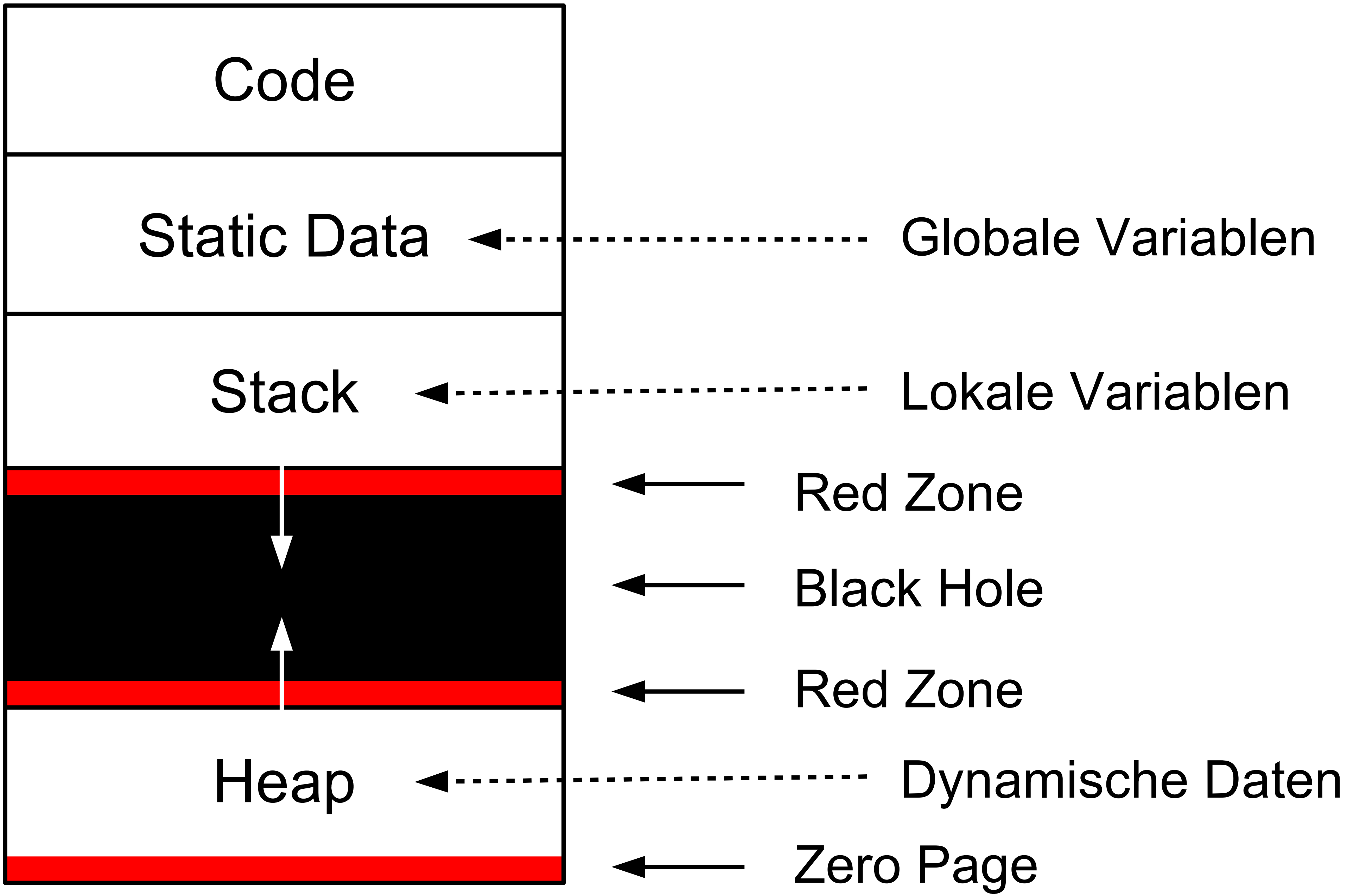
Der Compiler kennt den Speicherbedarf unseres Spiels nicht, denn:

- Bildschirmgröße variiert
- Verkleinern/Vergrößern des Fensters möglich

  ➜ Größe des Spielfelds muss zur Laufzeit vom Betriebssystem erfragt werden
- Die verkettete Liste erfordert ständiges Neuanlegen bzw. Wegwerfen von Listenelementen

  ➜ Das kann nur das Laufzeitsystem leisten

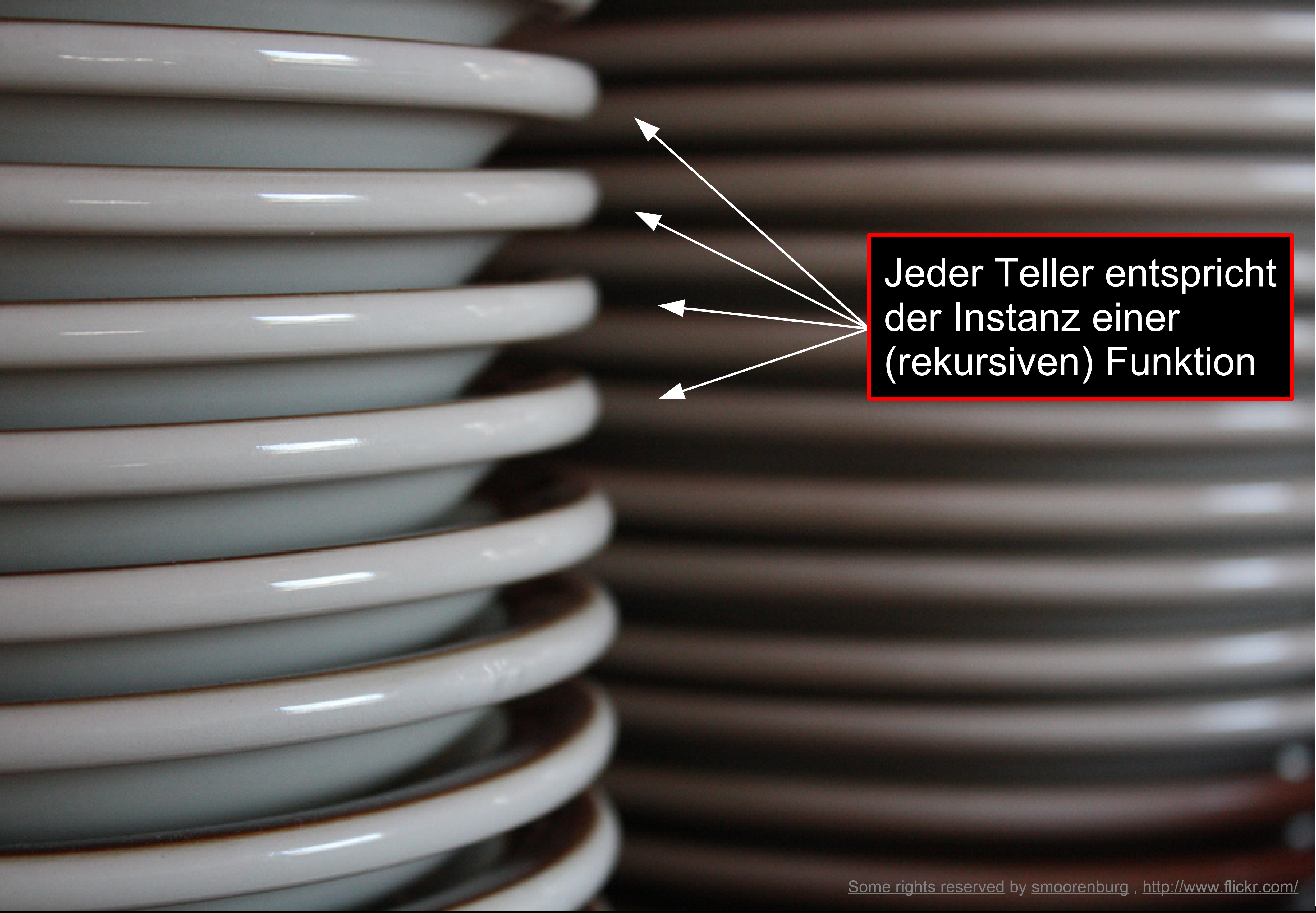
Können wir den Speicherbedarf statisch ermitteln?



Speicherorganisation eines Prozesses

Some rights reserved by smoorenburg , http://www.flickr.com/

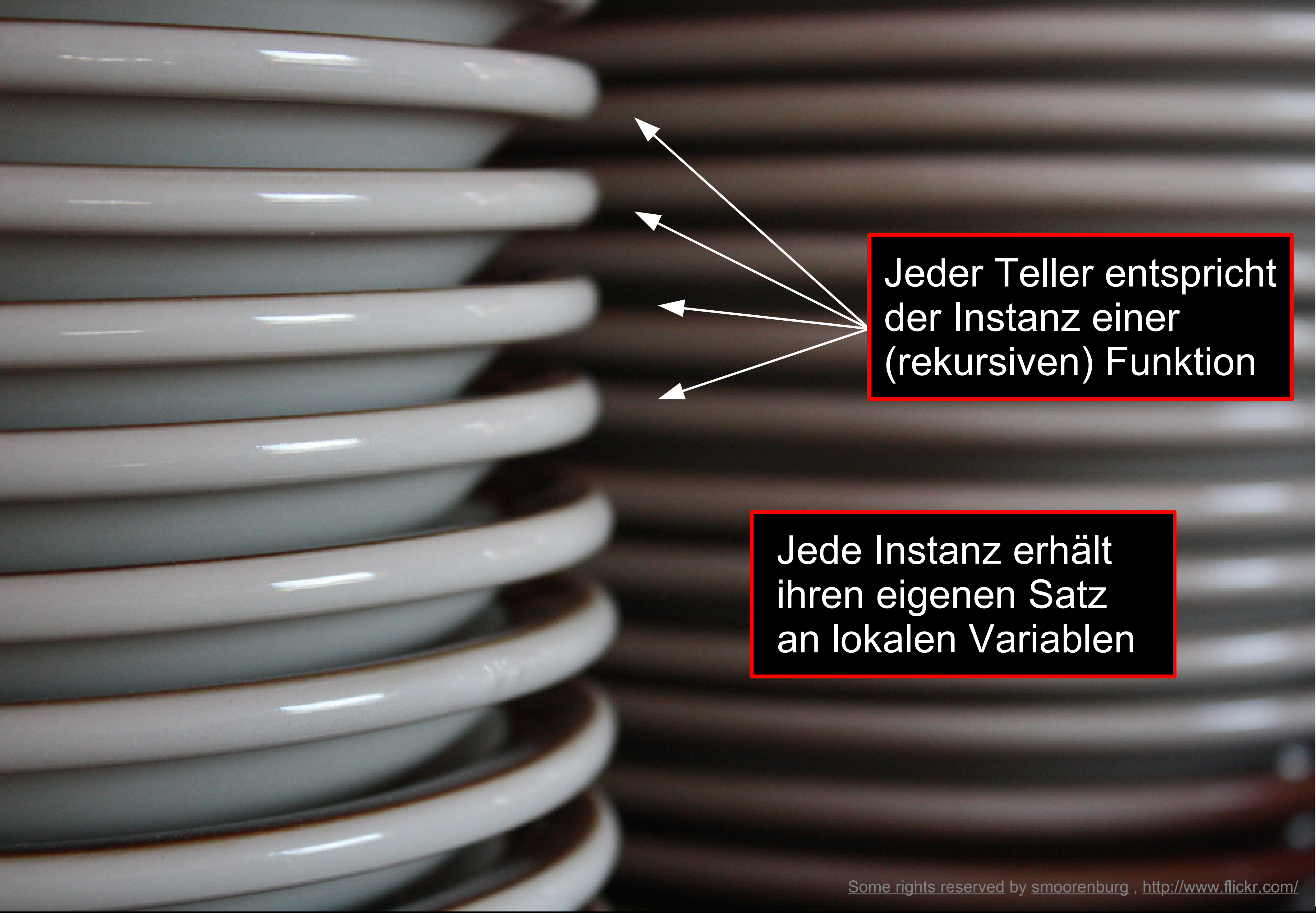
Der Stack



Jeder Teller entspricht der Instanz einer (rekursiven) Funktion

Some rights reserved by smoorenburg , http://www.flickr.com/

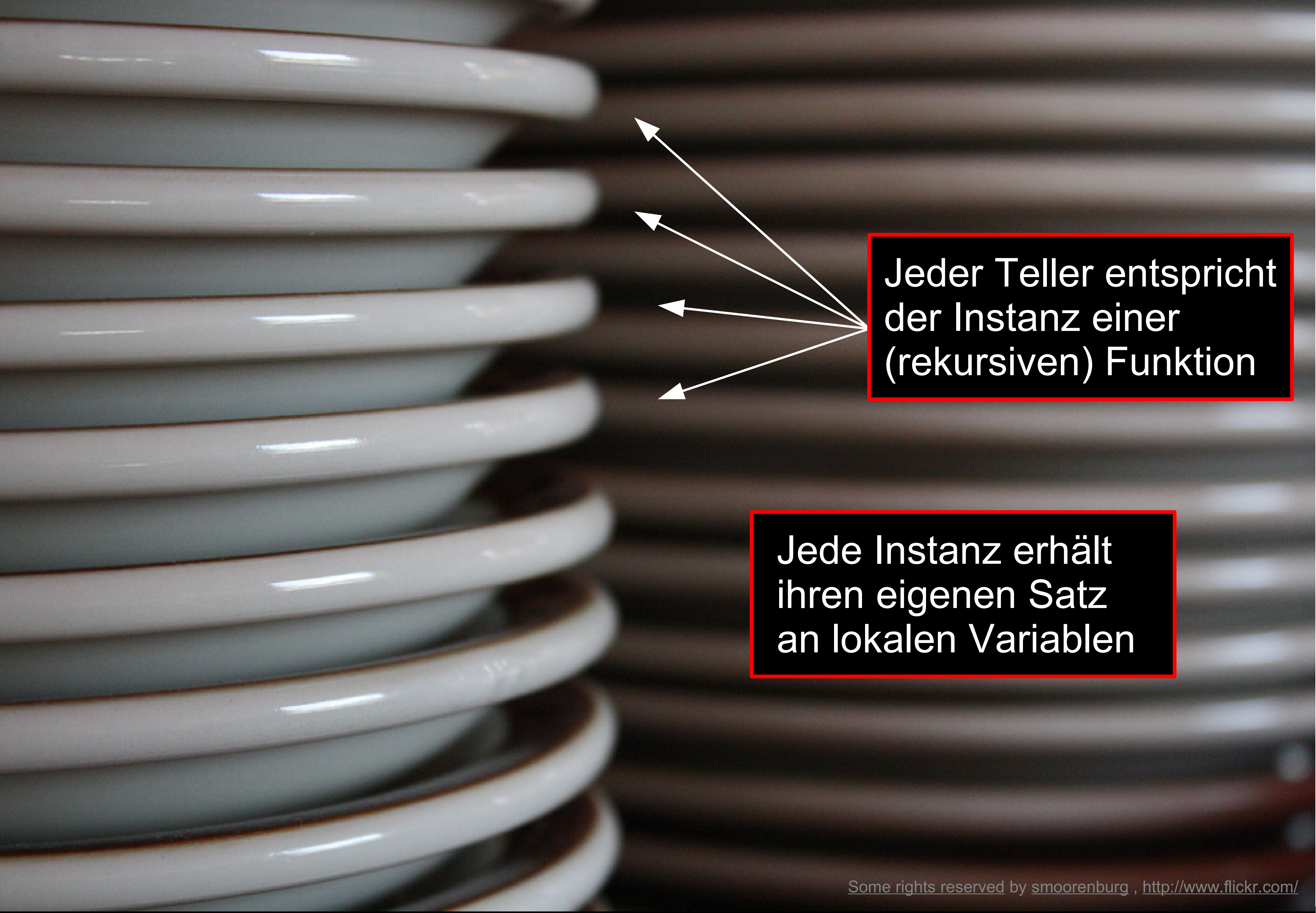
Der Stack



Jeder Teller entspricht der Instanz einer (rekursiven) Funktion

Jede Instanz erhält ihren eigenen Satz an lokalen Variablen

Some rights reserved by smoorenburg , http://www.flickr.com/



Jeder Teller entspricht der Instanz einer (rekursiven) Funktion

Jede Instanz erhält ihren eigenen Satz an lokalen Variablen

Some rights reserved by smoorenburg , http://www.flickr.com/

Der Stack – implementiert Rekursion!

```
int main(void){
    while(true) {
        play(screen, field, snake);
    }
}
void play(SDL_Surface *screen, Field *field, Snake *snake){
    crawl(snake, size(field));
    draw(snake, field);
}
void crawl(Snake *snake, Point fieldSize){
   push(snake, fieldSize);
   pop(snake);
}
```

Der Main-Loop des Screensavers

```
int main(void){

    while(true) {
        play(screen, field, snake);
    }
}

void play(SDL_Surface *screen, Field *field, Snake *snake){

    crawl(snake, size(field));
    draw(snake, field);
}

void crawl(Snake *snake, Point fieldSize){

   push(snake, fieldSize);
   pop(snake);
}
```

Der Main-Loop des Screensavers

```
int main(void){
    while(true) {
       play(screen, field, snake);
    }
}

void play(SDL_Surface *screen, Field *field, Snake *snake){
    crawl(snake, size(field));
    draw(snake, field);
}

void crawl(Snake *snake, Point fieldSize){
   push(snake, fieldSize);
   pop(snake);
}
```

Der Main-Loop des Screensavers

```
int main(void){
    while(true) {
        play(screen, field, snake);
    }
}
void play(SDL_Surface *screen, Field *field, Snake *snake){
    crawl(snake, size(field));
    draw(snake, field);
}
void crawl(Snake *snake, Point fieldSize){
   push(snake, fieldSize);
   pop(snake);
}
```

Der Main-Loop des Screensavers

```
int main(void){
    while(true) {
       play(screen, field, snake);
    }
}
void play(SDL_Surface *screen, Field *field, Snake *snake){
    crawl(snake, size(field));
    draw(snake, field);
}
void crawl(Snake *snake, Point fieldSize){
   push(snake, fieldSize);
   pop(snake);
}
```

Der Main-Loop des Screensavers

```
int main(void){
    while(true) {
        play(screen, field, snake);
    }
}
void play(SDL_Surface *screen, Field *field, Snake *snake){
    crawl(snake, size(field));
    draw(snake, field);
}
void crawl(Snake *snake, Point fieldSize){
   push(snake, fieldSize);
   pop(snake);
}
```

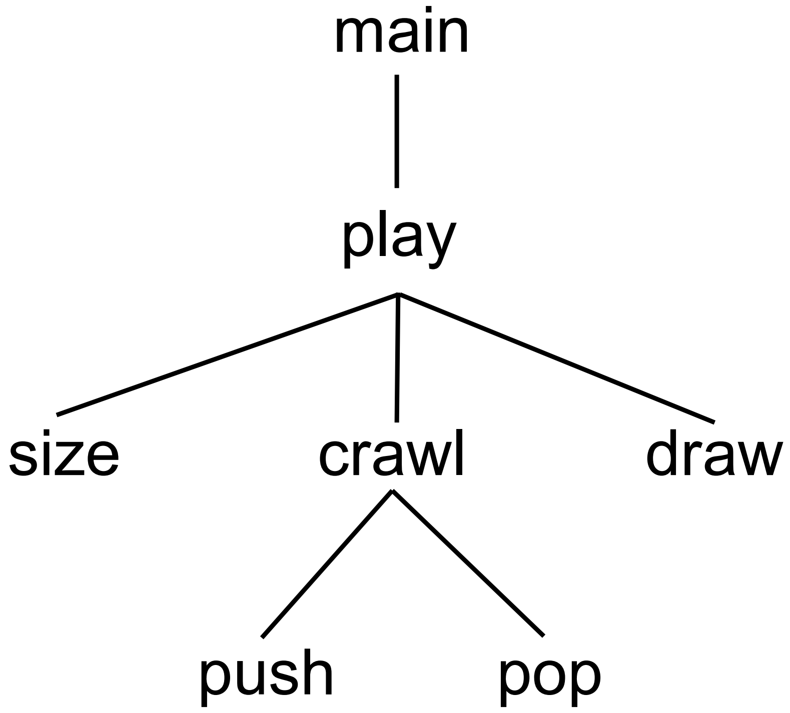
Der Main-Loop des Screensavers

```
int main(void){
    while(true) {
       play(screen, field, snake);
    }
}
void play(SDL_Surface *screen, Field *field, Snake *snake){
    crawl(snake, size(field));
    draw(snake, field);
}
void crawl(Snake *snake, Point fieldSize){
   push(snake, fieldSize);
   pop(snake);
}
```

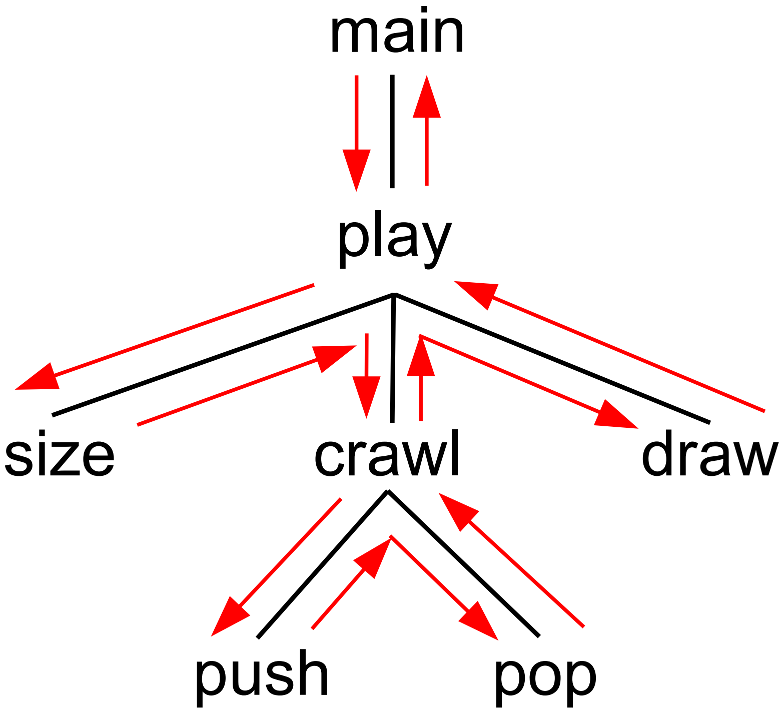
Der Main-Loop des Screensavers

```
int main(void){
    while(true) {
       play(screen, field, snake);
    }
}
void play(SDL_Surface *screen, Field *field, Snake *snake){
    crawl(snake, size(field));
    draw(snake, field);
}
void crawl(Snake *snake, Point fieldSize){
   push(snake, fieldSize);
   pop(snake);
}
```

Der Main-Loop des Screensavers

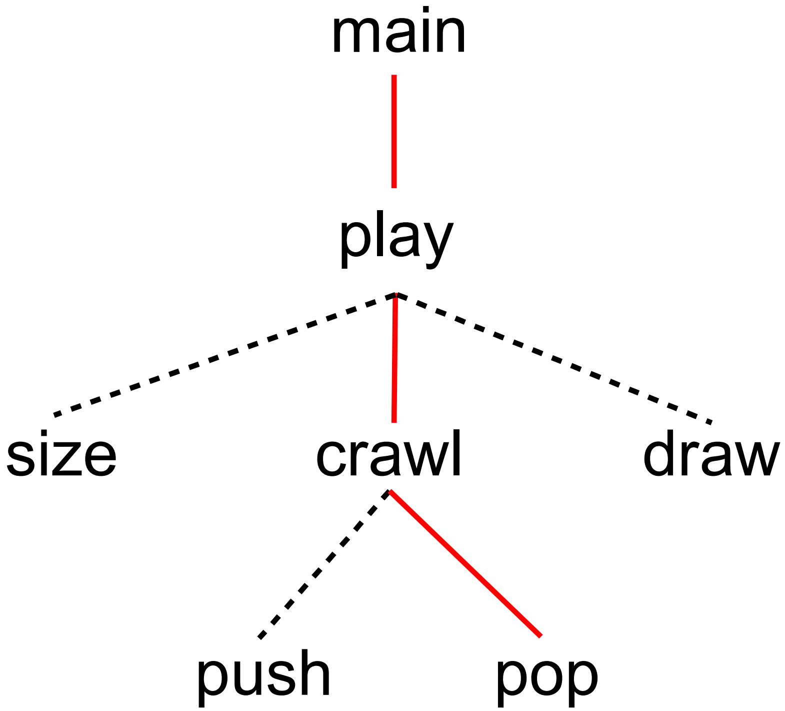


Der Call Tree des Screensavers



Preorder-Traversierung == Kontrollfluss

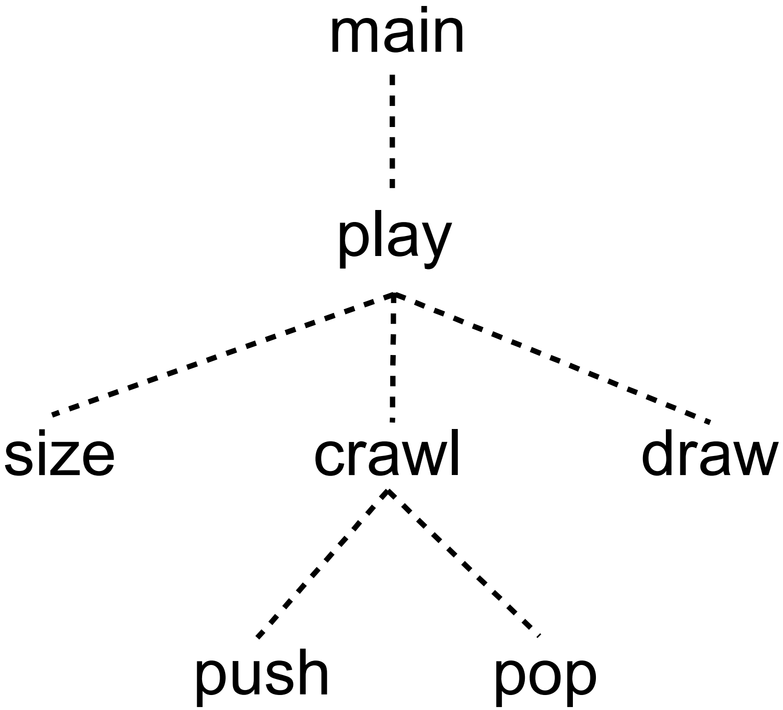
Der Call Tree des Screensavers



Pfad == Zustand des Stacks

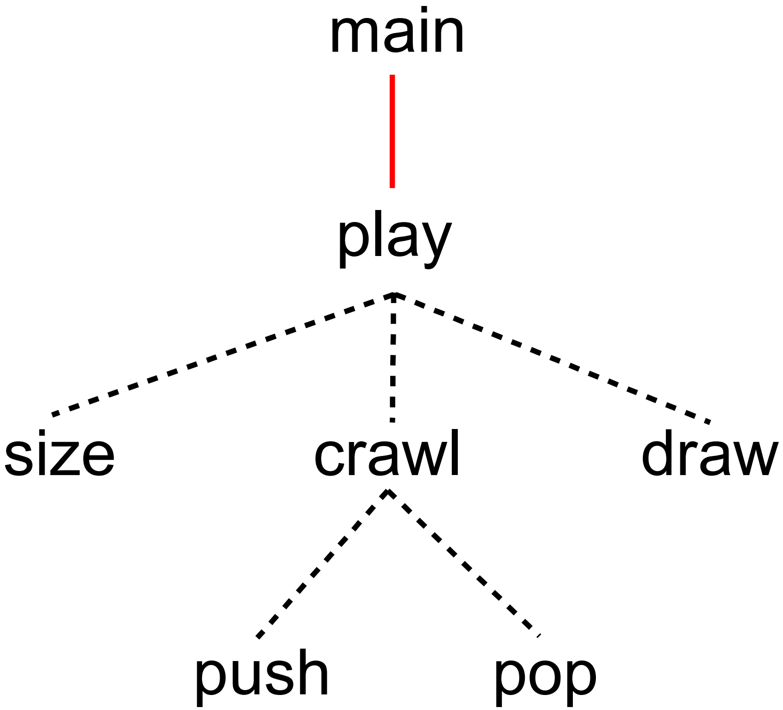
Preorder-Traversierung == Kontrollfluss

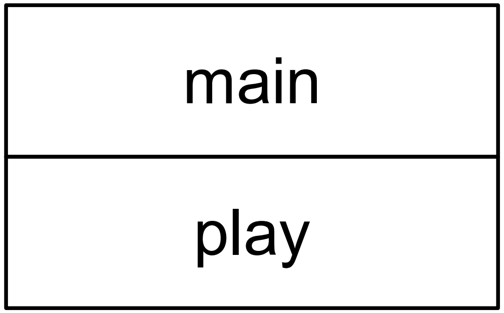
Der Call Tree des Screensavers



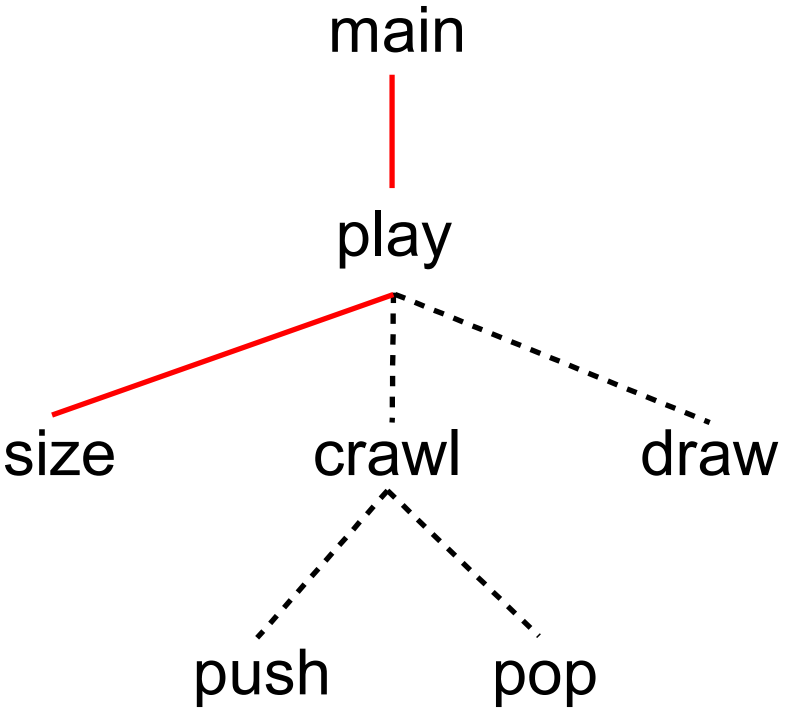
main

Der Call Tree des Screensavers



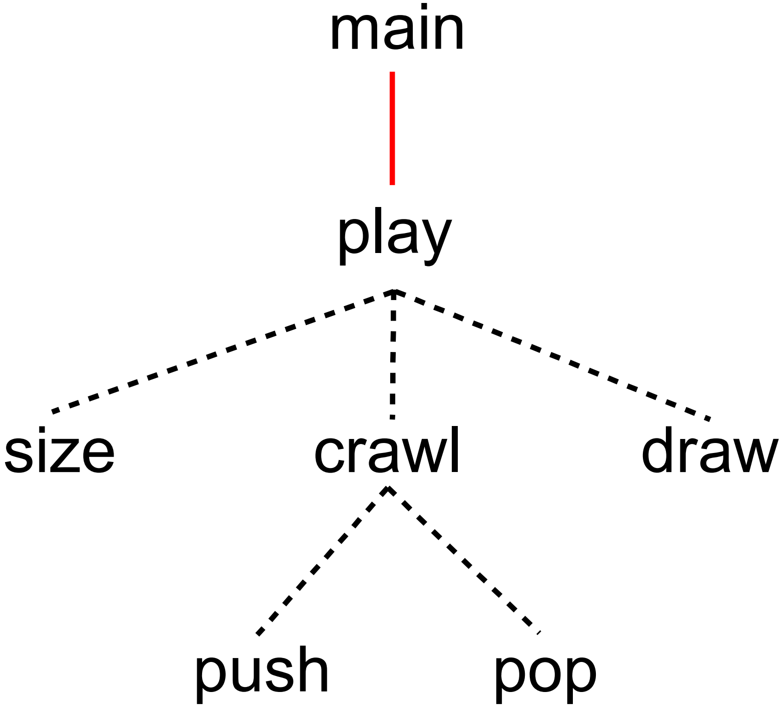


Der Call Tree des Screensavers



main
play
size

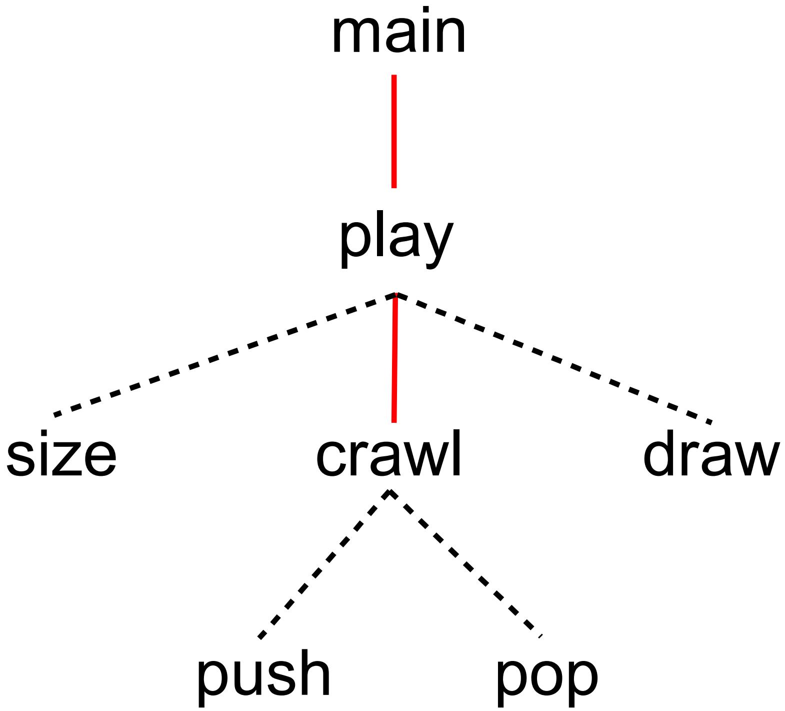
Der Call Tree des Screensavers



main

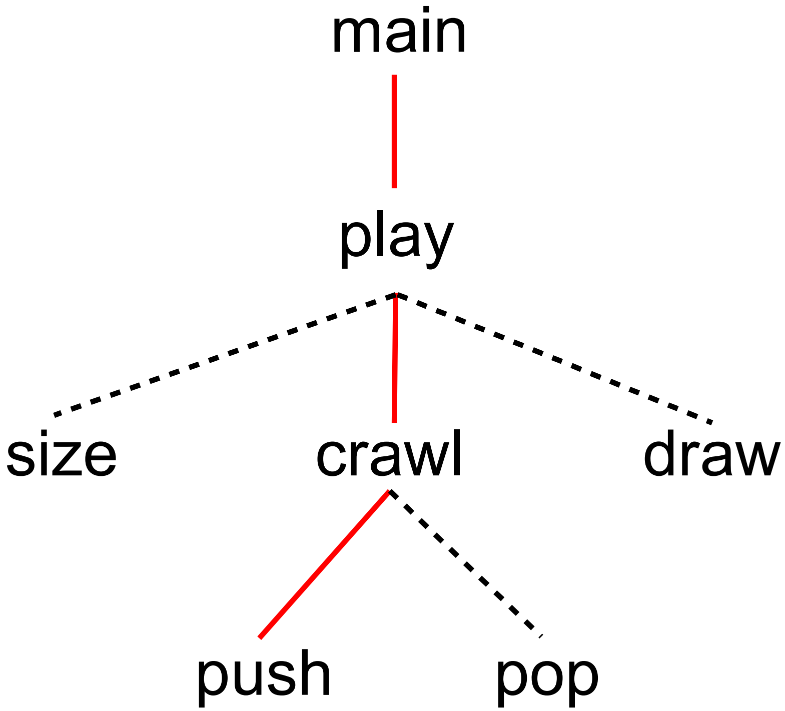
play

Der Call Tree des Screensavers



main
play
crawl

Der Call Tree des Screensavers



main

play

size

crawl

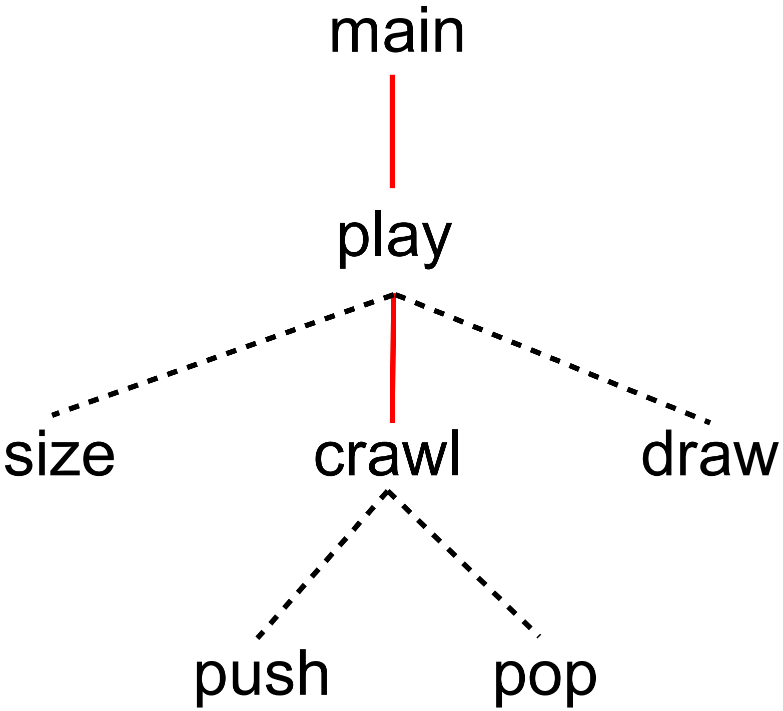
draw

push

pop

main
play
crawl
push

Der Call Tree des Screensavers



main

play

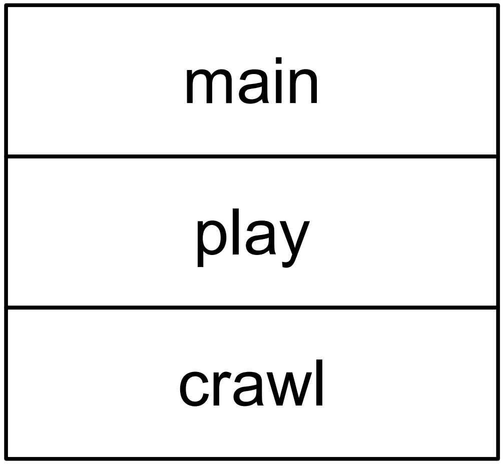
size

crawl

draw

push

pop

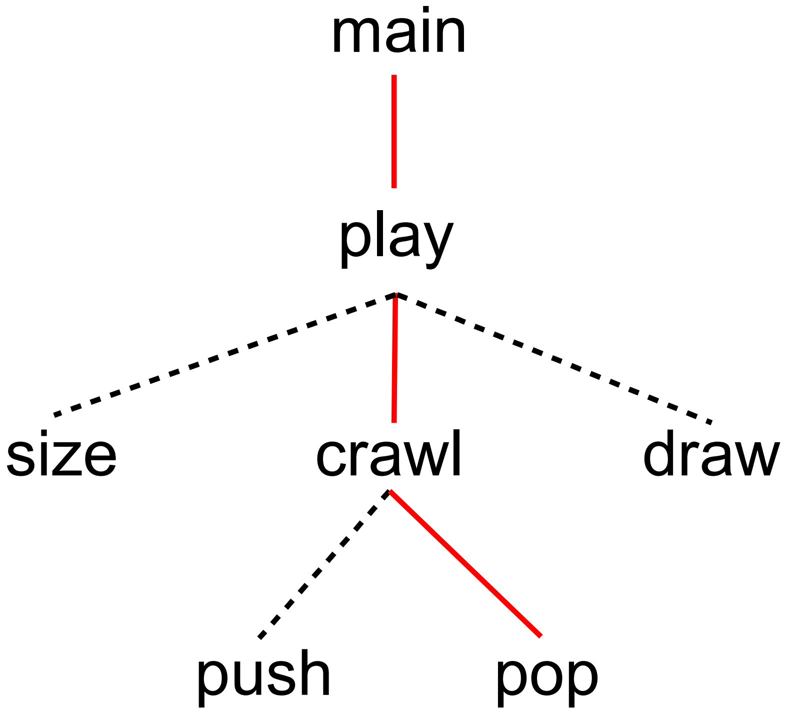


main

play

crawl

Der Call Tree des Screensavers



main

play

size

crawl

draw

push

pop

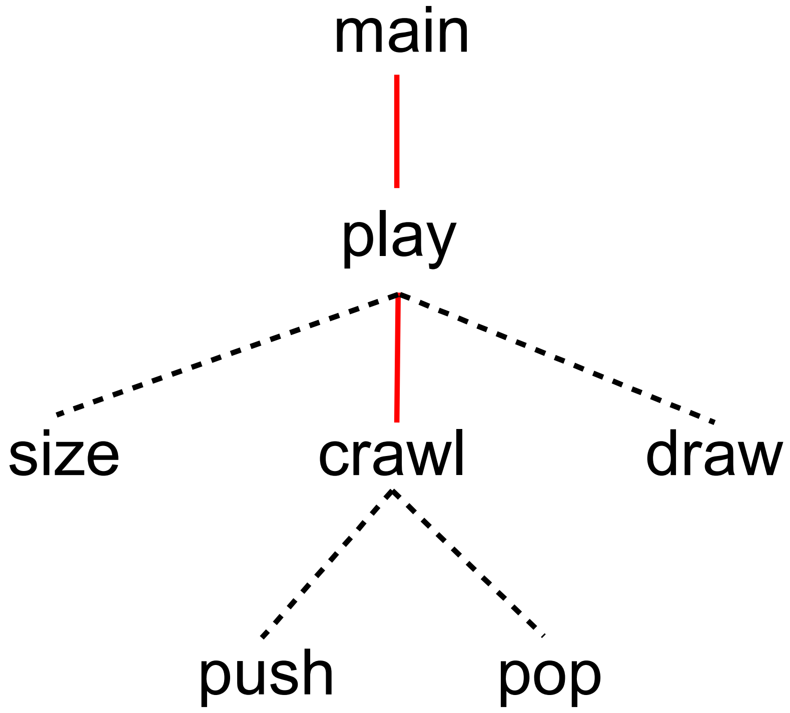
main

play

crawl

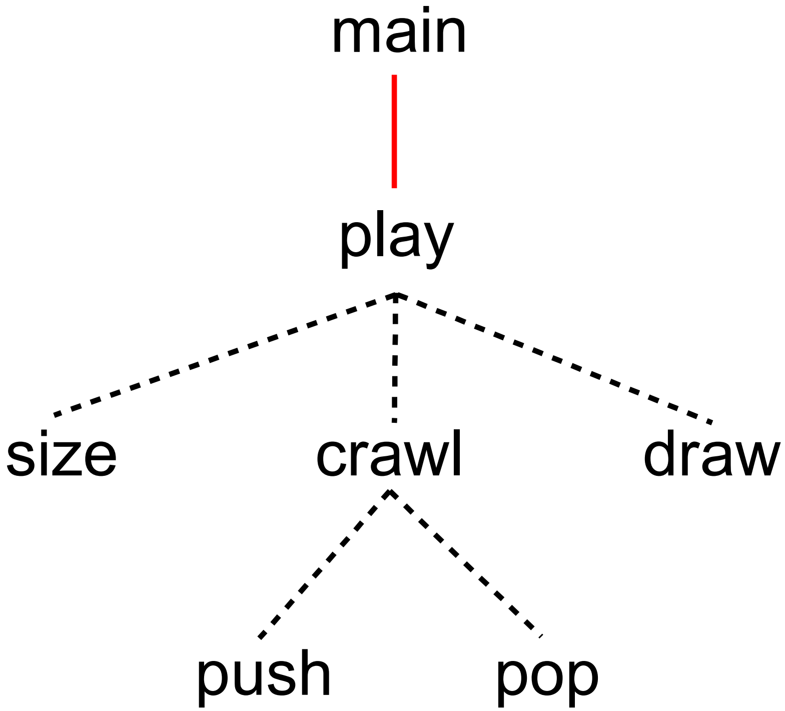
pop

Der Call Tree des Screensavers



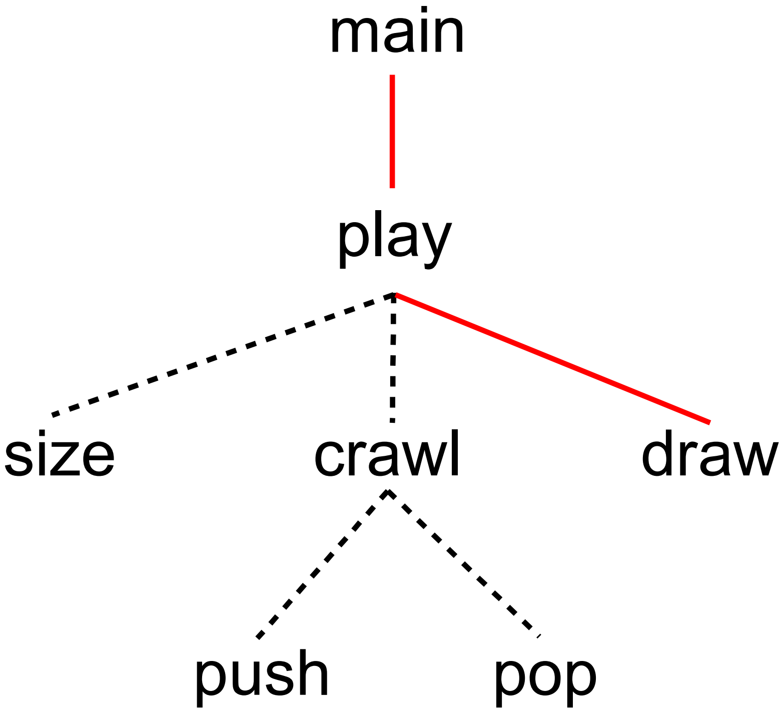
main
play
crawl

Der Call Tree des Screensavers



main
play

Der Call Tree des Screensavers



main

play

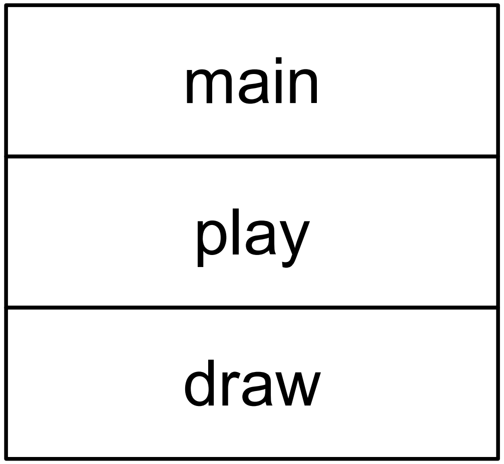
size

crawl

draw

push

pop

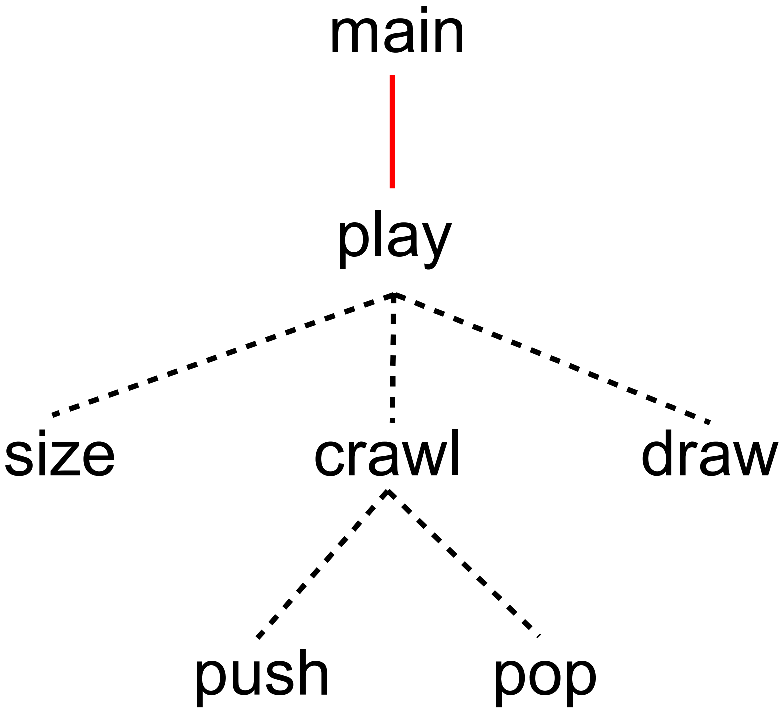


main

play

draw

Der Call Tree des Screensavers



main

play

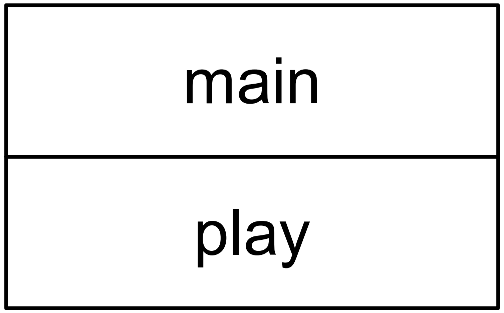
size

crawl

draw

push

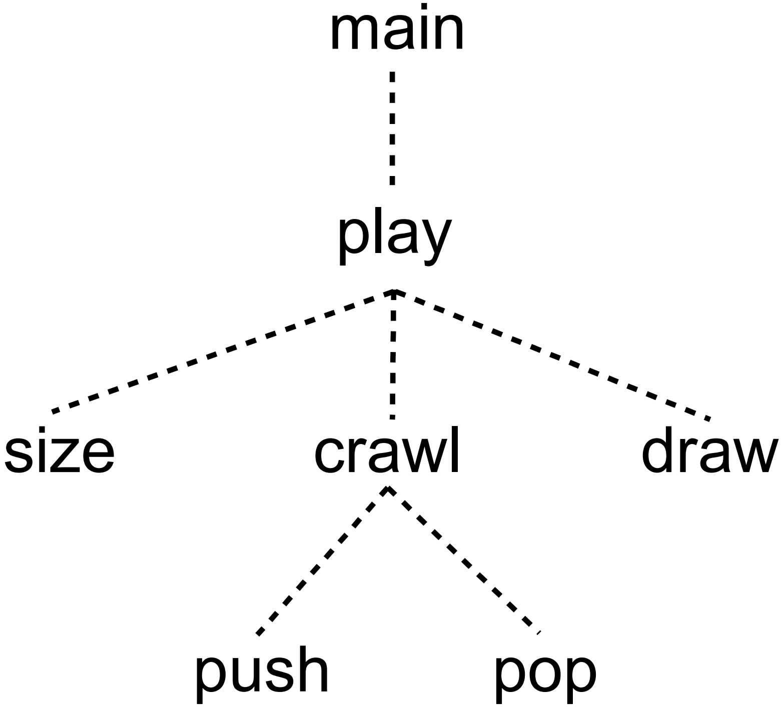
pop



main

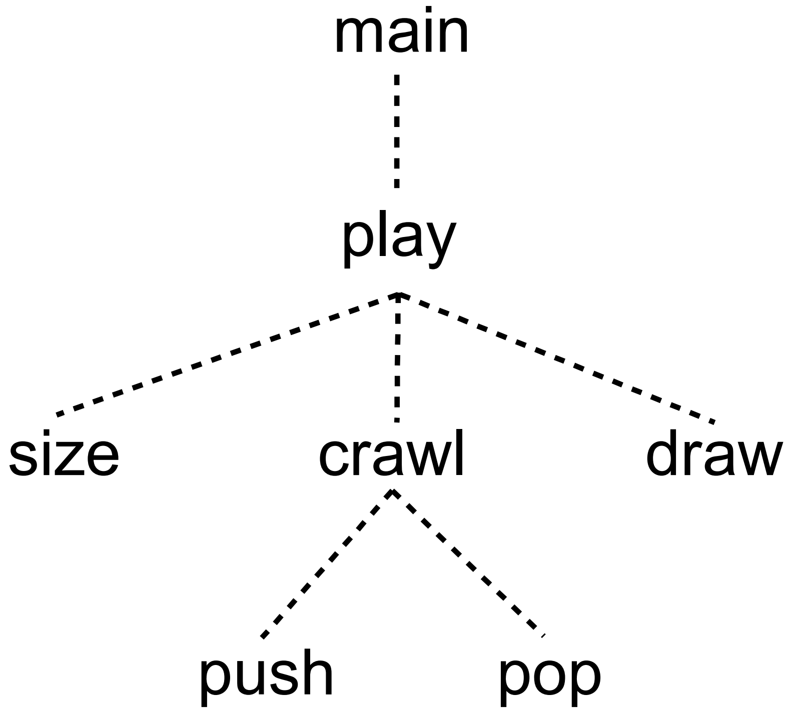
play

Der Call Tree des Screensavers



main

Der Call Tree des Screensavers


main

play

size

crawl

draw

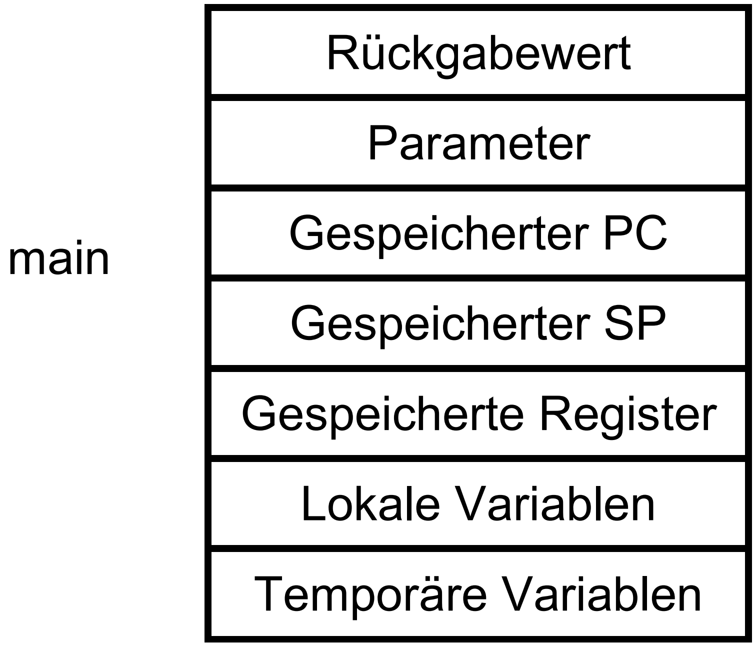
push

pop

Stackframe

main

Der Call Tree des Screensavers



main

Rückgabewert

Parameter

Gespeicherter PC

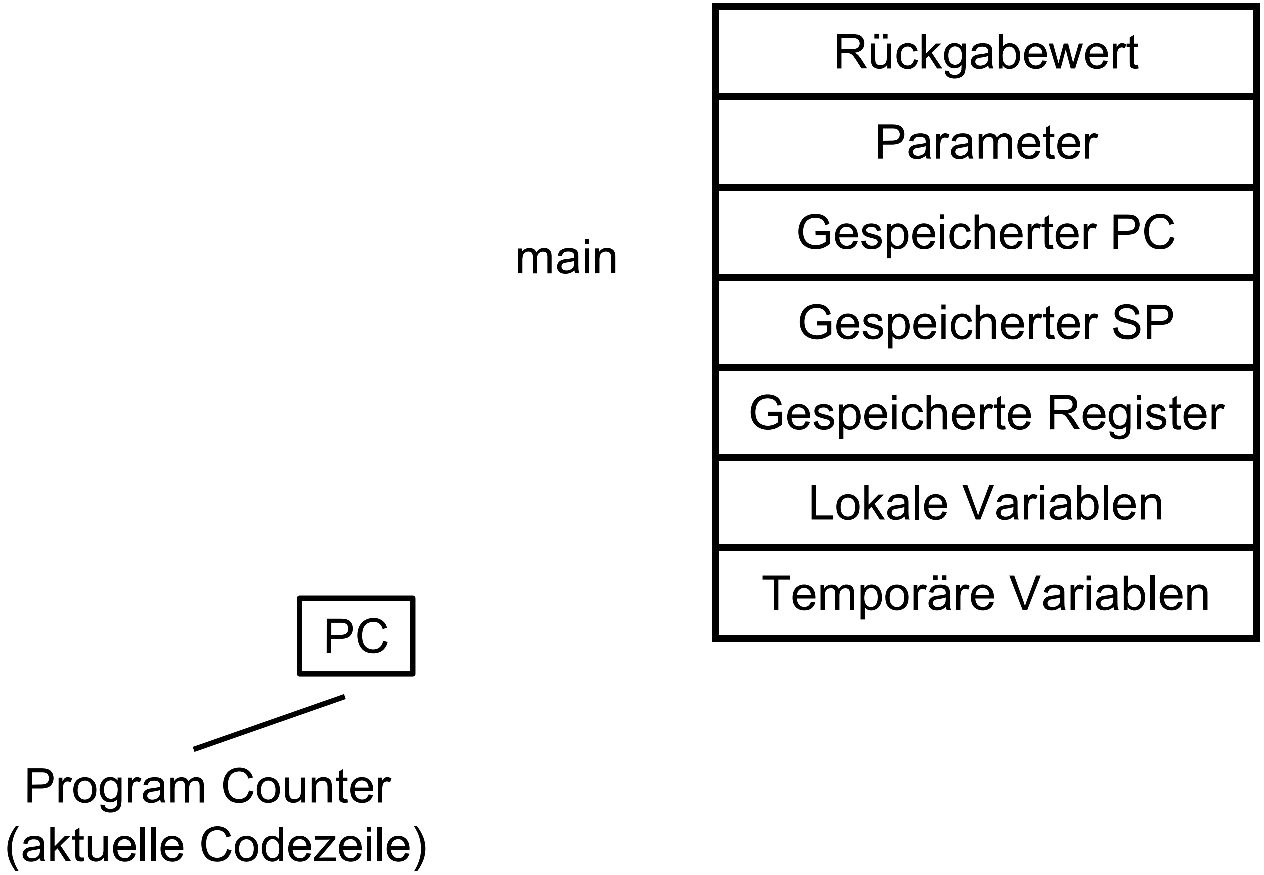
Gespeicherter SP

Gespeicherte Register

Lokale Variablen

Temporäre Variablen

Stack Frame



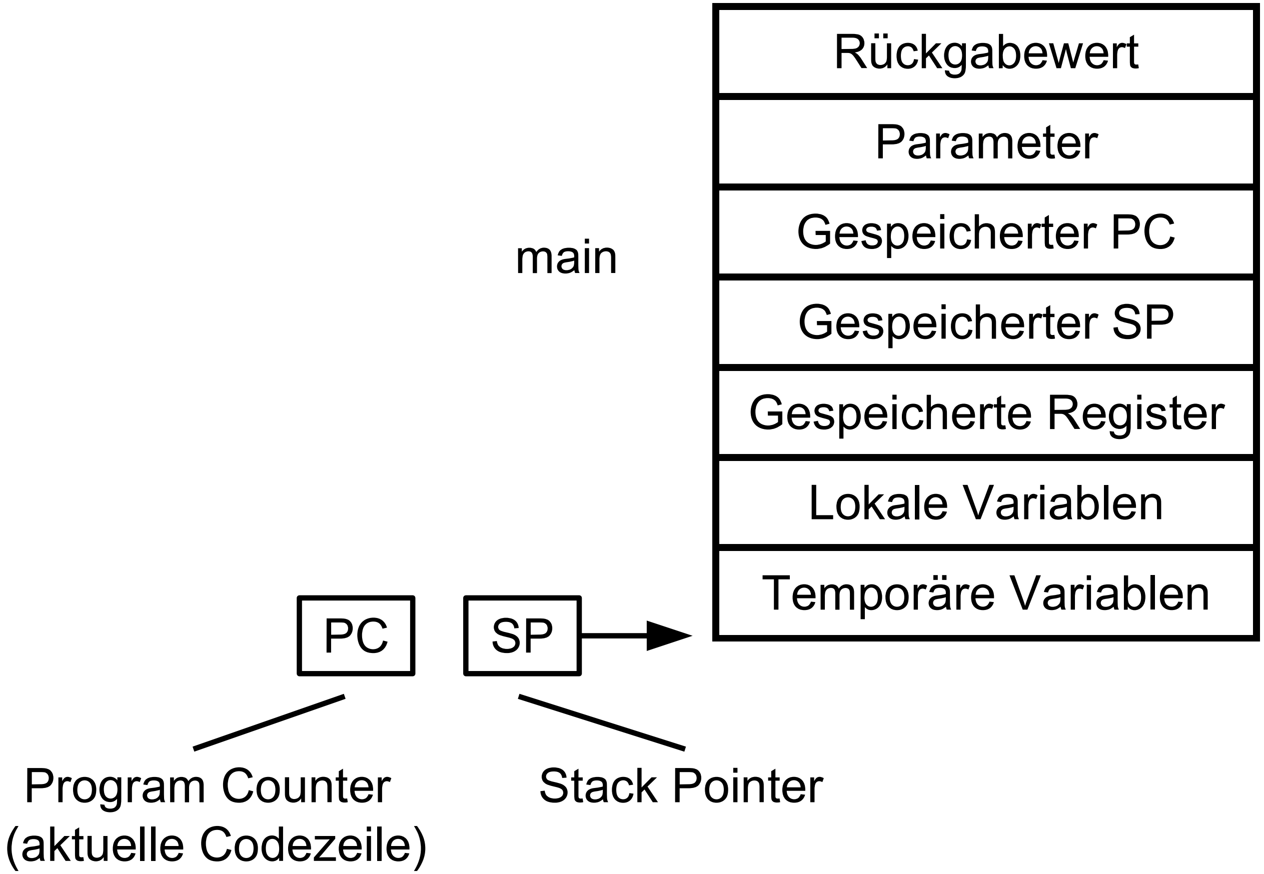
main

Rückgabewert
Parameter
Gespeicherter PC
Gespeicherter SP
Gespeicherte Register
Lokale Variablen
Temporäre Variablen

PC

Program Counter
(aktuelle Codezeile)

Stack Frame



main

Rückgabewert

Parameter

Gespeicherter PC

Gespeicherter SP

Gespeicherte Register

Lokale Variablen

Temporäre Variablen

PC

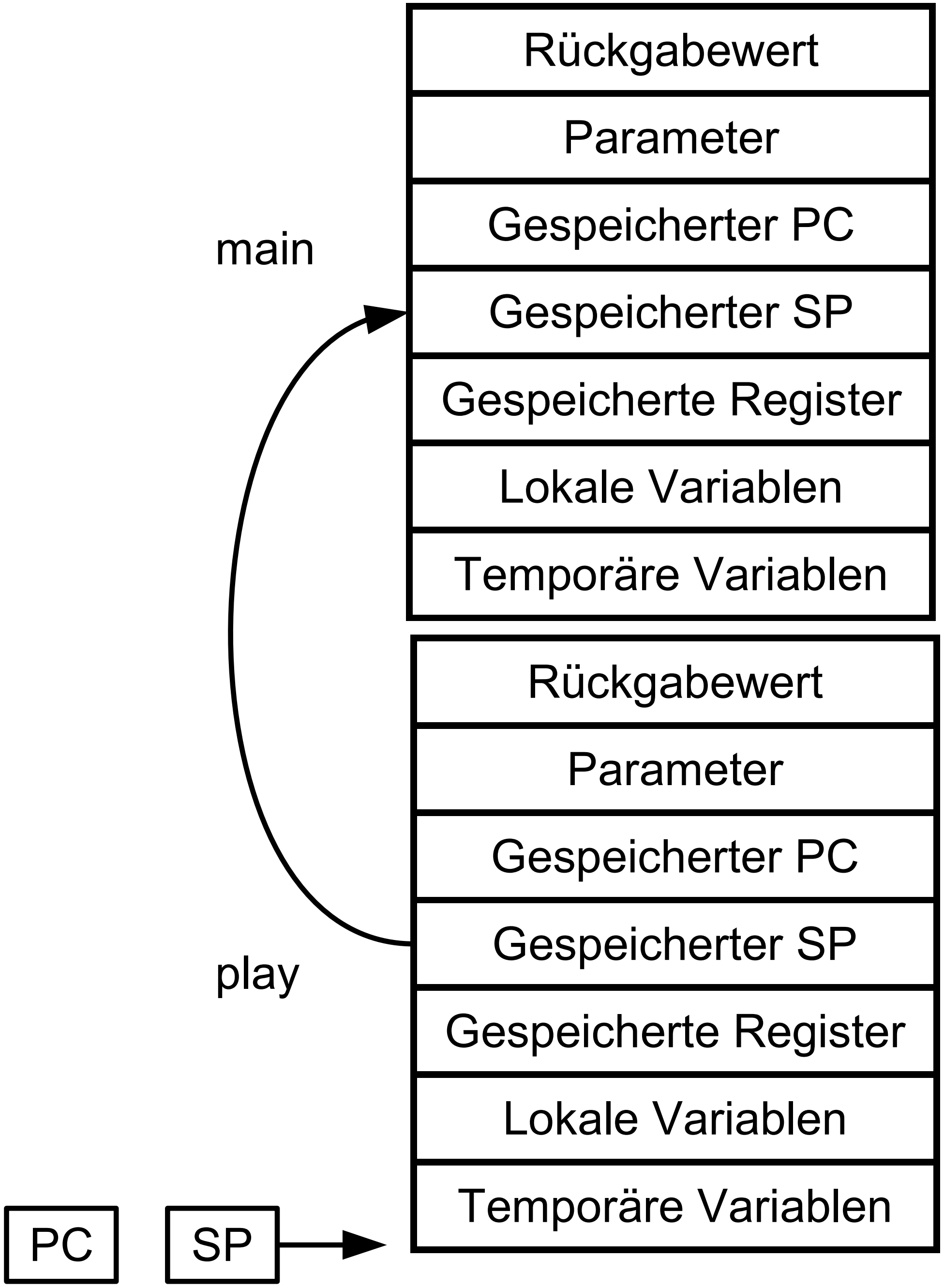
SP

Program Counter
(aktuelle Codezeile)

Stack Pointer

Stack Frame

Stack Frame



main

Rückgabewert

Parameter

Gespeicherter PC

Gespeicherter SP

Gespeicherte Register

Lokale Variablen

Temporäre Variablen

play

Rückgabewert

Parameter

Gespeicherter PC

Gespeicherter SP

Gespeicherte Register

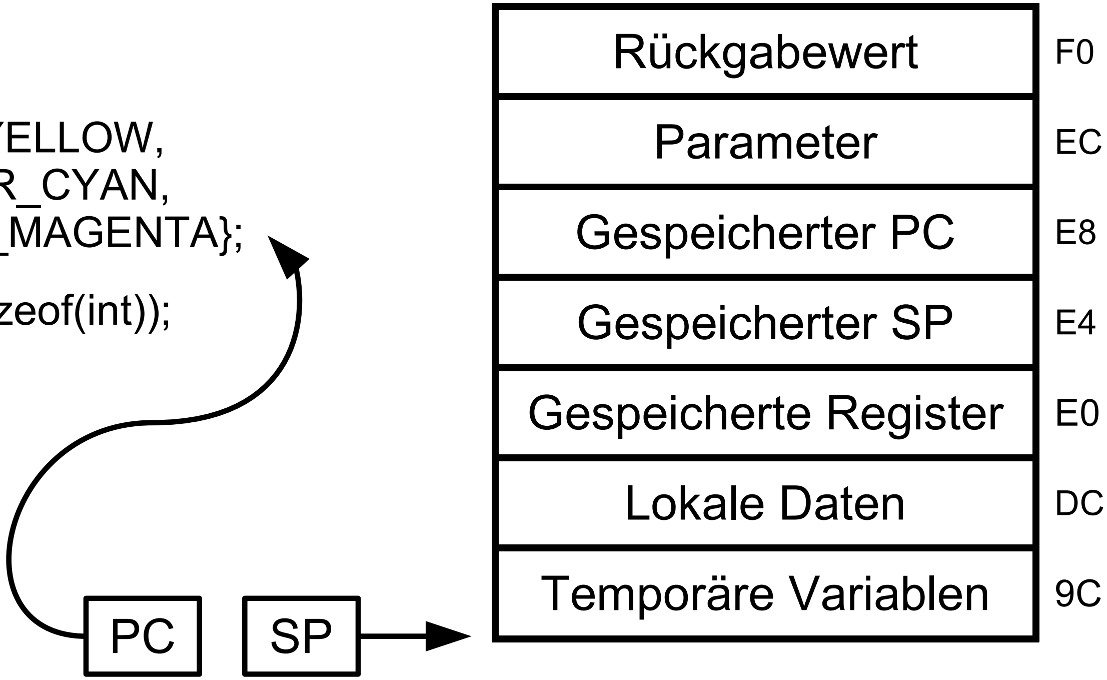
Lokale Variablen

Temporäre Variablen

PC

SP

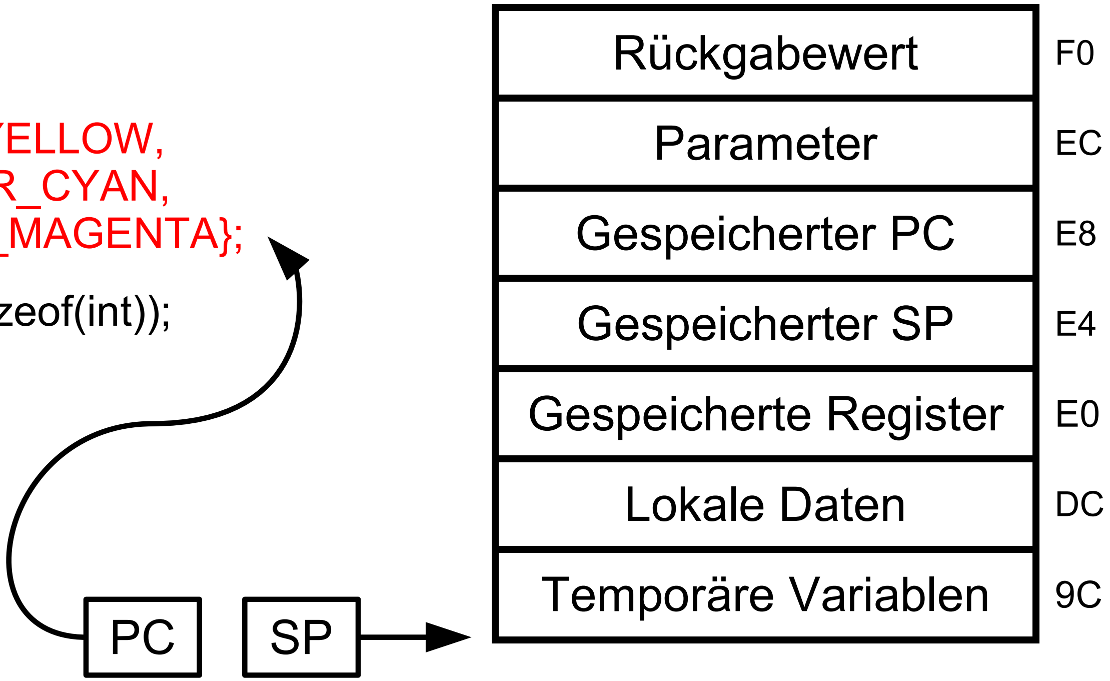
```
int main(void){

    int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                    COLOR_GREEN, COLOR_CYAN,
                    COLOR_BLUE, COLOR_MAGENTA};

    printRainbow(rainbow, sizeof(rainbow)/sizeof(int));

}
```



Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
Lokale Daten	DC
Temporäre Variablen	9C

```
void printRainbow(int rainbow[], int rainbowLength){

    for(int i= 0; i<rainbowLength; i++){

        attron(COLOR_PAIR(rainbow[i]));

        addstr("\u2588");

        attroff(COLOR_PAIR(rainbow[i]));

    }

}
```

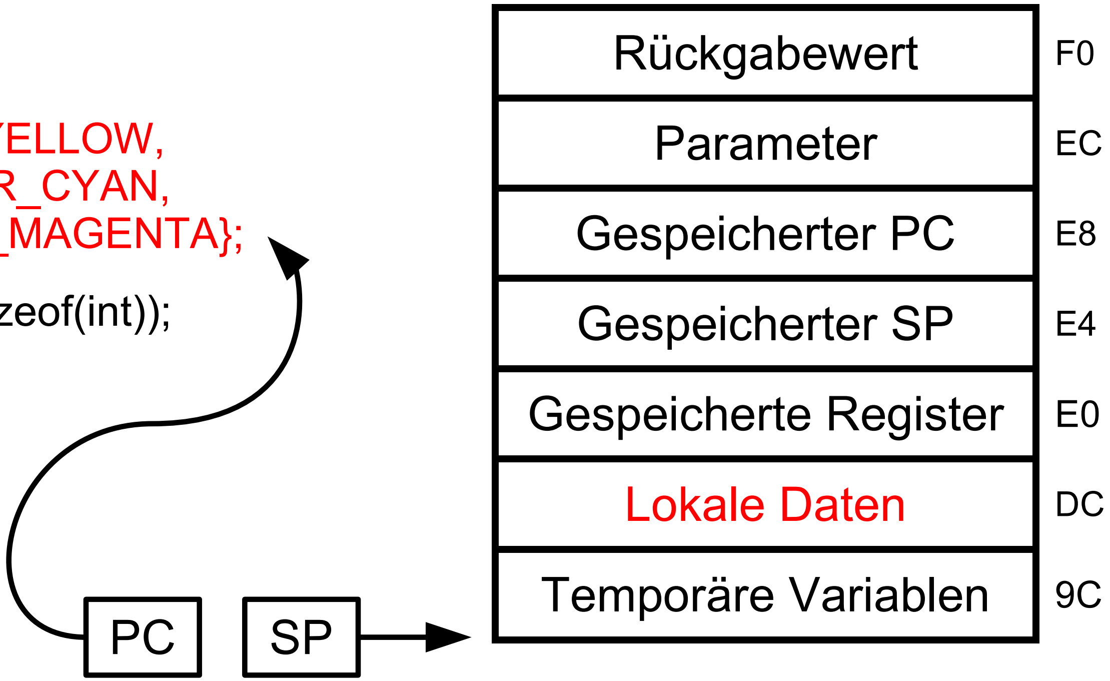
```
int main(void){

    int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                    COLOR_GREEN, COLOR_CYAN,
                    COLOR_BLUE, COLOR_MAGENTA};

    printRainbow(rainbow, sizeof(rainbow)/sizeof(int));
}

void printRainbow(int rainbow[], int rainbowLength){

     for(int i= 0; i<rainbowLength; i++){
         attron(COLOR_PAIR(rainbow[i]));
         addstr("\u2588");
         attroff(COLOR_PAIR(rainbow[i]));
    }
}
```



Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
Lokale Daten	DC
Temporäre Variablen	9C

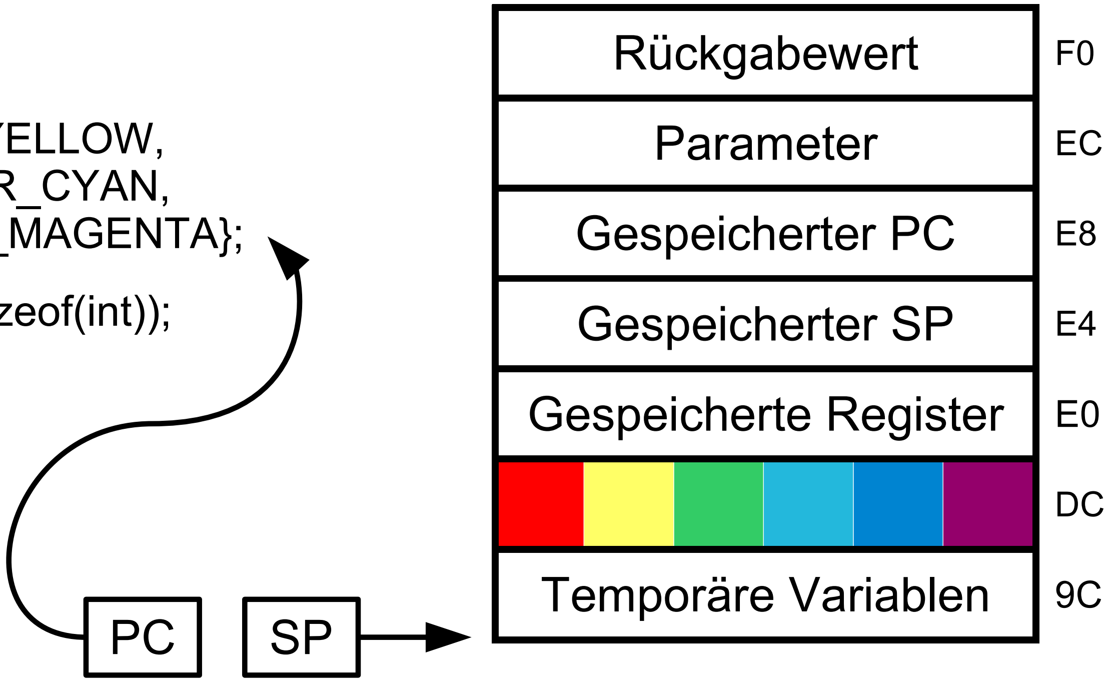
Rainbow revisited

```
int main(void){

    int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                    COLOR_GREEN, COLOR_CYAN,
                    COLOR_BLUE, COLOR_MAGENTA};

    printRainbow(rainbow, sizeof(rainbow)/sizeof(int));
}
```



```
void printRainbow(int rainbow[], int rainbowLength){

     for(int i= 0; i<rainbowLength; i++){
         attron(COLOR_PAIR(rainbow[i]));
         addstr("\u2588");
         attroff(COLOR_PAIR(rainbow[i]));
     }
}
```

```
int main(void){

    int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                        COLOR_GREEN, COLOR_CYAN,
                         COLOR_BLUE, COLOR_MAGENTA};

    printRainbow(rainbow, sizeof(rainbow)/sizeof(int));

}
```

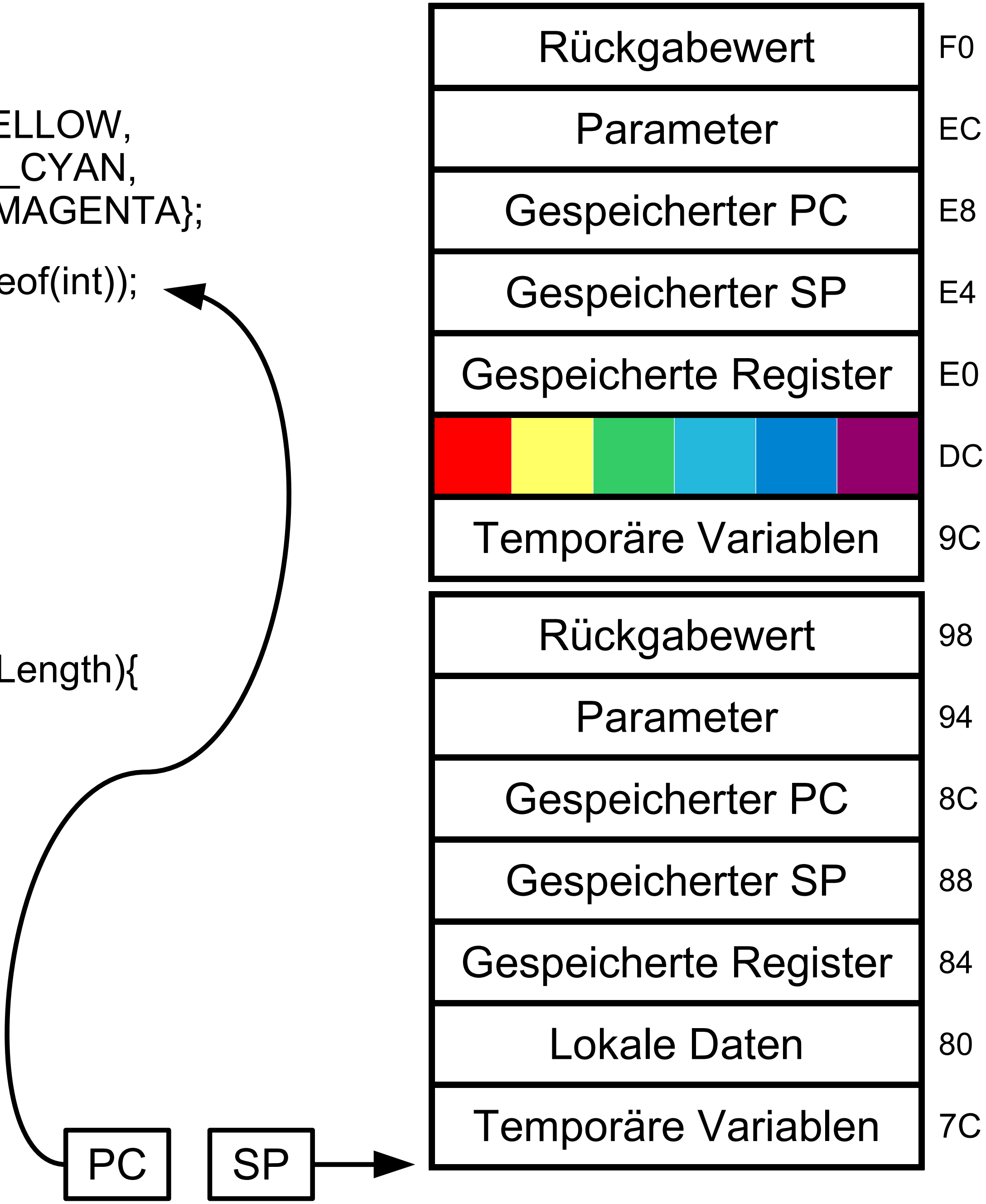


Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
🟥🟨🟩🟦🟦🟪	DC
Temporäre Variablen	9C

```
int main(void){
void printRainbow(int rainbow[], int rainbowLength){

     for(int i= 0; i<rainbowLength; i++){
         attron(COLOR_PAIR(rainbow[i]));
         addstr("\u2588");
         attroff(COLOR_PAIR(rainbow[i]));
     }
}
```

Rainbow revisited

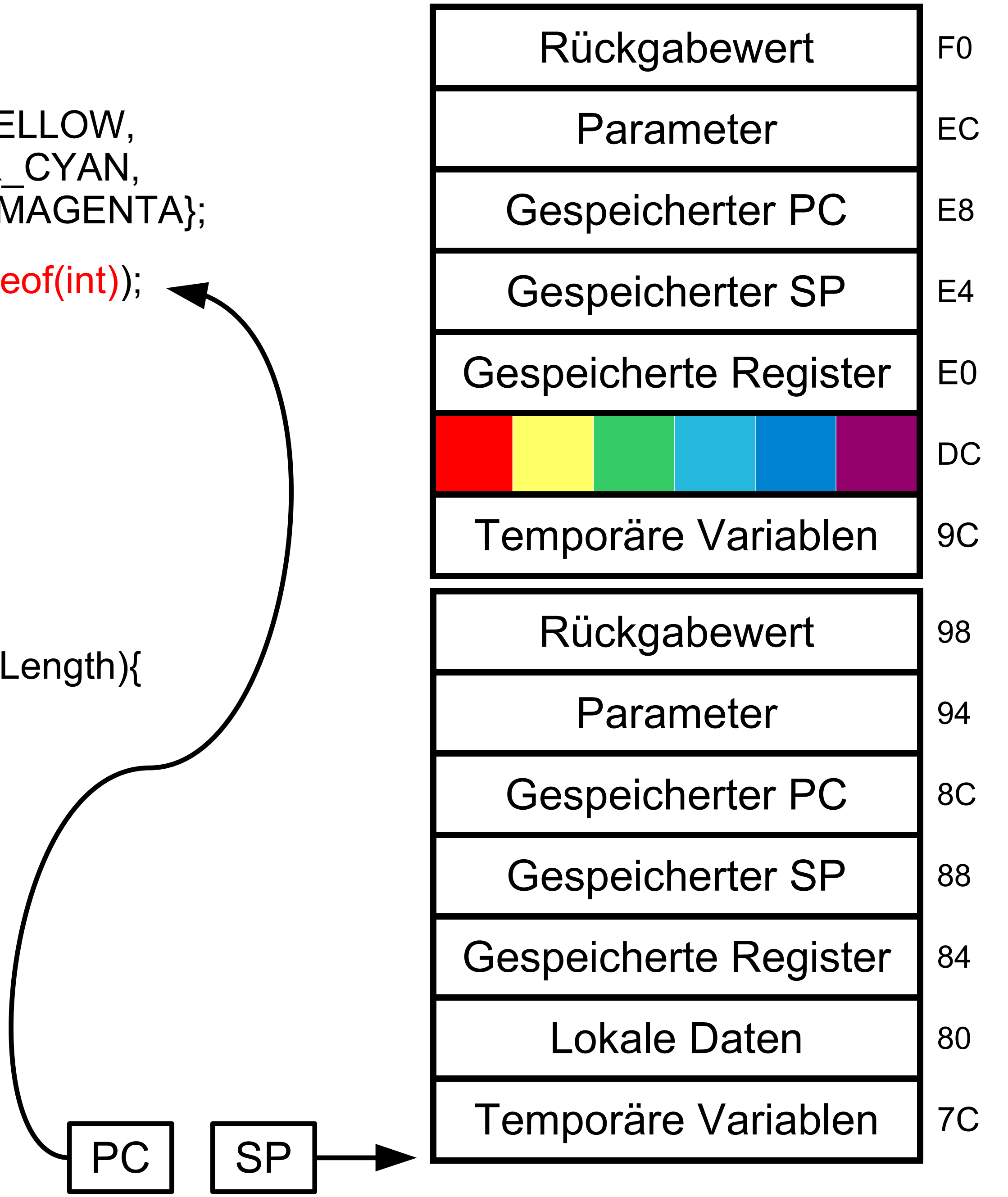
```
int main(void){

    int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                    COLOR_GREEN, COLOR_CYAN,
                    COLOR_BLUE, COLOR_MAGENTA};

    printRainbow(rainbow, sizeof(rainbow)/sizeof(int));

}



void printRainbow(int rainbow[], int rainbowLength){

     for(int i= 0; i<rainbowLength; i++){

        attron(COLOR_PAIR(rainbow[i]));

        addstr("\u2588");

        attroff(COLOR_PAIR(rainbow[i]));

    }

}
```



Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
🟥🟨🟩🟦🟦🟪	DC
Temporäre Variablen	9C
Rückgabewert	98
Parameter	94
Gespeicherter PC	8C
Gespeicherter SP	88
Gespeicherte Register	84
Lokale Daten	80
Temporäre Variablen	7C

Rainbow revisited

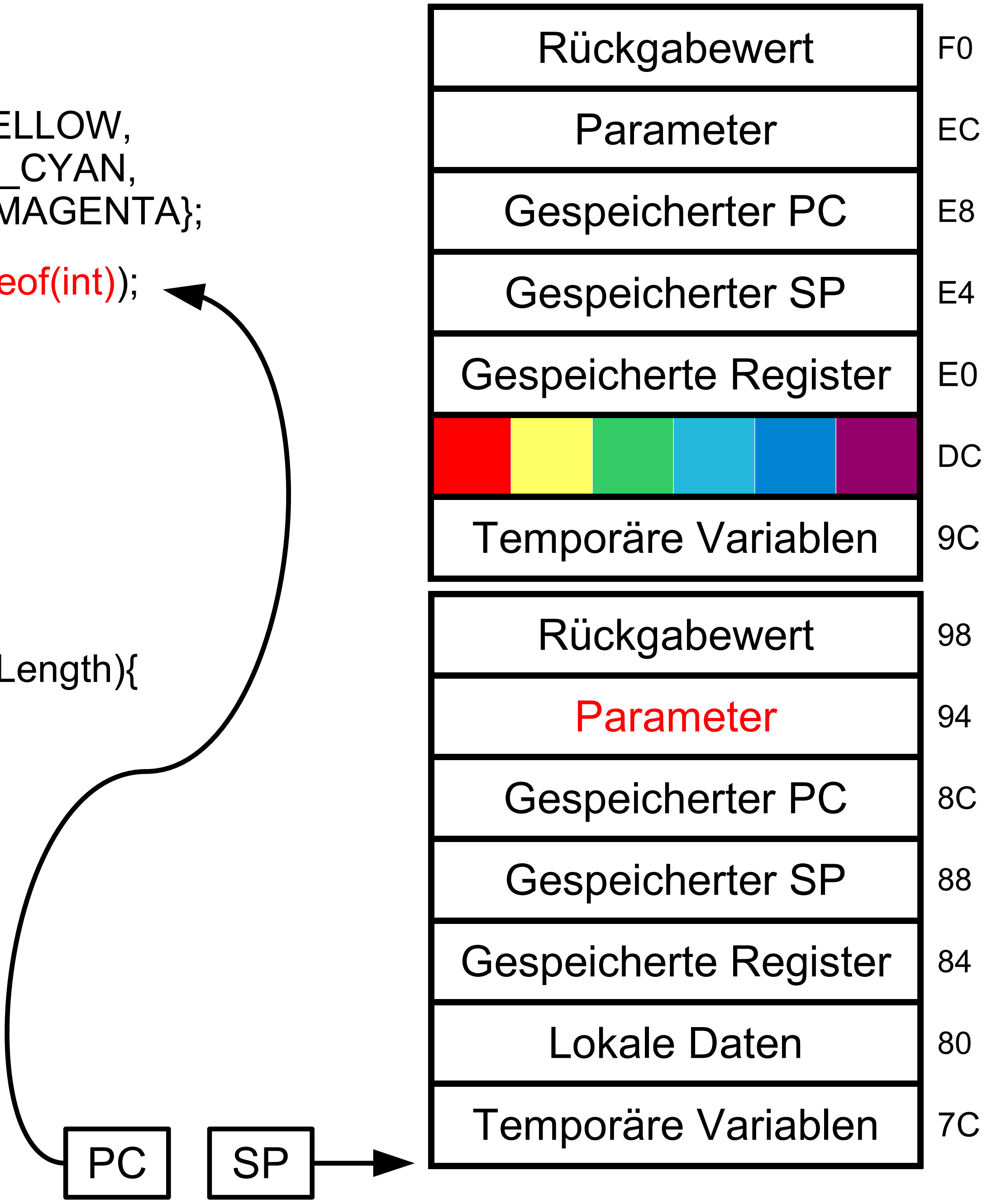
```
int main(void){

    int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                     COLOR_GREEN, COLOR_CYAN,
                     COLOR_BLUE, COLOR_MAGENTA};

    printRainbow(rainbow, sizeof(rainbow)/sizeof(int));
}



int main(void){
void printRainbow(int rainbow[], int rainbowLength){

     for(int i= 0; i<rainbowLength; i++){
         attron(COLOR_PAIR(rainbow[i]));
         addstr("\u2588");
         attroff(COLOR_PAIR(rainbow[i]));
     }
}
```



Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
🟥🟨🟩🟦🟦🟪	DC
Temporäre Variablen	9C
Rückgabewert	98
Parameter	94
Gespeicherter PC	8C
Gespeicherter SP	88
Gespeicherte Register	84
Lokale Daten	80
Temporäre Variablen	7C

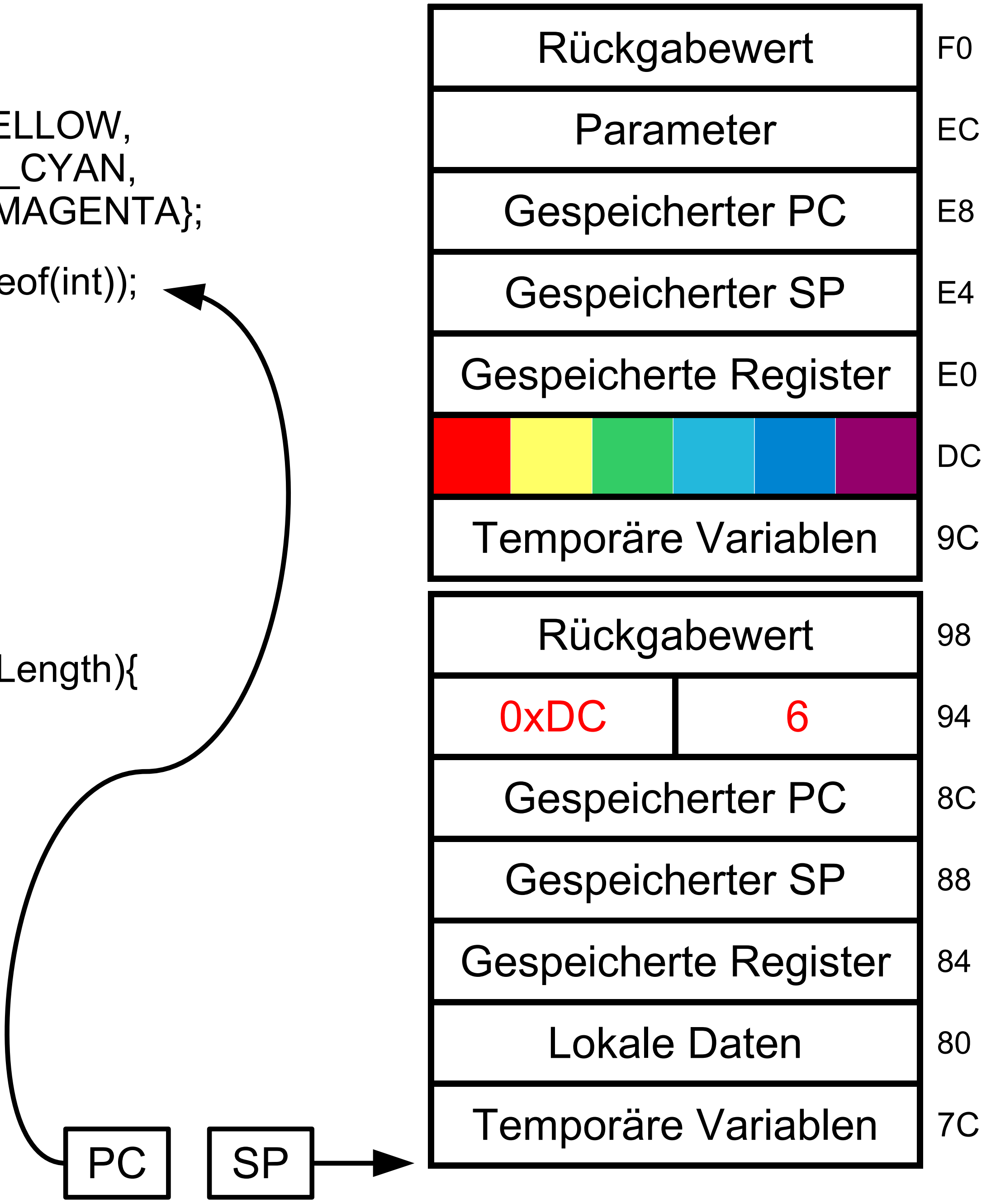
Rainbow revisited

```
int main(void){

    int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                    COLOR_GREEN, COLOR_CYAN,
                    COLOR_BLUE, COLOR_MAGENTA};

    printRainbow(rainbow, sizeof(rainbow)/sizeof(int));
}


void printRainbow(int rainbow[], int rainbowLength){

     for(int i= 0; i<rainbowLength; i++){
         attron(COLOR_PAIR(rainbow[i]));
         addstr("\u2588");
         attroff(COLOR_PAIR(rainbow[i]));
     }
}
```



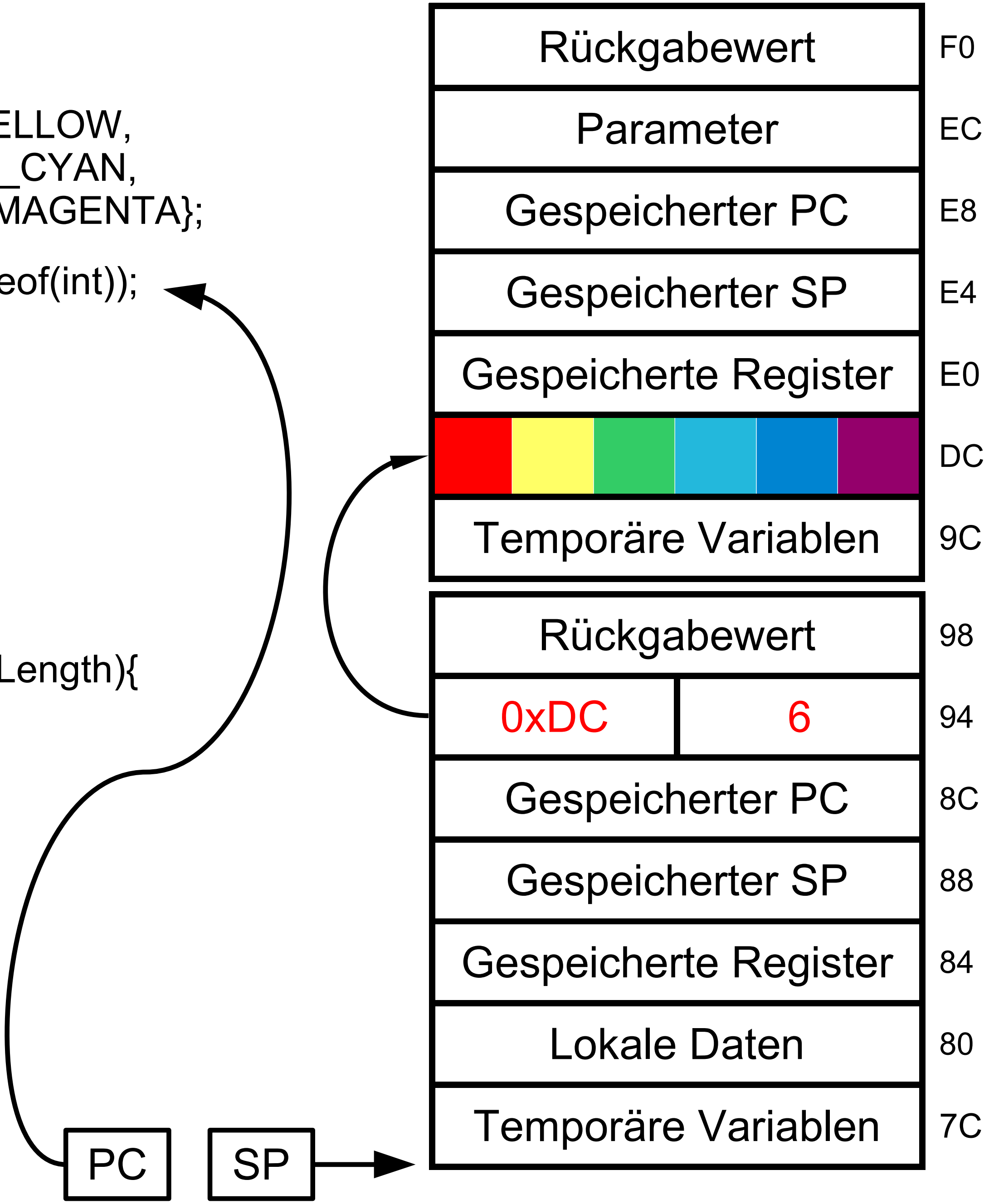
Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
🟥🟨🟩🟦🟦🟪	DC
Temporäre Variablen	9C
Rückgabewert	98
Parameter	94
Gespeicherter PC	8C
Gespeicherter SP	88
Gespeicherte Register	84
Lokale Daten	80
Temporäre Variablen	7C

```
int main(void){

    int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                    COLOR_GREEN, COLOR_CYAN,
                    COLOR_BLUE, COLOR_MAGENTA};

    printRainbow(rainbow, sizeof(rainbow)/sizeof(int));

}


void printRainbow(int rainbow[], int rainbowLength){

     for(int i= 0; i<rainbowLength; i++){
        attron(COLOR_PAIR(rainbow[i]));
        addstr("\u2588");
        attroff(COLOR_PAIR(rainbow[i]));
    }
}
```



Rainbow revisited

```
int main(void){

    int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                        COLOR_GREEN, COLOR_CYAN,
                         COLOR_BLUE, COLOR_MAGENTA};

    printRainbow(rainbow, sizeof(rainbow)/sizeof(int));
}

void printRainbow(int rainbow[], int rainbowLength){

     for(int i= 0; i<rainbowLength; i++){
         attron(COLOR_PAIR(rainbow[i]));
         addstr("\u2588");
         attroff(COLOR_PAIR(rainbow[i]));
    }
}
```



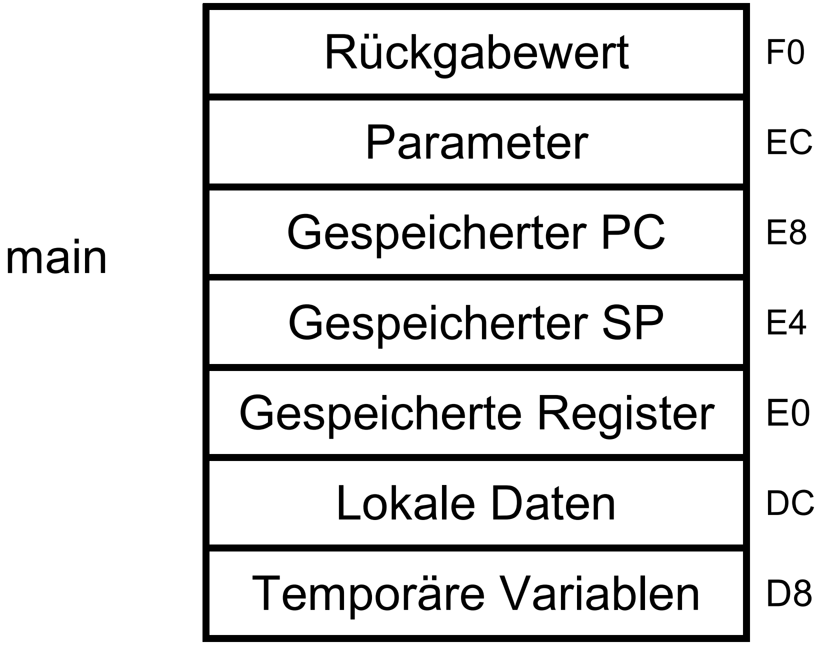
Rückgabewert		F0
Parameter		EC
Gespeicherter PC		E8
Gespeicherter SP		E4
Gespeicherte Register		E0
🟥 🟨 🟩 🟦 🟦 🟪		DC
Temporäre Variablen		9C
Rückgabewert		98
0xDC	6	94
Gespeicherter PC		8C
Gespeicherter SP		88
Gespeicherte Register		84
Lokale Daten		80
Temporäre Variablen		7C

Rainbow revisited

```
int main(){

    int rainbow[]= makeRainbow();
    printRainbow(rainbow, 6);
}

int *makeRainbow(){

     int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                     COLOR_GREEN, COLOR_CYAN,
                     COLOR_BLUE, COLOR_MAGENTA};
   return rainbow;
}

void printRainbow(int rainbow[], int rainbowLength){

     for(int i= 0; i<rainbowLength; i++){
          attron(COLOR_PAIR(rainbow[i]));
          addstr("\u2588");
          attroff(COLOR_PAIR(rainbow[i]));
    }
}
```

Warum geht das schief?

```
int main(){

   int rainbow[]= makeRainbow();
   printRainbow(rainbow, 6);
}

int *makeRainbow(){

    int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                    COLOR_GREEN, COLOR_CYAN,
                    COLOR_BLUE, COLOR_MAGENTA};
  return rainbow;
}

void printRainbow(int rainbow[], int rainbowLength){

    for(int i= 0; i<rainbowLength; i++){
        attron(COLOR_PAIR(rainbow[i]));
        addstr("\u2588");
        attroff(COLOR_PAIR(rainbow[i]));
   }
}
```

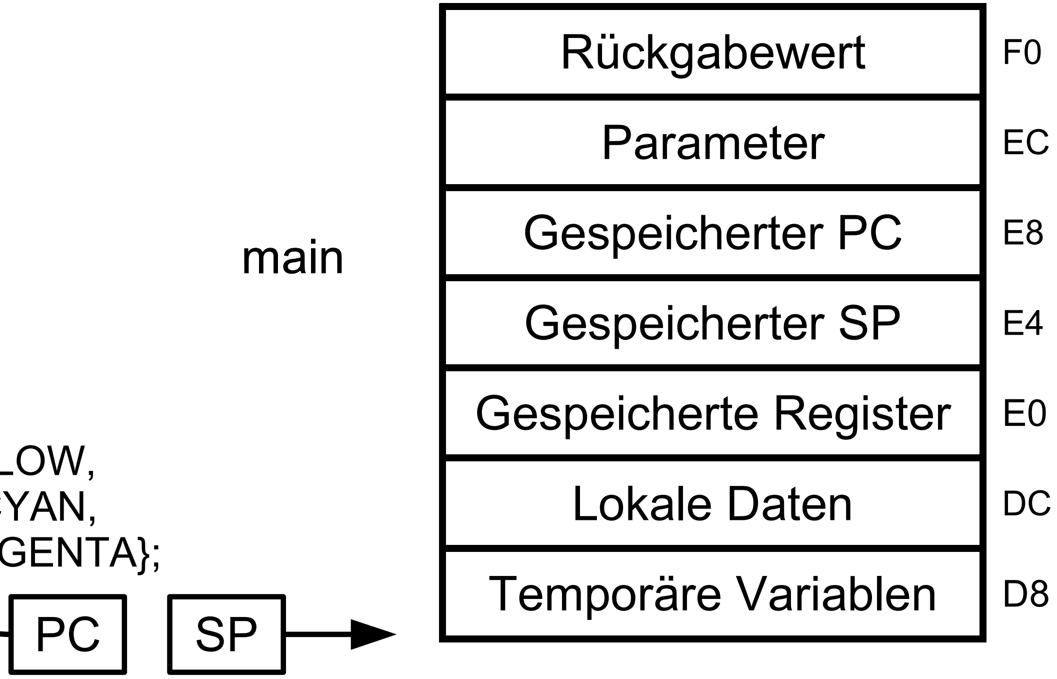


main

Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
Lokale Daten	DC
Temporäre Variablen	D8

Warum geht das schief?

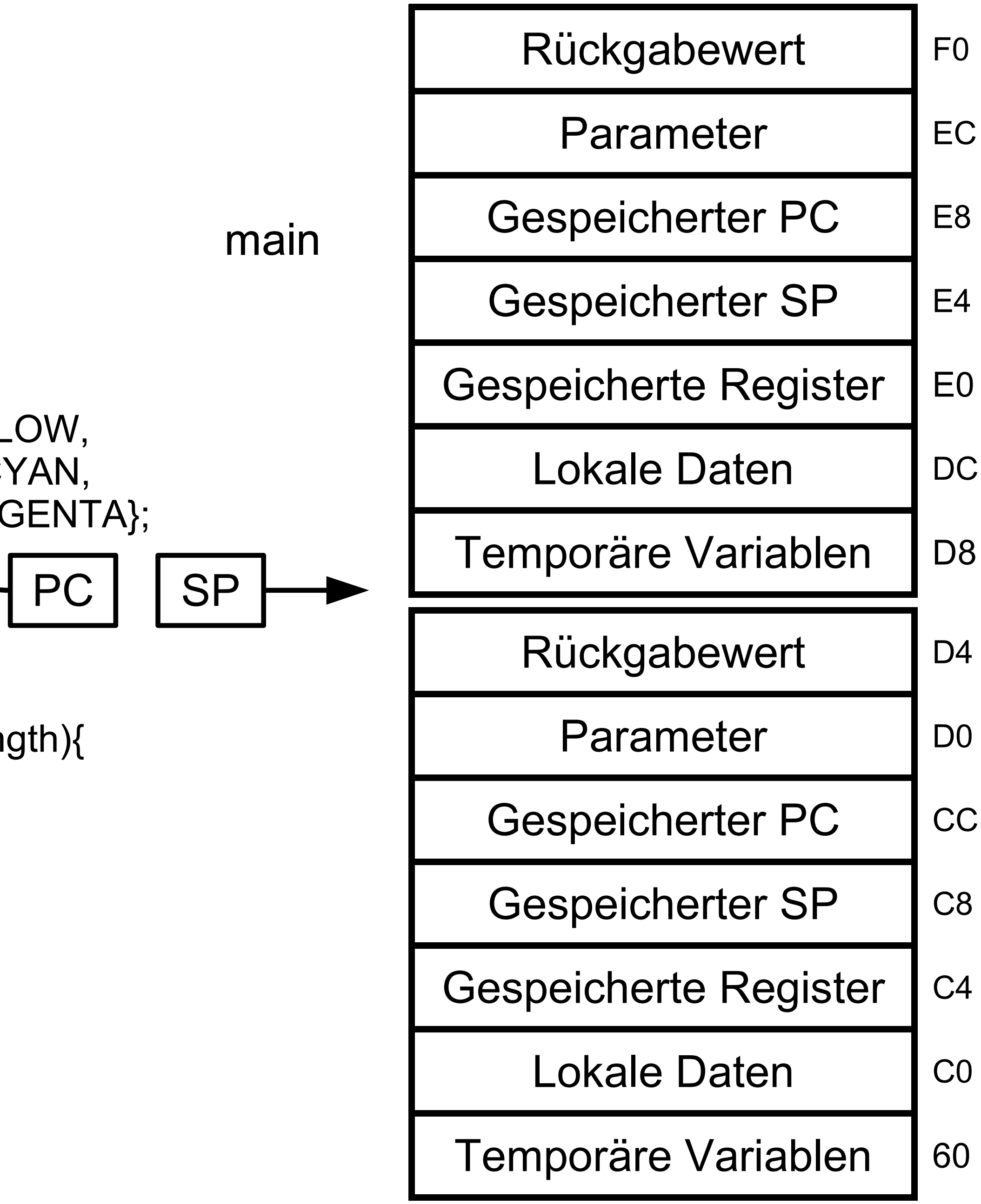
```
int main(){

    int rainbow[]= makeRainbow();
    printRainbow(rainbow, 6);
}

int *makeRainbow(){

     int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                     COLOR_GREEN, COLOR_CYAN,
                     COLOR_BLUE, COLOR_MAGENTA};

   return rainbow;
}

void printRainbow(int rainbow[], int rainbowLength){

     for(int i= 0; i<rainbowLength; i++){
          attron(COLOR_PAIR(rainbow[i]));
          addstr("\u2588");
          attroff(COLOR_PAIR(rainbow[i]));
    }
}
```



Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
Lokale Daten	DC
Temporäre Variablen	D8

Warum geht das schief?

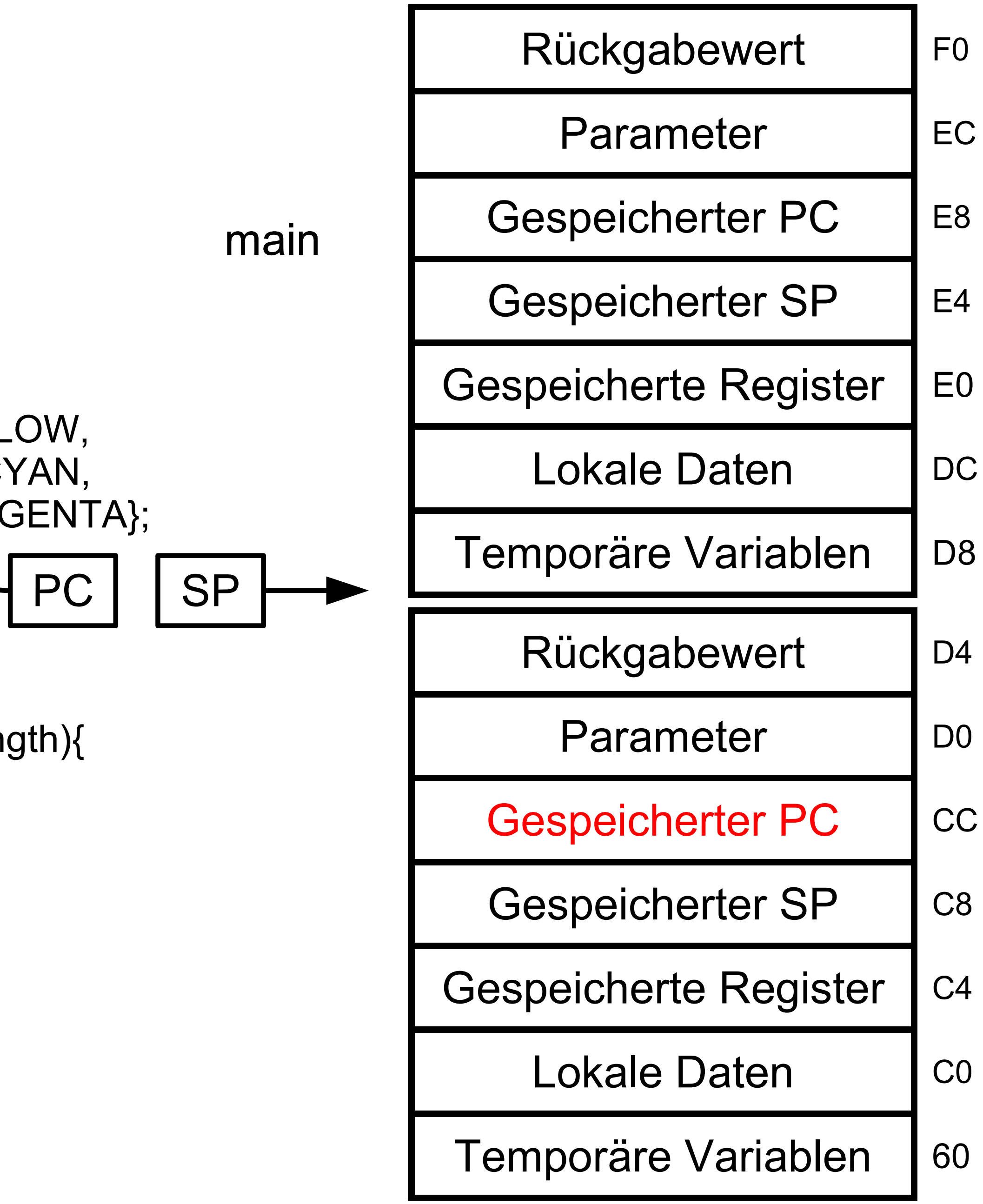
```
int main(){

    int rainbow[]= makeRainbow();
    printRainbow(rainbow, 6);
}

int *makeRainbow(){

     int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                     COLOR_GREEN, COLOR_CYAN,
                     COLOR_BLUE, COLOR_MAGENTA};

   return rainbow;
}

void printRainbow(int rainbow[], int rainbowLength){

     for(int i= 0; i<rainbowLength; i++){
         attron(COLOR_PAIR(rainbow[i]));
         addstr("\u2588");
         attroff(COLOR_PAIR(rainbow[i]));
    }
}
```



Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
Lokale Daten	DC
Temporäre Variablen	D8
Rückgabewert	D4
Parameter	D0
Gespeicherter PC	CC
Gespeicherter SP	C8
Gespeicherte Register	C4
Lokale Daten	C0
Temporäre Variablen	60

Warum geht das schief?

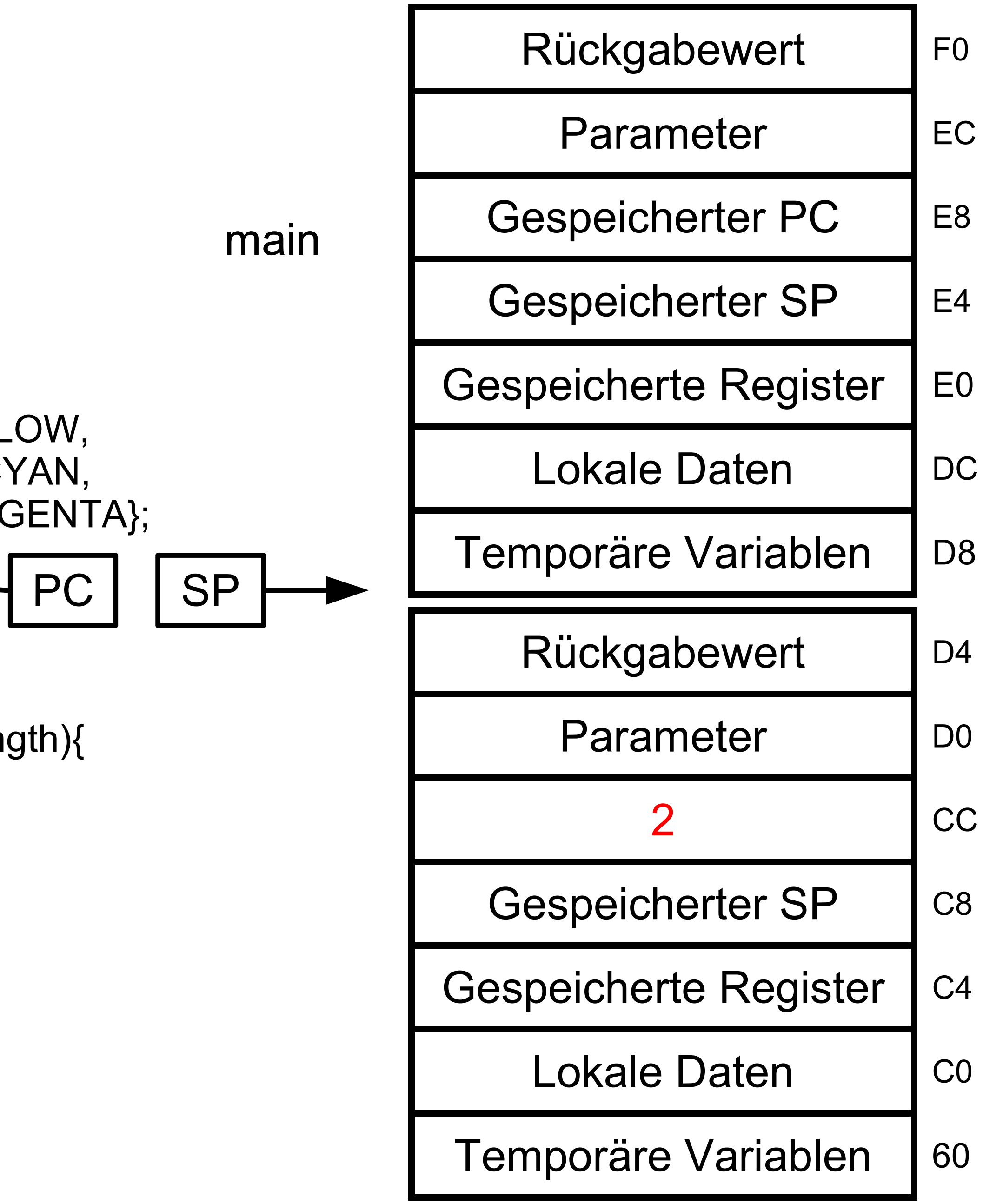
```
1   int main(){

2       int rainbow[]= makeRainbow();
3       printRainbow(rainbow, 6);
    }

4   int *makeRainbow(){

5        int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                         COLOR_GREEN, COLOR_CYAN,
                         COLOR_BLUE, COLOR_MAGENTA};

6      return rainbow;
    }

7   void printRainbow(int rainbow[], int rainbowLength){

8        for(int i= 0; i<rainbowLength; i++){
9            attron(COLOR_PAIR(rainbow[i]));
10           addstr("\u2588");
11           attroff(COLOR_PAIR(rainbow[i]));
        }
   }
```



Rückgabewert F0
Parameter EC
Gespeicherter PC E8
Gespeicherter SP E4
Gespeicherte Register E0
Lokale Daten DC
Temporäre Variablen D8
Rückgabewert D4
Parameter D0
Gespeicherter PC CC
Gespeicherter SP C8
Gespeicherte Register C4
Lokale Daten C0
Temporäre Variablen 60

Warum geht das schief?

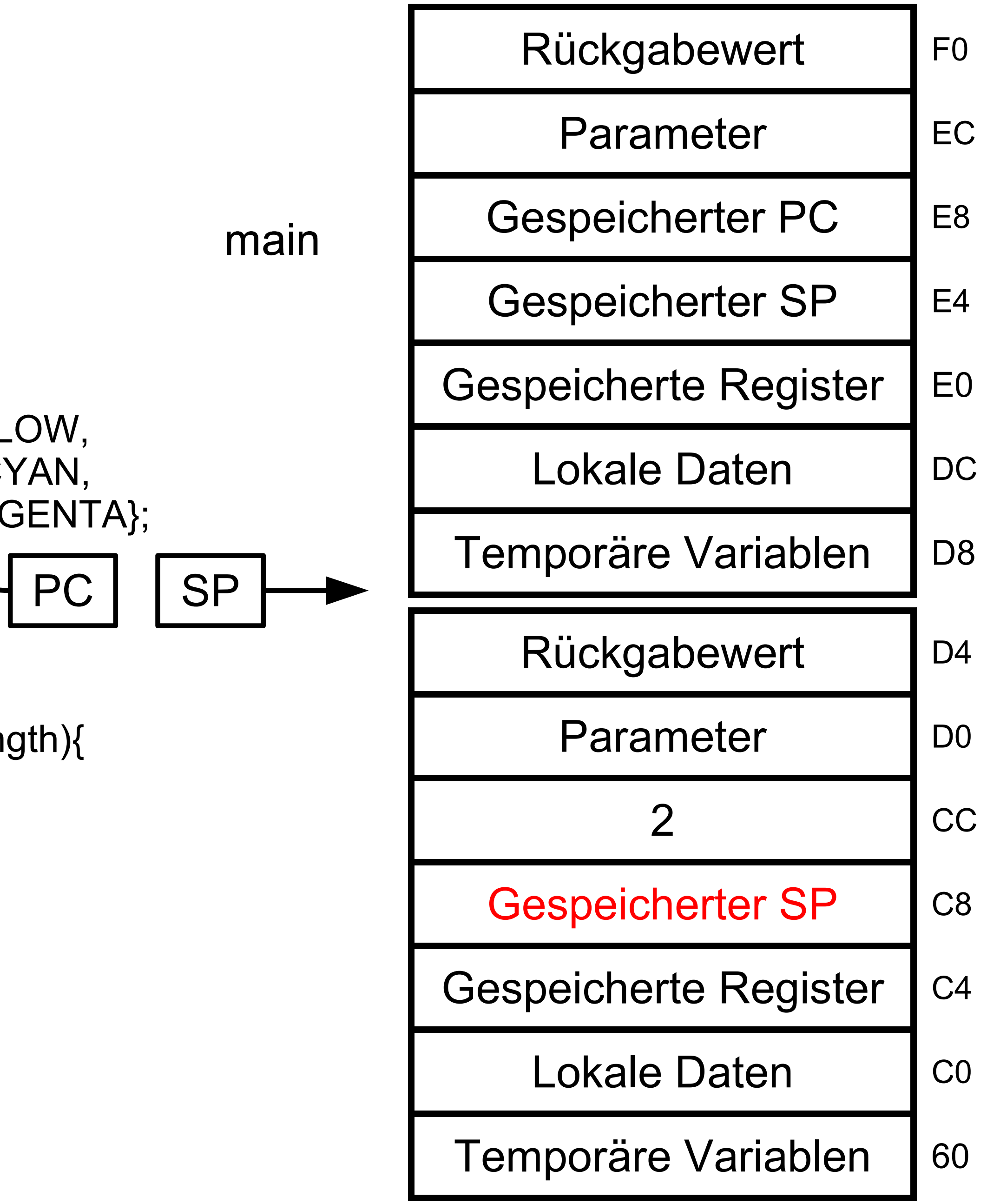
```
int main(){

    int rainbow[]= makeRainbow();
    printRainbow(rainbow, 6);
}

int *makeRainbow(){

     int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                     COLOR_GREEN, COLOR_CYAN,
                     COLOR_BLUE, COLOR_MAGENTA};

   return rainbow;
}

void printRainbow(int rainbow[], int rainbowLength){

     for(int i= 0; i<rainbowLength; i++){
          attron(COLOR_PAIR(rainbow[i]));
          addstr("\u2588");
          attroff(COLOR_PAIR(rainbow[i]));
    }
}
```



Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
Lokale Daten	DC
Temporäre Variablen	D8
Rückgabewert	D4
Parameter	D0
2	CC
Gespeicherter SP	C8
Gespeicherte Register	C4
Lokale Daten	C0
Temporäre Variablen	60

Warum geht das schief?

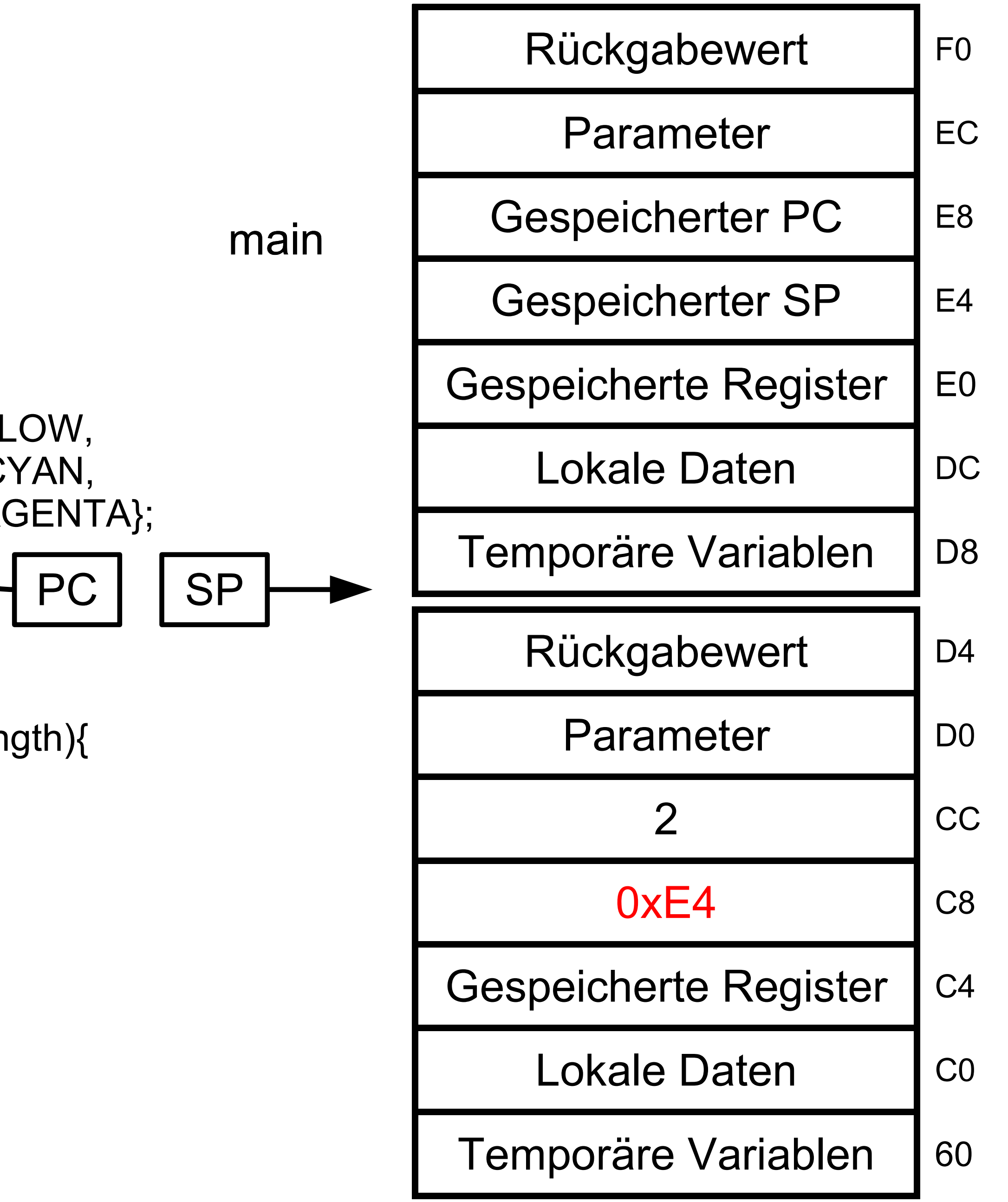
```
1   int main(){

2       int rainbow[]= makeRainbow();
3       printRainbow(rainbow, 6);
    }

4   int *makeRainbow(){

5        int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                         COLOR_GREEN, COLOR_CYAN,
                         COLOR_BLUE, COLOR_MAGENTA};
6       return rainbow;
    }

7   void printRainbow(int rainbow[], int rainbowLength){

8        for(int i= 0; i<rainbowLength; i++){
9            attron(COLOR_PAIR(rainbow[i]));
10           addstr("\u2588");
11           attroff(COLOR_PAIR(rainbow[i]));
         }
    }
```



Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
Lokale Daten	DC
Temporäre Variablen	D8
Rückgabewert	D4
Parameter	D0
2	CC
Gespeicherter SP	C8
Gespeicherte Register	C4
Lokale Daten	C0
Temporäre Variablen	60

Warum geht das schief?

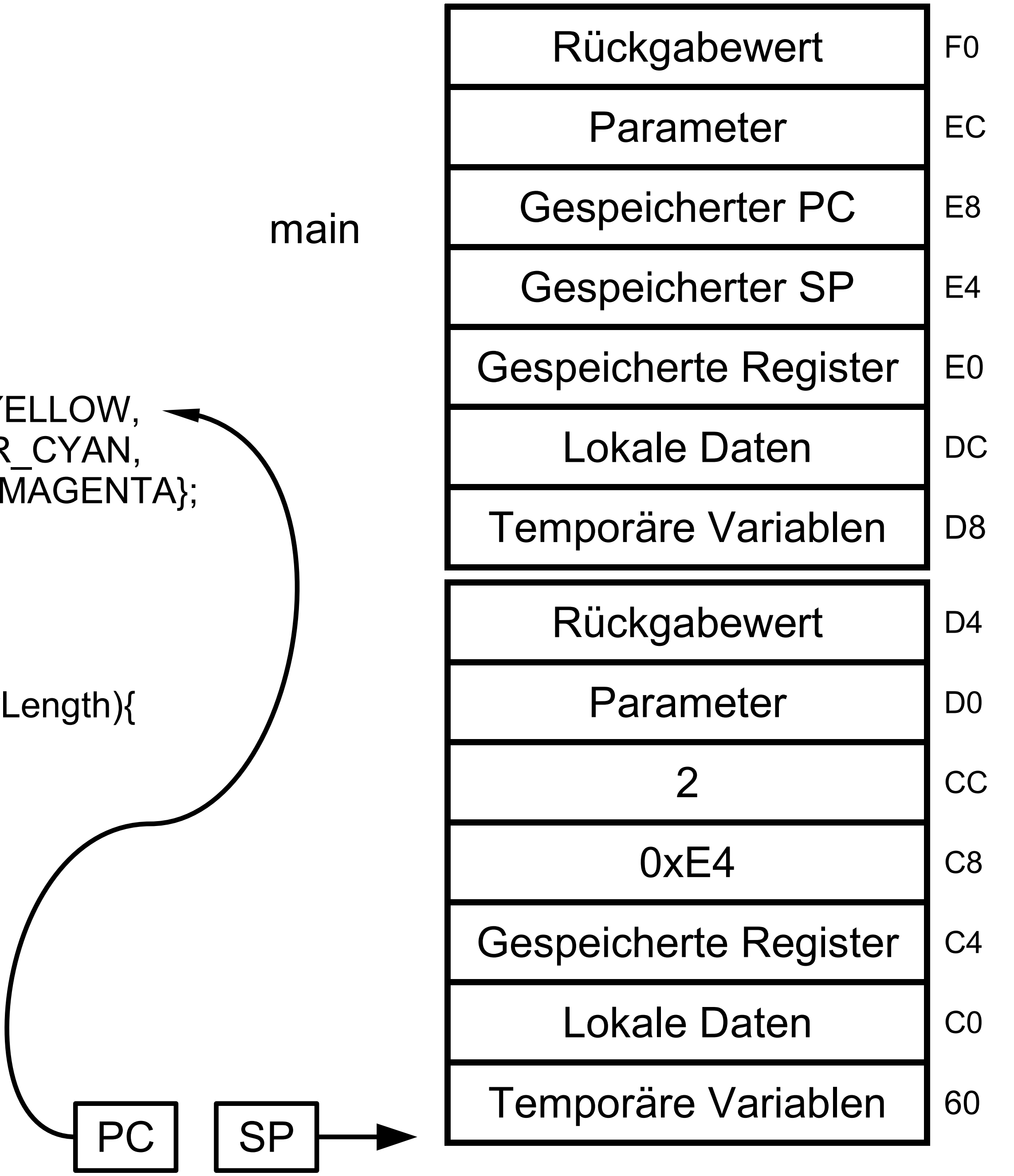
```
int main(){

    int rainbow[]= makeRainbow();
    printRainbow(rainbow, 6);
}

int *makeRainbow(){

     int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                     COLOR_GREEN, COLOR_CYAN,
                     COLOR_BLUE, COLOR_MAGENTA};

    return rainbow;
}

void printRainbow(int rainbow[], int rainbowLength){

     for(int i= 0; i<rainbowLength; i++){
         attron(COLOR_PAIR(rainbow[i]));
         addstr("\u2588");
         attroff(COLOR_PAIR(rainbow[i]));
    }
}
```



Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
Lokale Daten	DC
Temporäre Variablen	D8
Rückgabewert	D4
Parameter	D0
2	CC
0xE4	C8
Gespeicherte Register	C4
Lokale Daten	C0
Temporäre Variablen	60

Warum geht das schief?

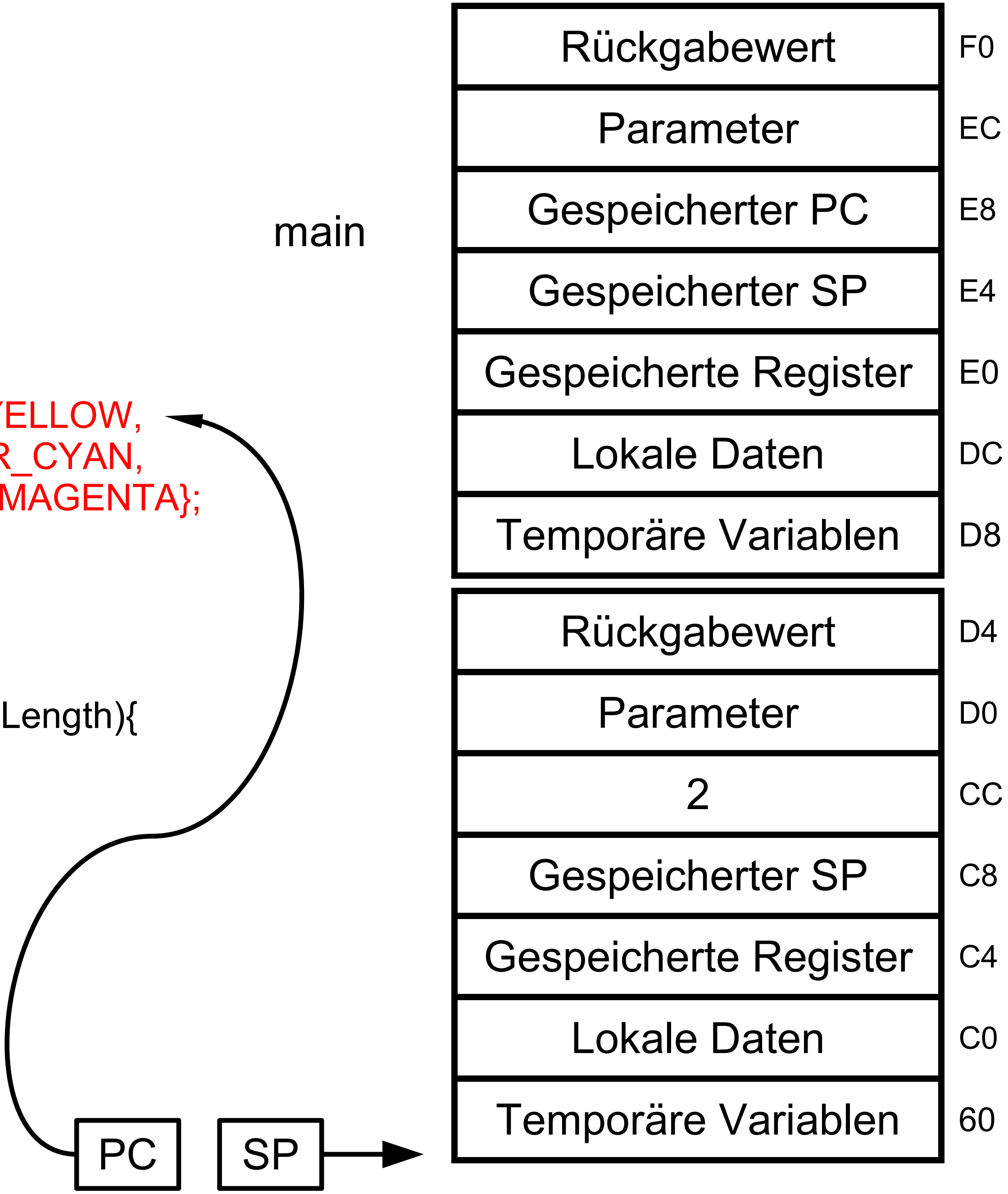
```
1   int main(){

2       int rainbow[]= makeRainbow();
3       printRainbow(rainbow, 6);
    }

4   int *makeRainbow(){

5        int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                         COLOR_GREEN, COLOR_CYAN,
                         COLOR_BLUE, COLOR_MAGENTA};
6      return rainbow;
    }

7   void printRainbow(int rainbow[], int rainbowLength){

8        for(int i= 0; i<rainbowLength; i++){
9            attron(COLOR_PAIR(rainbow[i]));
10           addstr("\u2588");
11           attroff(COLOR_PAIR(rainbow[i]));
         }
    }
```



Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
Lokale Daten	DC
Temporäre Variablen	D8
Rückgabewert	D4
Parameter	D0
2	CC
0xE4	C8
Gespeicherte Register	C4
Lokale Daten	C0
Temporäre Variablen	60

Warum geht das schief?

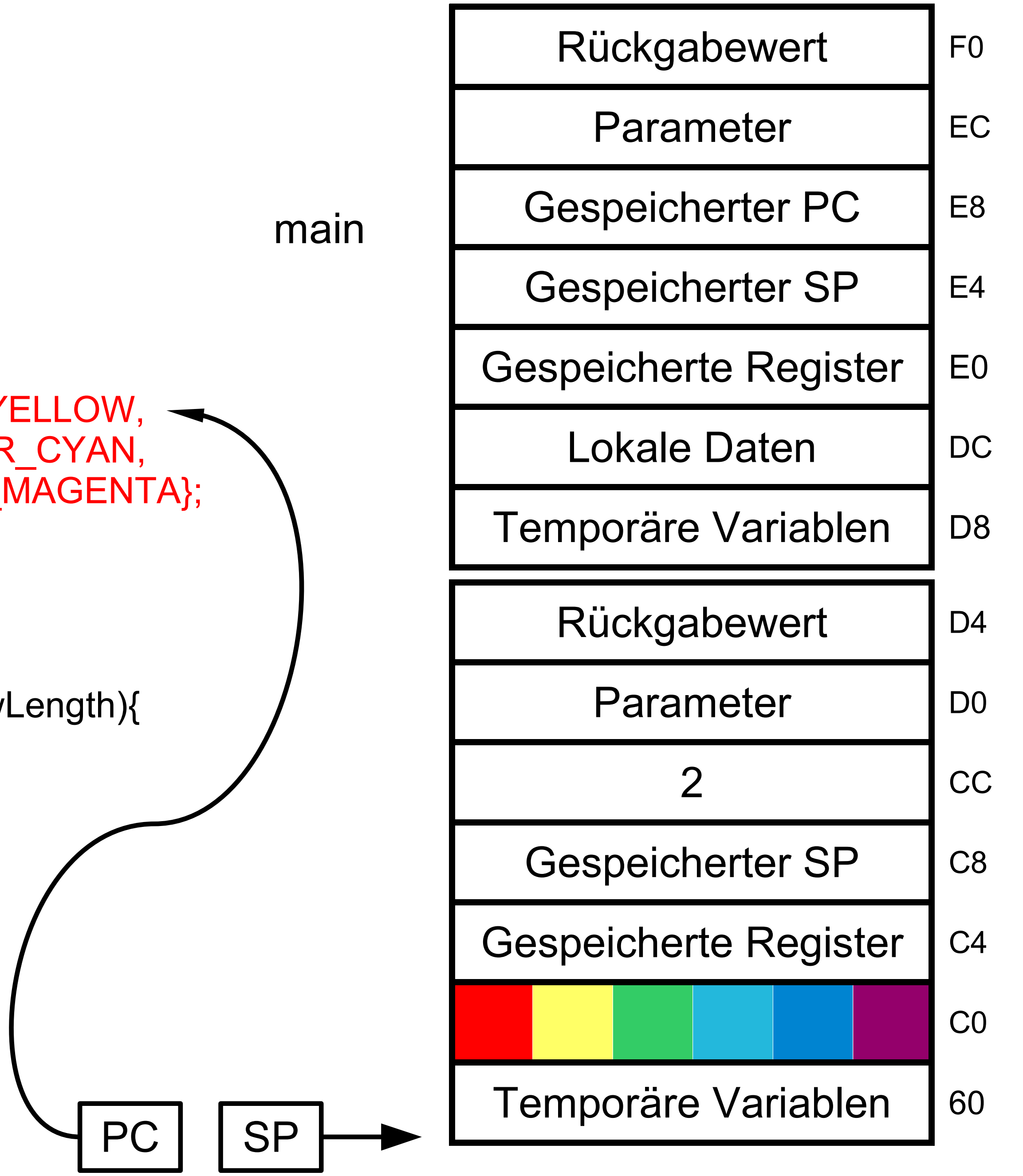
```
1   int main(){

2       int rainbow[]= makeRainbow();
3       printRainbow(rainbow, 6);
    }

4   int *makeRainbow(){

5        int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                         COLOR_GREEN, COLOR_CYAN,
                         COLOR_BLUE, COLOR_MAGENTA};

6      return rainbow;
    }

7   void printRainbow(int rainbow[], int rainbowLength){

8        for(int i= 0; i<rainbowLength; i++){
9            attron(COLOR_PAIR(rainbow[i]));
10           addstr("\u2588");
11           attroff(COLOR_PAIR(rainbow[i]));
         }
    }
```



Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
Lokale Daten	DC
Temporäre Variablen	D8
Rückgabewert	D4
Parameter	D0
2	CC
Gespeicherter SP	C8
Gespeicherte Register	C4
Lokale Daten	C0
Temporäre Variablen	60

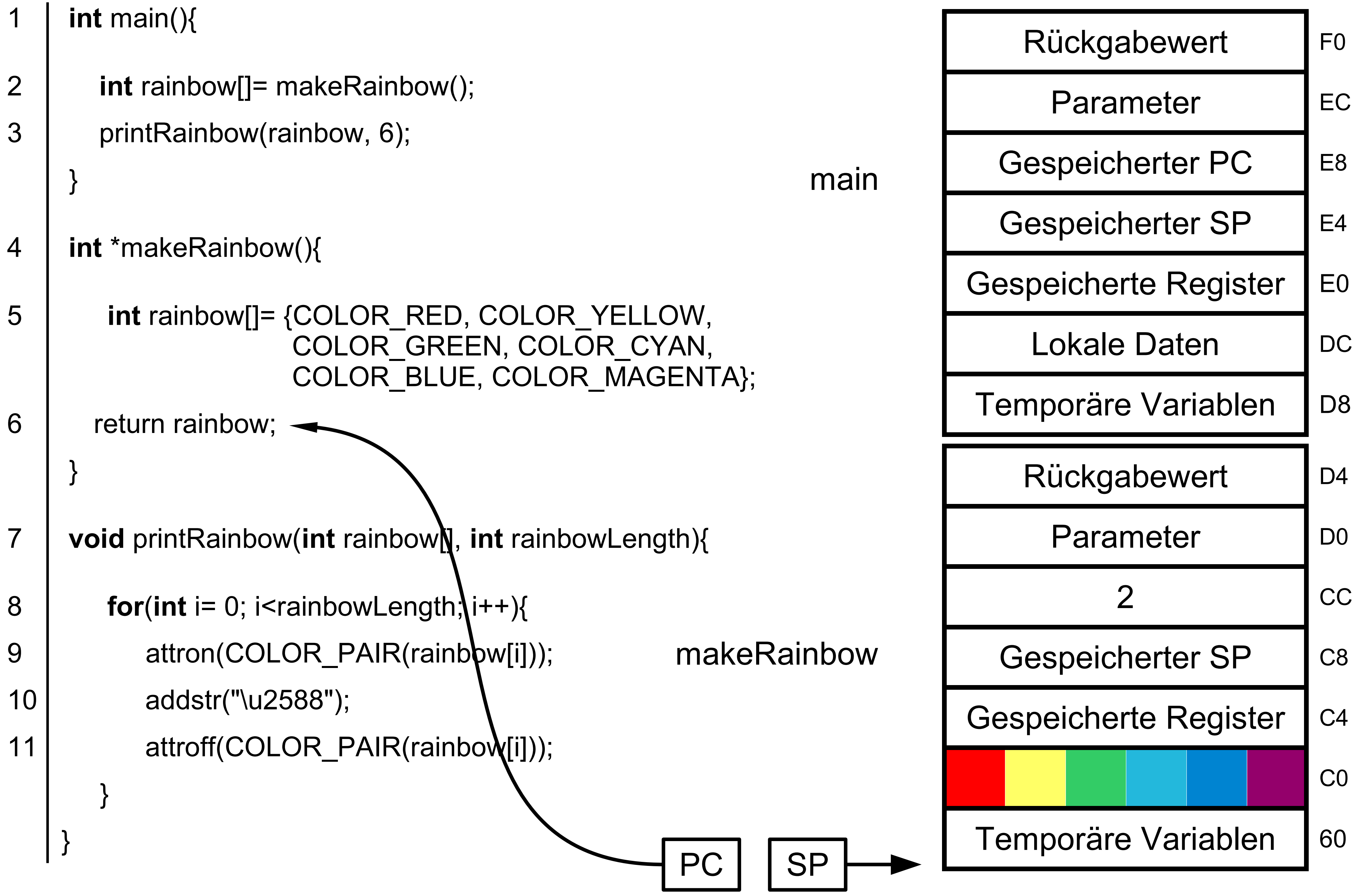
Warum geht das schief?

```
int main(){

    int rainbow[]= makeRainbow();
    printRainbow(rainbow, 6);
}

int *makeRainbow(){

     int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                     COLOR_GREEN, COLOR_CYAN,
                     COLOR_BLUE, COLOR_MAGENTA};

   return rainbow;
}

void printRainbow(int rainbow[], int rainbowLength){

     for(int i= 0; i<rainbowLength; i++){
         attron(COLOR_PAIR(rainbow[i]));
         addstr("\u2588");
         attroff(COLOR_PAIR(rainbow[i]));
    }
}
```



Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
Lokale Daten	DC
Temporäre Variablen	D8
Rückgabewert	D4
Parameter	D0
2	CC
Gespeicherter SP	C8
Gespeicherte Register	C4
🟥🟨🟩🟦🟦🟪	C0
Temporäre Variablen	60

Warum geht das schief?

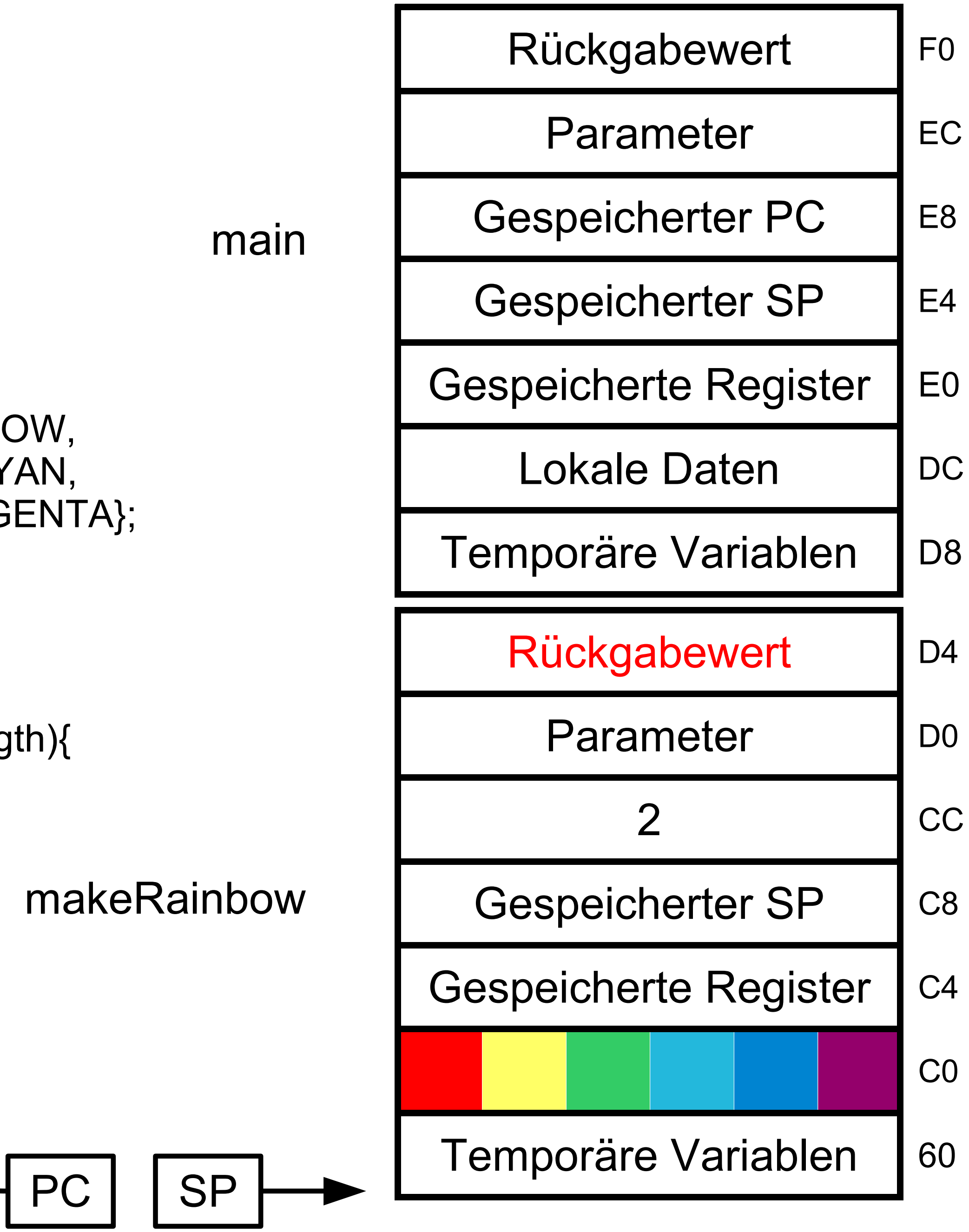


```
int main(){

    int rainbow[]= makeRainbow();
    printRainbow(rainbow, 6);
}

int *makeRainbow(){

     int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                     COLOR_GREEN, COLOR_CYAN,
                     COLOR_BLUE, COLOR_MAGENTA};

   return rainbow;
}

void printRainbow(int rainbow[], int rainbowLength){

     for(int i= 0; i<rainbowLength; i++){
          attron(COLOR_PAIR(rainbow[i]));
          addstr("\u2588");
          attroff(COLOR_PAIR(rainbow[i]));
     }
}
```

main	Rückgabewert	F0
	Parameter	EC
	Gespeicherter PC	E8
	Gespeicherter SP	E4
	Gespeicherte Register	E0
	Lokale Daten	DC
	Temporäre Variablen	D8
makeRainbow	Rückgabewert	D4
	Parameter	D0
	2	CC
	Gespeicherter SP	C8
	Gespeicherte Register	C4
	🟥 🟨 🟩 🟦 🟦 🟪	C0
	Temporäre Variablen	60

Warum geht das schief?

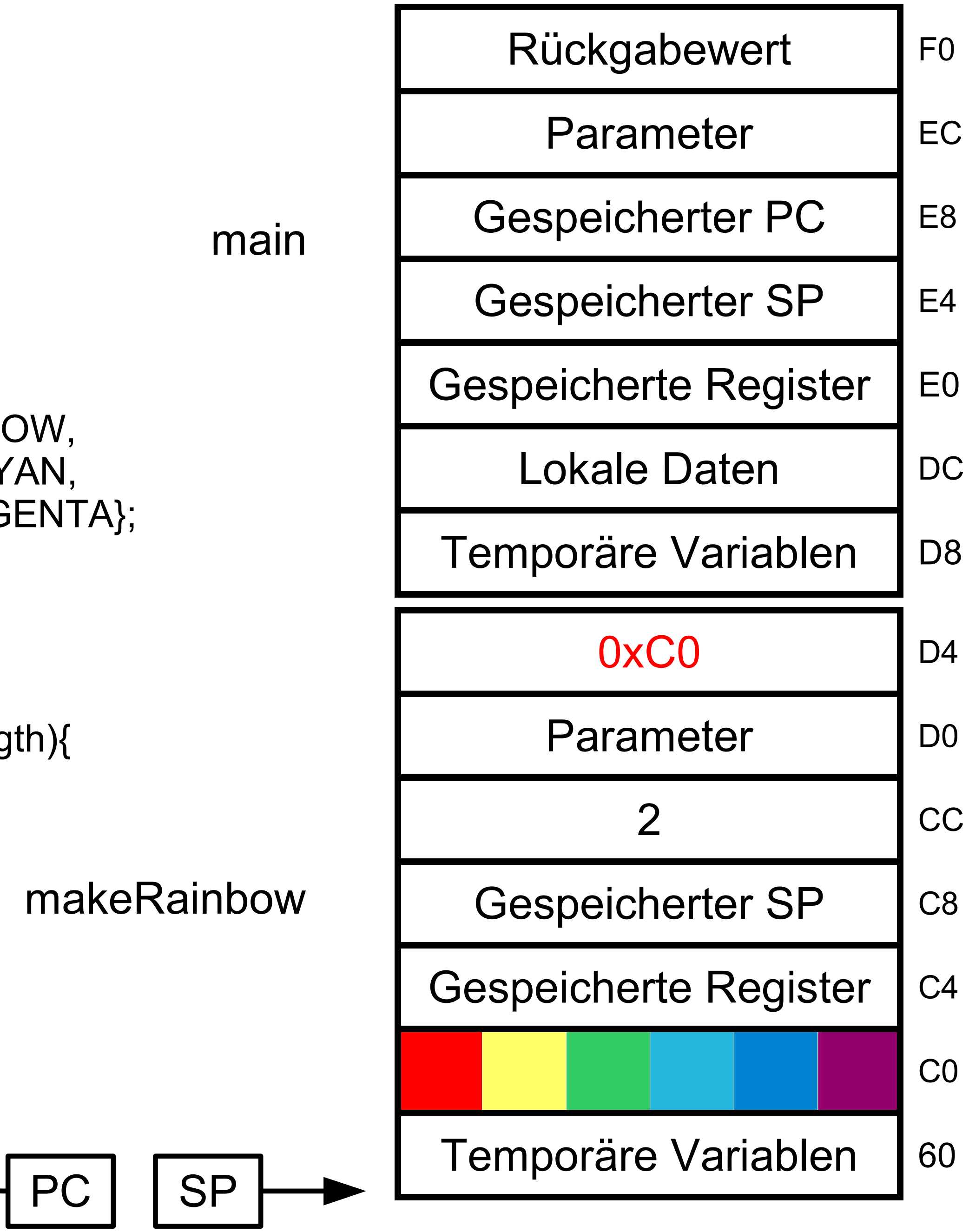
```
int main(){

    int rainbow[]= makeRainbow();
    printRainbow(rainbow, 6);
}

int *makeRainbow(){

     int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                     COLOR_GREEN, COLOR_CYAN,
                     COLOR_BLUE, COLOR_MAGENTA};

   return rainbow;
}

void printRainbow(int rainbow[], int rainbowLength){

     for(int i= 0; i<rainbowLength; i++){
         attron(COLOR_PAIR(rainbow[i]));
         addstr("\u2588");
         attroff(COLOR_PAIR(rainbow[i]));
    }
}
```



main	Rückgabewert	F0
	Parameter	EC
	Gespeicherter PC	E8
	Gespeicherter SP	E4
	Gespeicherte Register	E0
	Lokale Daten	DC
	Temporäre Variablen	D8
makeRainbow	Rückgabewert	D4
	Parameter	D0
	2	CC
	Gespeicherter SP	C8
	Gespeicherte Register	C4
	🟥🟨🟩🟦🟦🟪	C0
	Temporäre Variablen	60

Warum geht das schief?

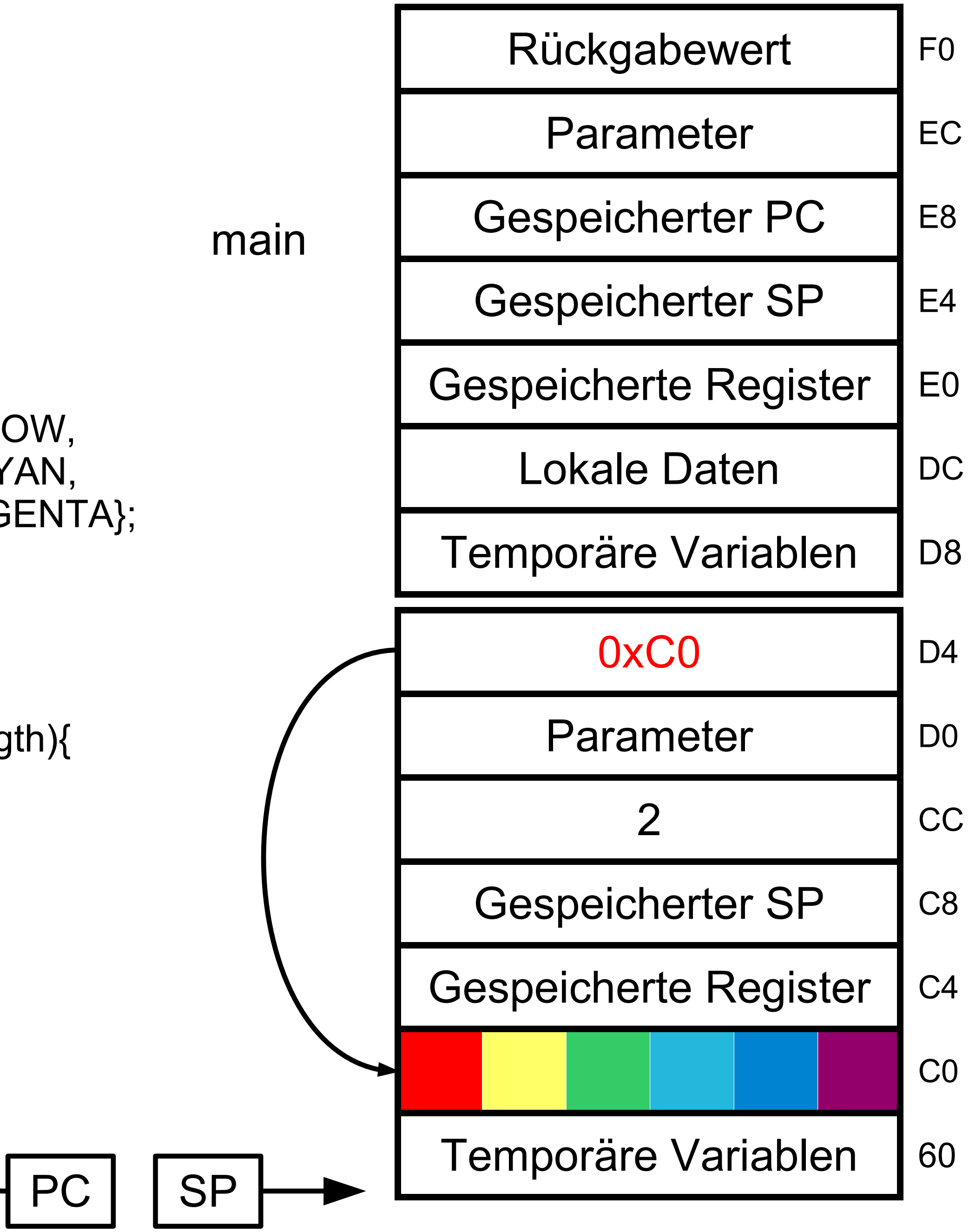
```
int main(){

    int rainbow[]= makeRainbow();
    printRainbow(rainbow, 6);
}

int *makeRainbow(){

     int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                     COLOR_GREEN, COLOR_CYAN,
                     COLOR_BLUE, COLOR_MAGENTA};

   return rainbow;
}

void printRainbow(int rainbow[], int rainbowLength){

     for(int i= 0; i<rainbowLength; i++){
          attron(COLOR_PAIR(rainbow[i]));
          addstr("\u2588");
          attroff(COLOR_PAIR(rainbow[i]));
     }
}
```



main	Rückgabewert	F0
	Parameter	EC
	Gespeicherter PC	E8
	Gespeicherter SP	E4
	Gespeicherte Register	E0
	Lokale Daten	DC
	Temporäre Variablen	D8
makeRainbow	0xC0	D4
	Parameter	D0
	2	CC
	Gespeicherter SP	C8
	Gespeicherte Register	C4
	🟥🟨🟩🟦🟦🟪	C0
	Temporäre Variablen	60

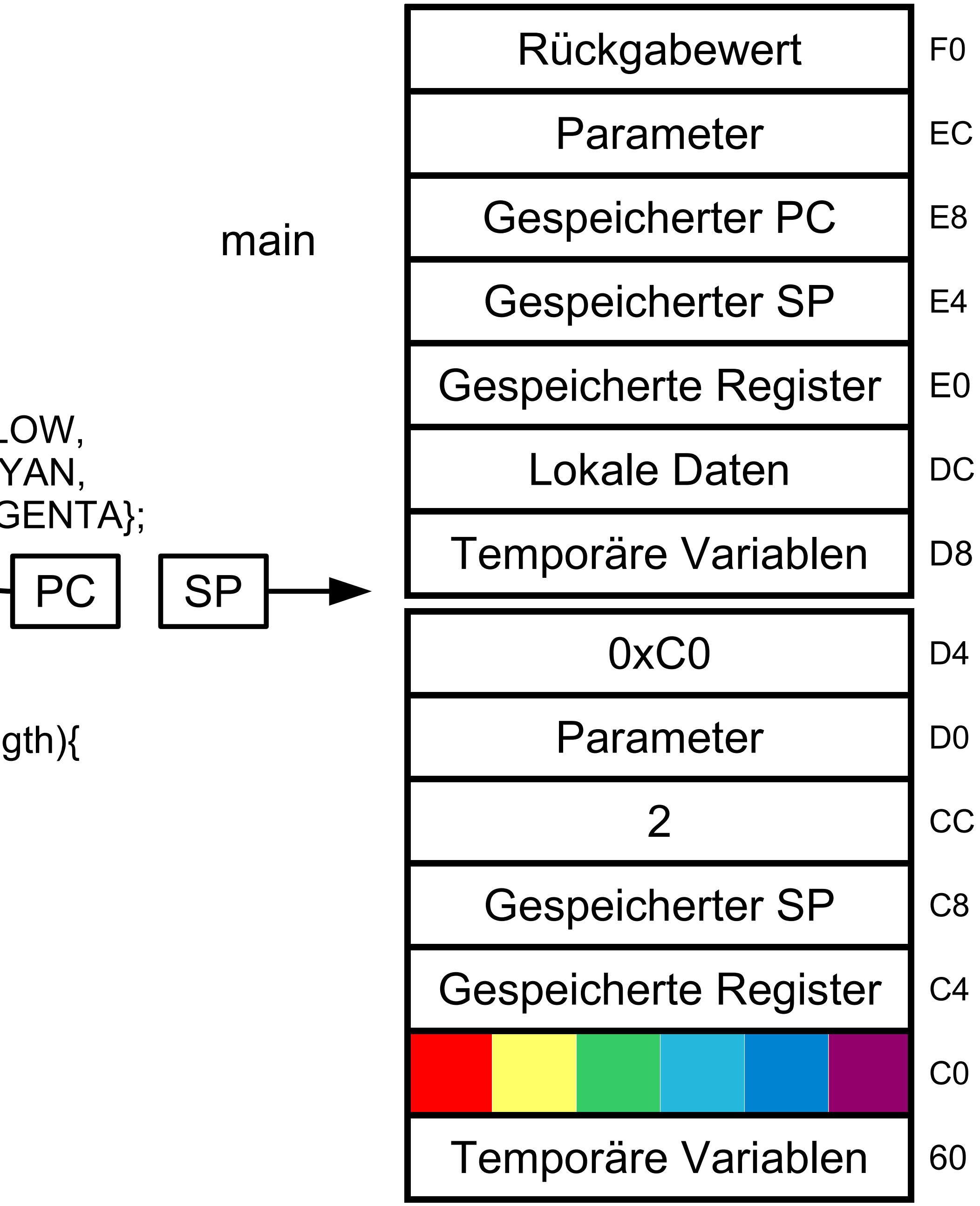
Warum geht das schief?

```
int main(){

    int rainbow[]= makeRainbow();
    printRainbow(rainbow, 6);
}

int *makeRainbow(){

     int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                     COLOR_GREEN, COLOR_CYAN,
                     COLOR_BLUE, COLOR_MAGENTA};

   return rainbow;
}

void printRainbow(int rainbow[], int rainbowLength){

     for(int i= 0; i<rainbowLength; i++){
          attron(COLOR_PAIR(rainbow[i]));
          addstr("\u2588");
          attroff(COLOR_PAIR(rainbow[i]));
    }
}
```



Warum geht das schief?

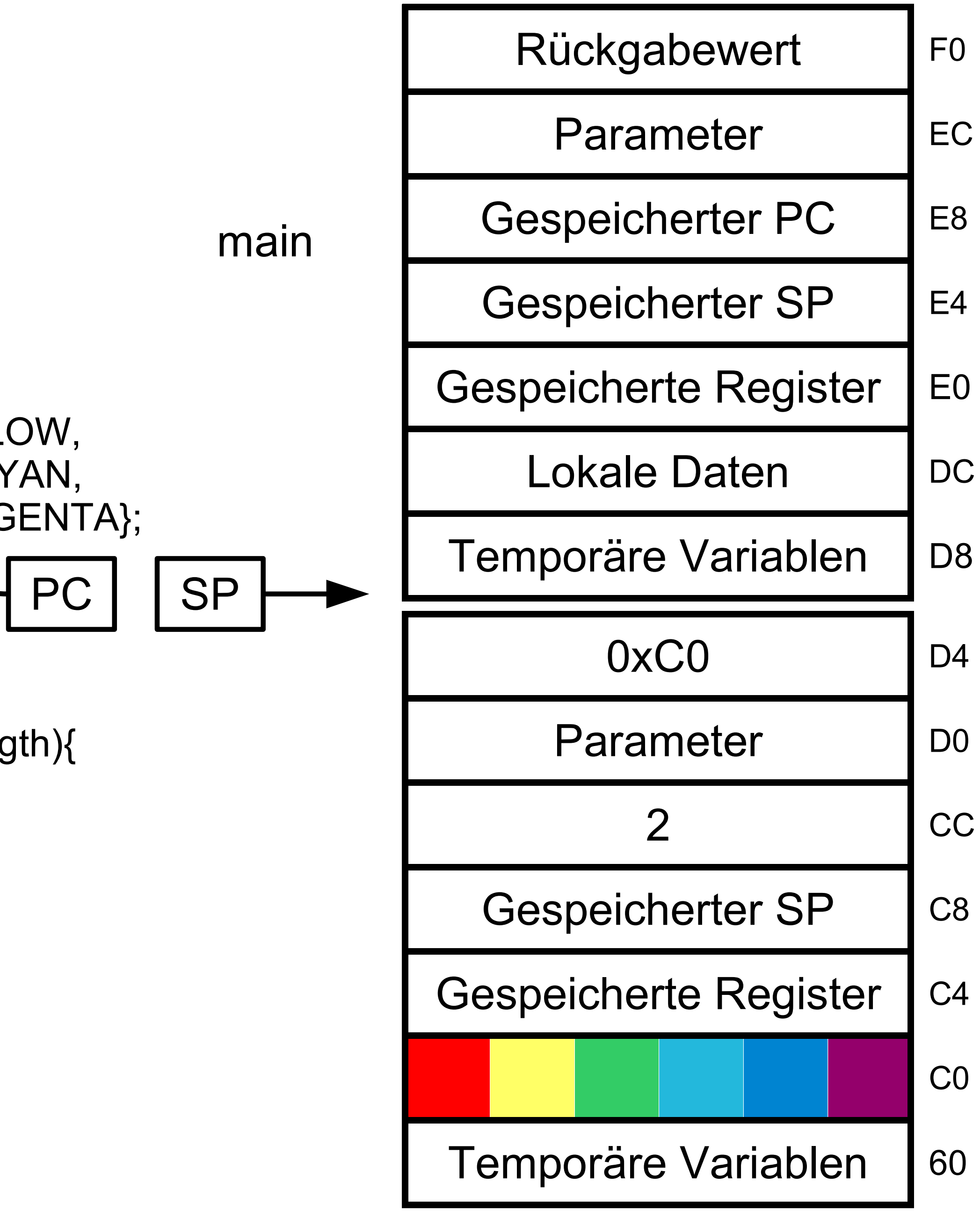
```
int main(){

    int rainbow[]= makeRainbow();
    printRainbow(rainbow, 6);
}

int *makeRainbow(){

     int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                     COLOR_GREEN, COLOR_CYAN,
                     COLOR_BLUE, COLOR_MAGENTA};

    return rainbow;
}

void printRainbow(int rainbow[], int rainbowLength){

     for(int i= 0; i<rainbowLength; i++){
         attron(COLOR_PAIR(rainbow[i]));
         addstr("\u2588");
         attroff(COLOR_PAIR(rainbow[i]));
    }
}
```



Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
Lokale Daten	DC
Temporäre Variablen	D8
0xC0	D4
Parameter	D0
2	CC
Gespeicherter SP	C8
Gespeicherte Register	C4
🟥🟨🟩🟦🟦🟪	C0
Temporäre Variablen	60

Warum geht das schief?

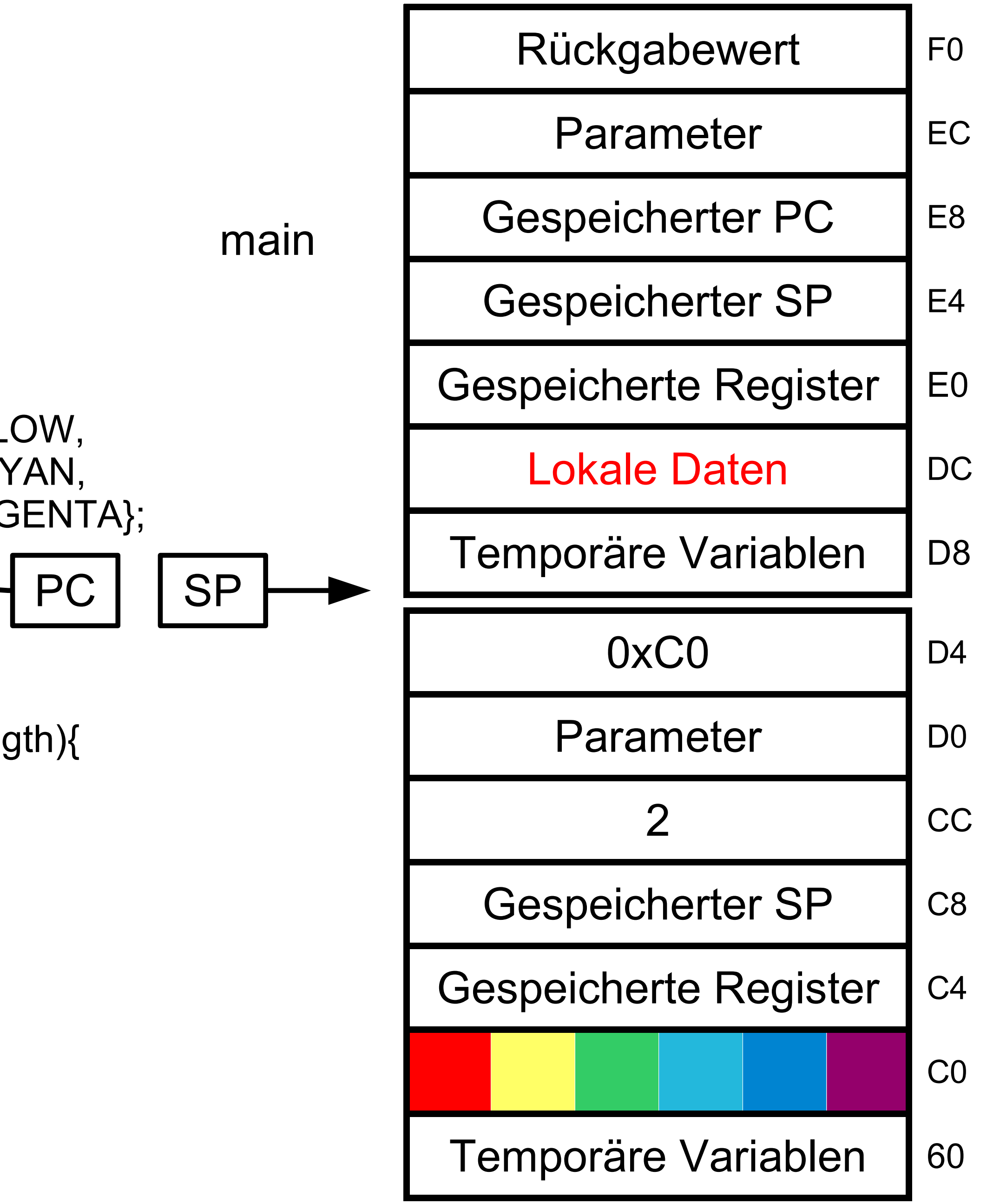
```
int main(){

   int rainbow[]= makeRainbow();
   printRainbow(rainbow, 6);
}

int *makeRainbow(){

    int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                    COLOR_GREEN, COLOR_CYAN,
                    COLOR_BLUE, COLOR_MAGENTA};
  return rainbow;
}

void printRainbow(int rainbow[], int rainbowLength){

    for(int i= 0; i<rainbowLength; i++){
        attron(COLOR_PAIR(rainbow[i]));
        addstr("\u2588");
        attroff(COLOR_PAIR(rainbow[i]));
   }
}
```



Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
Lokale Daten	DC
Temporäre Variablen	D8
0xC0	D4
Parameter	D0
2	CC
Gespeicherter SP	C8
Gespeicherte Register	C4
🟥🟨🟩🟦🟦🟪	C0
Temporäre Variablen	60

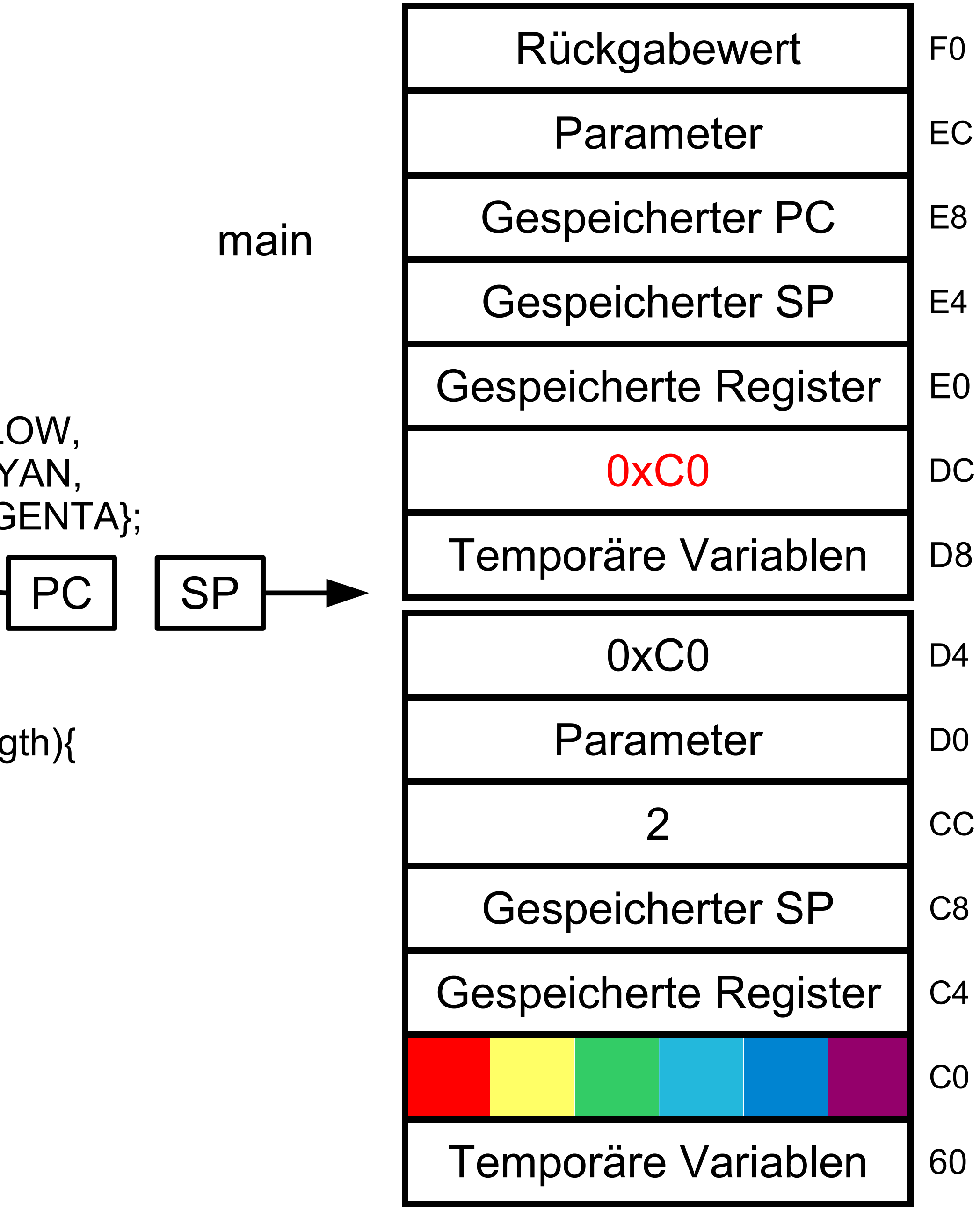
Warum geht das schief?

```
1   int main(){

2       int rainbow[]= makeRainbow();
3       printRainbow(rainbow, 6);
    }

4   int *makeRainbow(){

5        int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                        COLOR_GREEN, COLOR_CYAN,
                        COLOR_BLUE, COLOR_MAGENTA};
6       return rainbow;
    }

7   void printRainbow(int rainbow[], int rainbowLength){

8        for(int i= 0; i<rainbowLength; i++){
9            attron(COLOR_PAIR(rainbow[i]));
10           addstr("\u2588");
11           attroff(COLOR_PAIR(rainbow[i]));
         }
    }
```



Warum geht das schief?

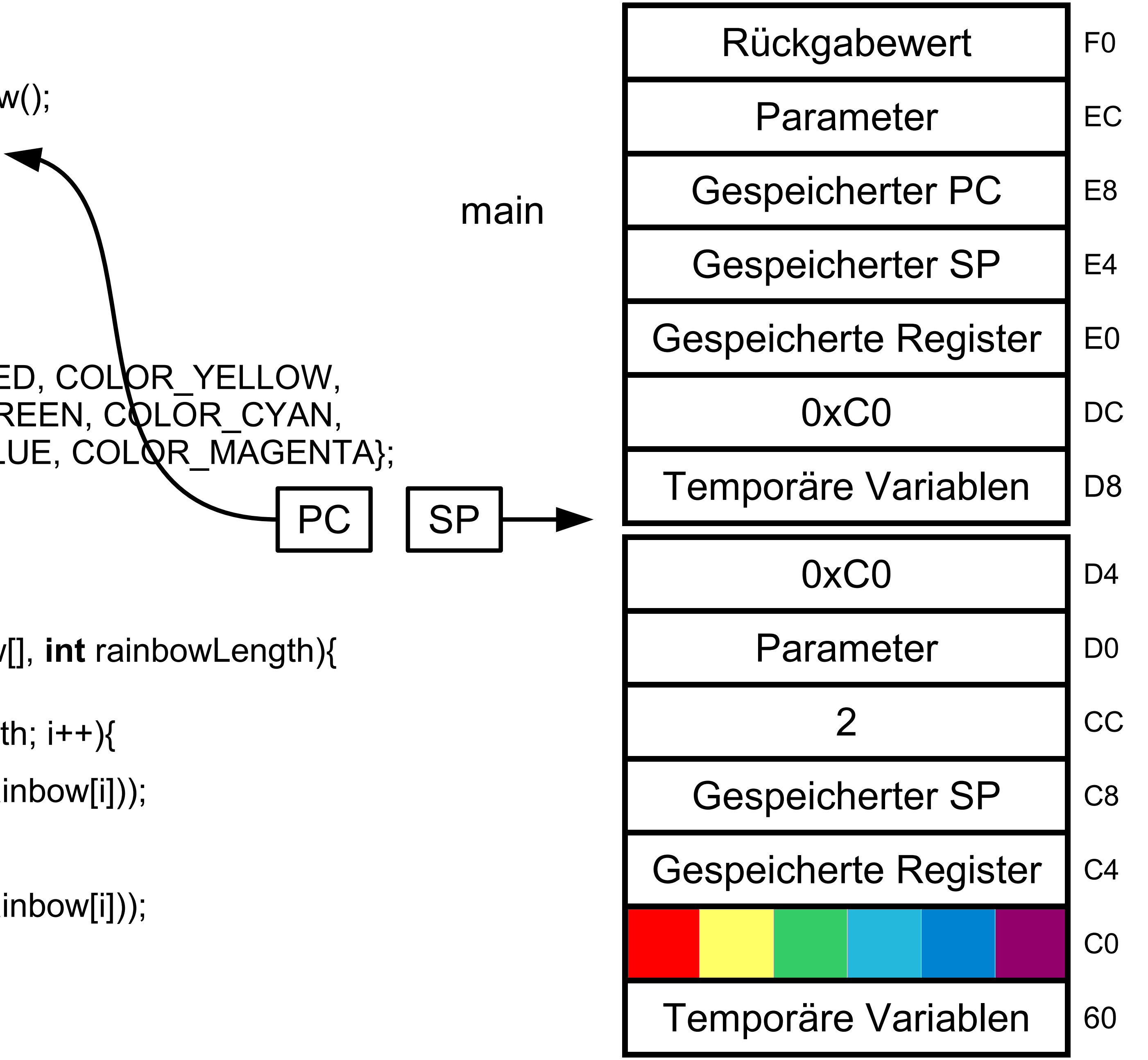
```
int main(){

    int rainbow[]= makeRainbow();
    printRainbow(rainbow, 6);
}

int *makeRainbow(){

     int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                     COLOR_GREEN, COLOR_CYAN,
                     COLOR_BLUE, COLOR_MAGENTA};

   return rainbow;
}

void printRainbow(int rainbow[], int rainbowLength){

     for(int i= 0; i<rainbowLength; i++){
        attron(COLOR_PAIR(rainbow[i]));
        addstr("\u2588");
        attroff(COLOR_PAIR(rainbow[i]));
    }
}
```



Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
0xC0	DC
Temporäre Variablen	D8
0xC0	D4
Parameter	D0
2	CC
Gespeicherter SP	C8
Gespeicherte Register	C4
🟥🟨🟩🟦🟦🟪	C0
Temporäre Variablen	60

Warum geht das schief?

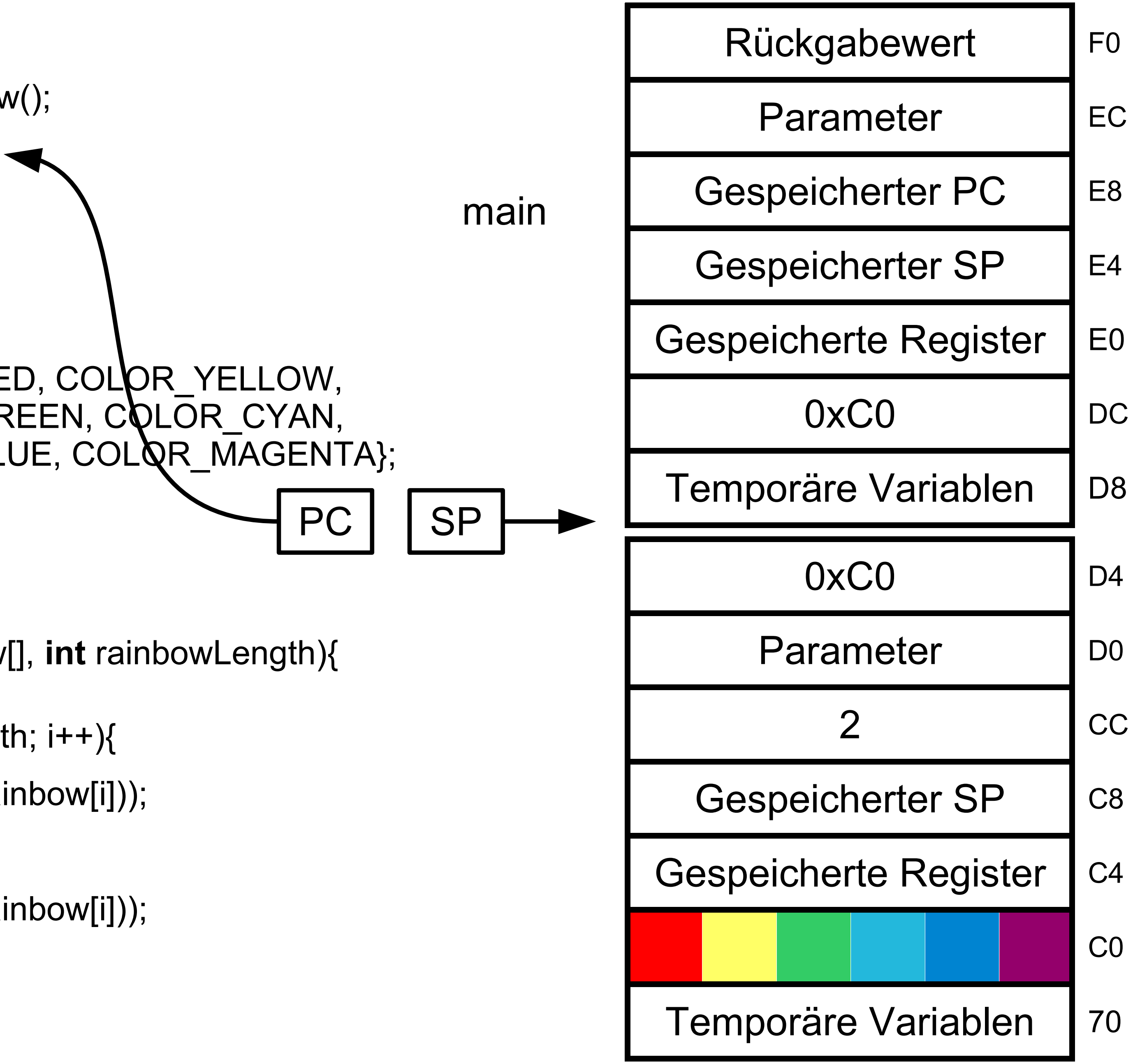
```
int main(){

    int rainbow[]= makeRainbow();
    printRainbow(rainbow, 6);
}

int *makeRainbow(){

     int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                     COLOR_GREEN, COLOR_CYAN,
                     COLOR_BLUE, COLOR_MAGENTA};

   return rainbow;
}

void printRainbow(int rainbow[], int rainbowLength){

     for(int i= 0; i<rainbowLength; i++){
         attron(COLOR_PAIR(rainbow[i]));
         addstr("\u2588");
         attroff(COLOR_PAIR(rainbow[i]));
    }
}
```



Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
0xC0	DC
Temporäre Variablen	D8
0xC0	D4
Parameter	D0
2	CC
Gespeicherter SP	C8
Gespeicherte Register	C4
🟥🟨🟩🟦🟦🟪	C0
Temporäre Variablen	60

Warum geht das schief?

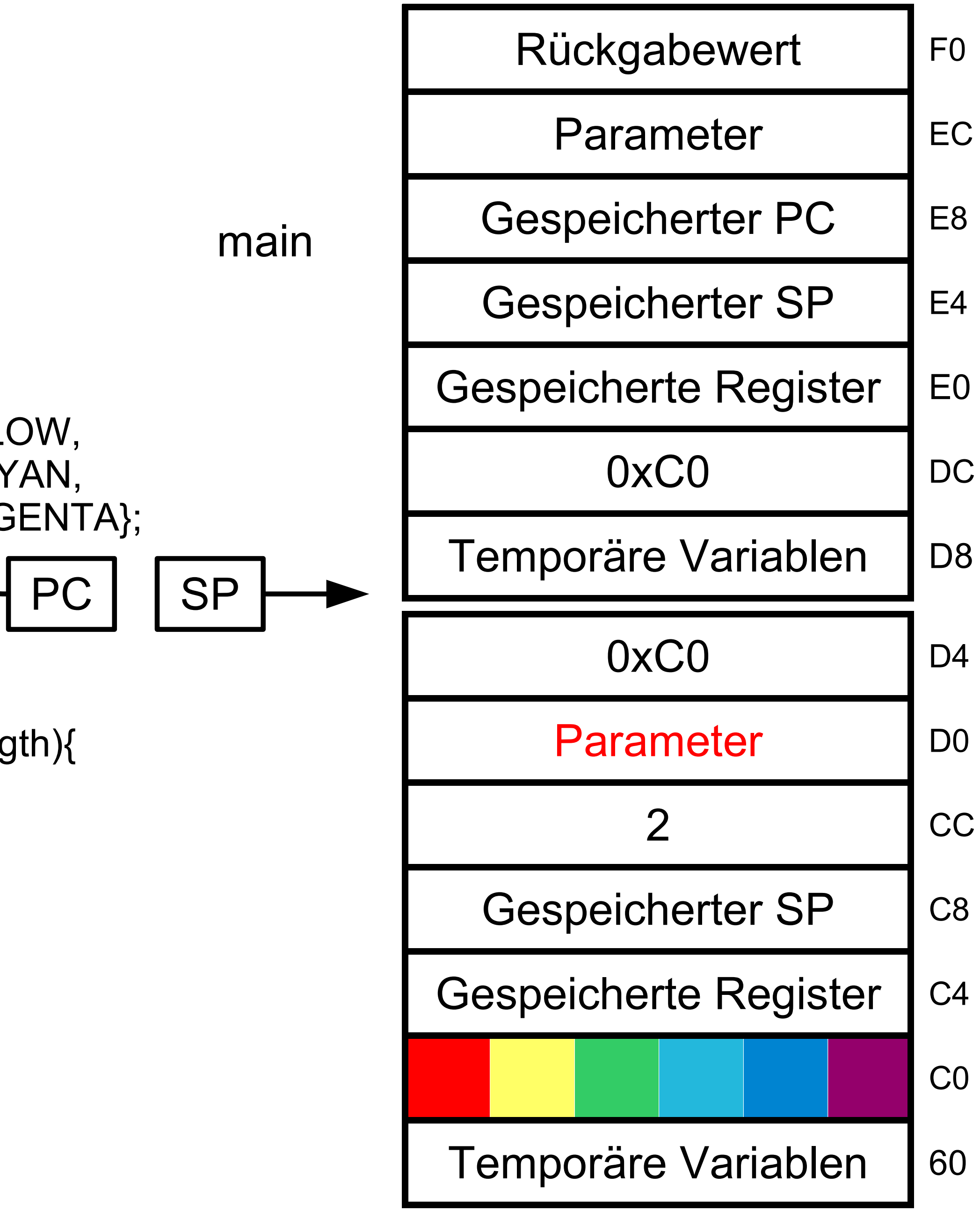
```
int main(){

    int rainbow[]= makeRainbow();
    printRainbow(rainbow, 6);
}

int *makeRainbow(){

     int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                    COLOR_GREEN, COLOR_CYAN,
                    COLOR_BLUE, COLOR_MAGENTA};

   return rainbow;
}

void printRainbow(int rainbow[], int rainbowLength){

     for(int i= 0; i<rainbowLength; i++){
         attron(COLOR_PAIR(rainbow[i]));
         addstr("\u2588");
         attroff(COLOR_PAIR(rainbow[i]));
    }
}
```



Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
0xC0	DC
Temporäre Variablen	D8
0xC0	D4
Parameter	D0
2	CC
Gespeicherter SP	C8
Gespeicherte Register	C4
🟥 🟨 🟩 🟦 🟦 🟪	C0
Temporäre Variablen	70

Warum geht das schief?

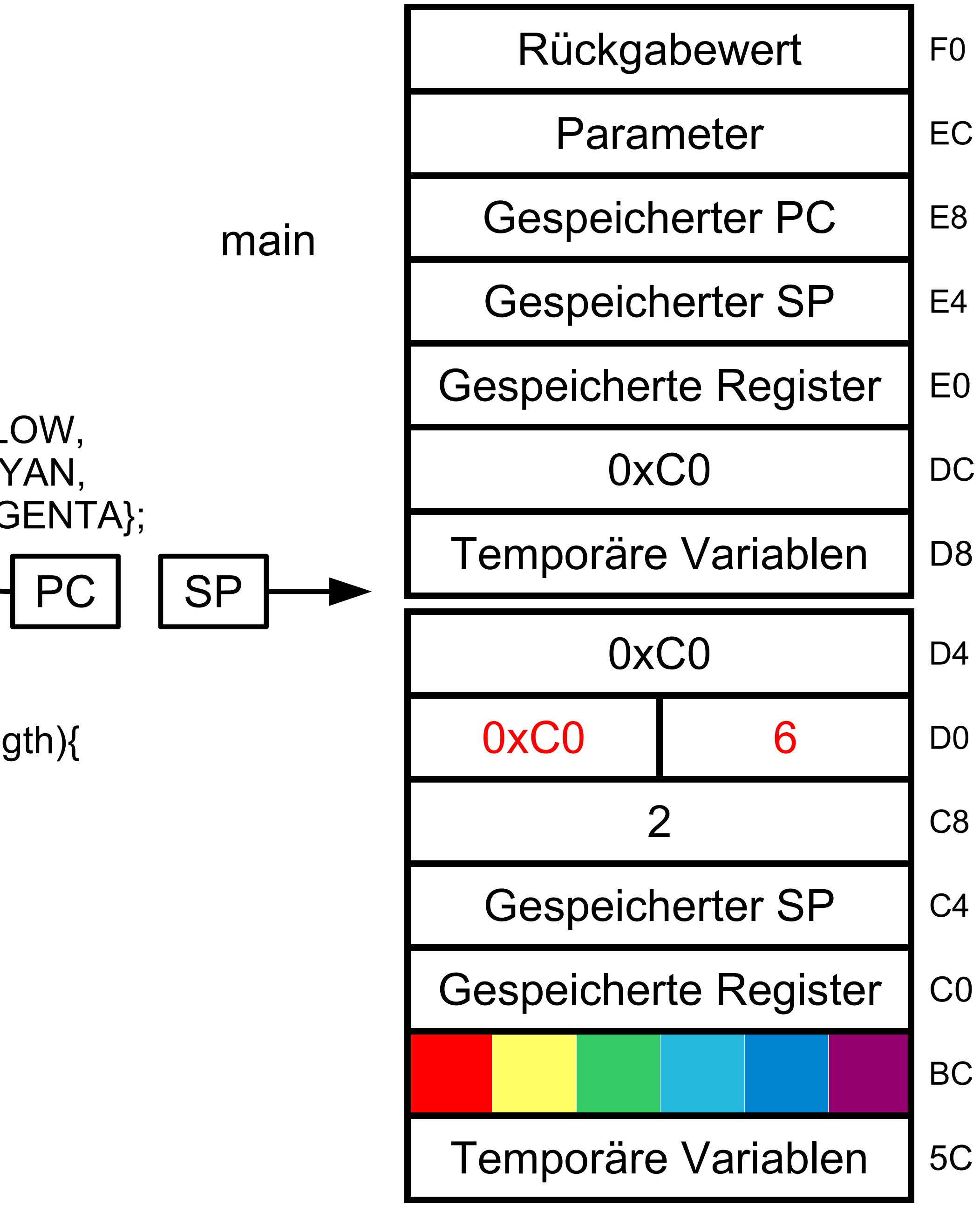
```
int main(){

    int rainbow[]= makeRainbow();
    printRainbow(rainbow, 6);
}

int *makeRainbow(){

     int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                     COLOR_GREEN, COLOR_CYAN,
                     COLOR_BLUE, COLOR_MAGENTA};

   return rainbow;
}

void printRainbow(int rainbow[], int rainbowLength){

     for(int i= 0; i<rainbowLength; i++){
         attron(COLOR_PAIR(rainbow[i]));
         addstr("\u2588");
         attroff(COLOR_PAIR(rainbow[i]));
    }
}
```



Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
0xC0	DC
Temporäre Variablen	D8
0xC0	D4
Parameter	D0
2	CC
Gespeicherter SP	C8
Gespeicherte Register	C4
🟥🟨🟩🟦🟦🟪	C0
Temporäre Variablen	60

Warum geht das schief?

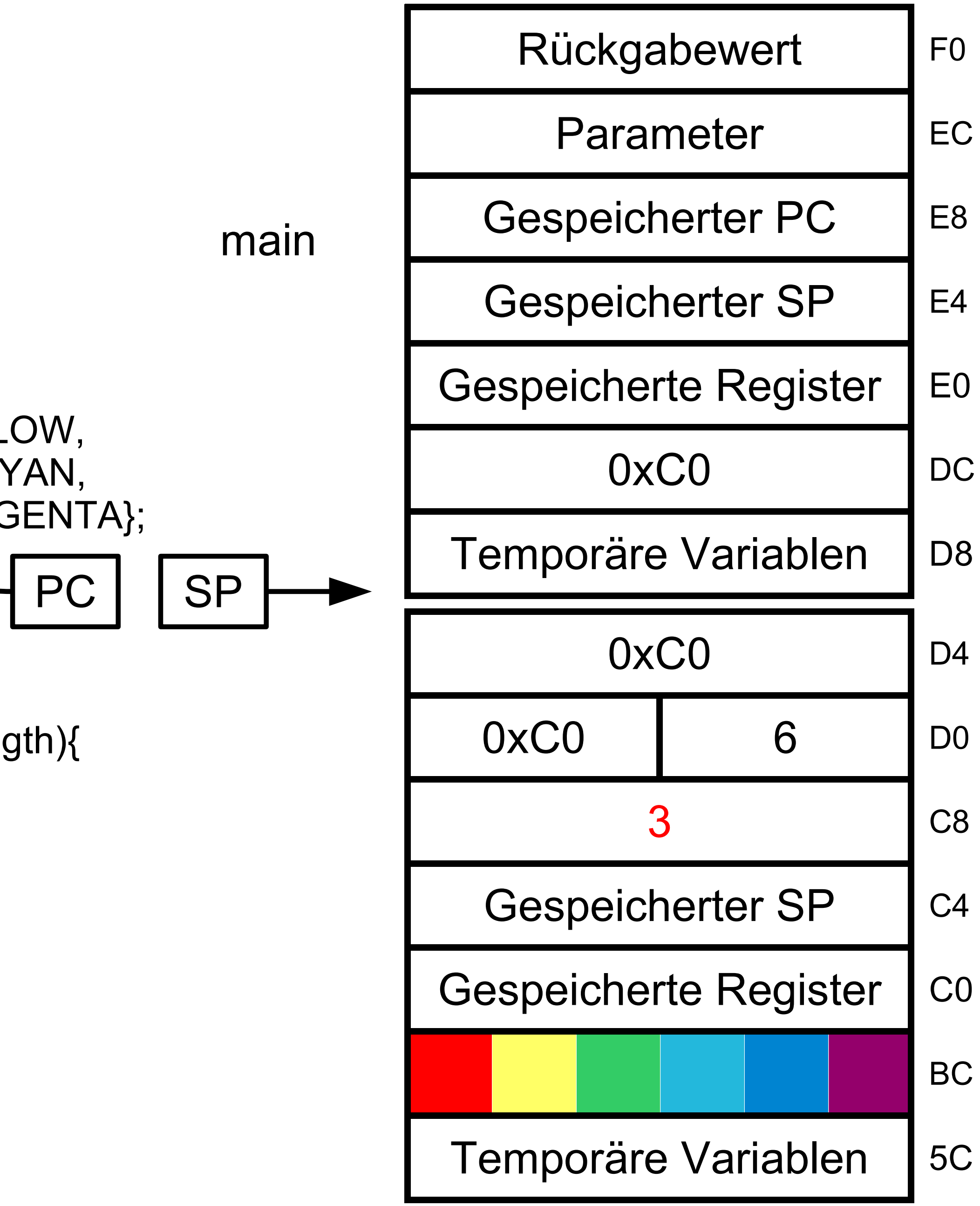
```
int main(){

    int rainbow[]= makeRainbow();
    printRainbow(rainbow, 6);
}

int *makeRainbow(){

     int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                     COLOR_GREEN, COLOR_CYAN,
                     COLOR_BLUE, COLOR_MAGENTA};

   return rainbow;
}

void printRainbow(int rainbow[], int rainbowLength){

     for(int i= 0; i<rainbowLength; i++){
        attron(COLOR_PAIR(rainbow[i]));
        addstr("\u2588");
        attroff(COLOR_PAIR(rainbow[i]));
    }
}
```



Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
0xC0	DC
Temporäre Variablen	D8
0xC0	D4
0xC0 | 6	D0
2	C8
Gespeicherter SP	C4
Gespeicherte Register	C0
🟥 🟨 🟩 🟦 🟦 🟪	BC
Temporäre Variablen	5C

Warum geht das schief?

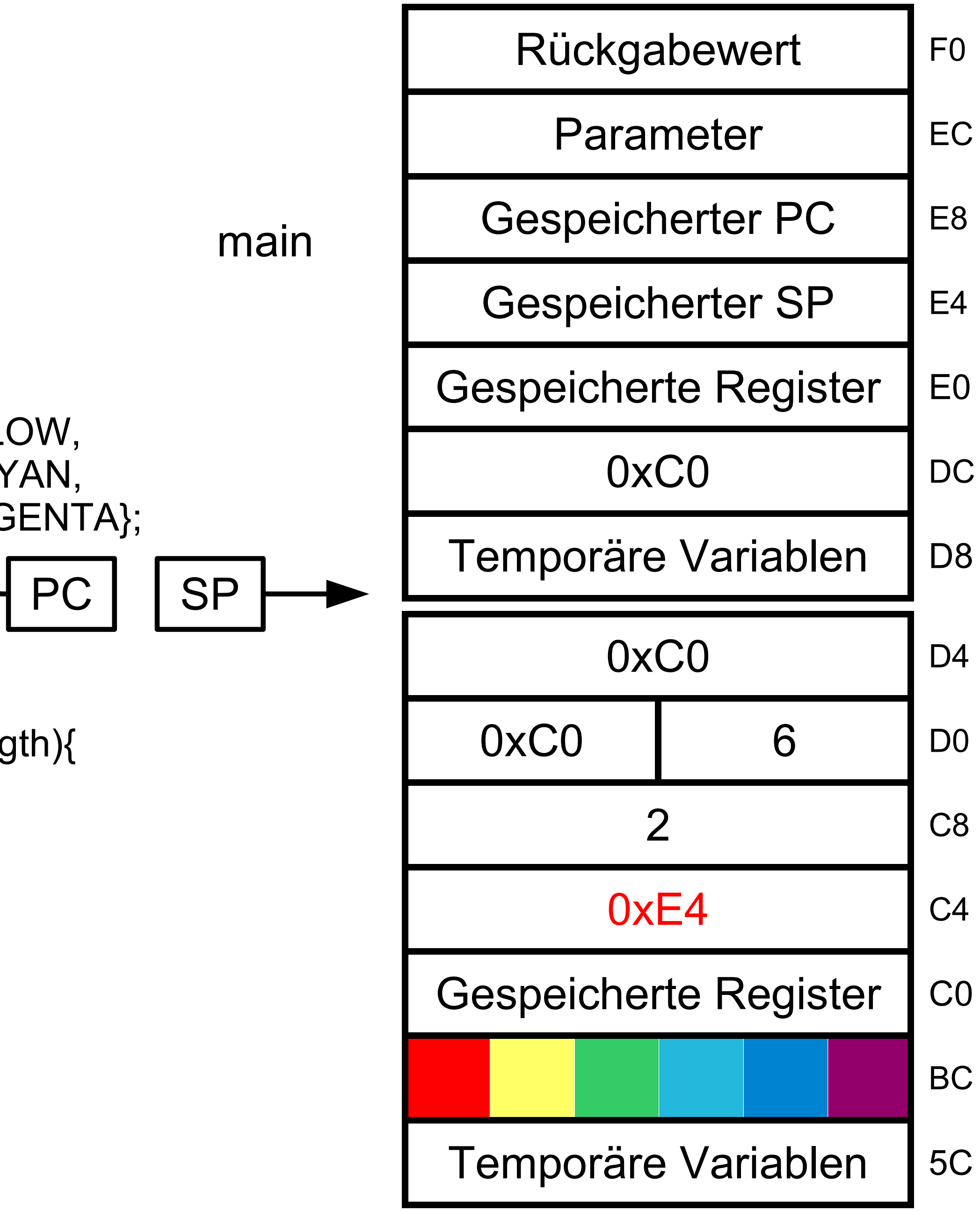
```
1   int main(){

2       int rainbow[]= makeRainbow();
3       printRainbow(rainbow, 6);
    }

4   int *makeRainbow(){

5        int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                         COLOR_GREEN, COLOR_CYAN,
                         COLOR_BLUE, COLOR_MAGENTA};

6      return rainbow;
    }

7   void printRainbow(int rainbow[], int rainbowLength){

8        for(int i= 0; i<rainbowLength; i++){
9            attron(COLOR_PAIR(rainbow[i]));
10           addstr("\u2588");
11           attroff(COLOR_PAIR(rainbow[i]));
         }
    }
```



Rückgabewert		F0
Parameter		EC
Gespeicherter PC		E8
Gespeicherter SP		E4
Gespeicherte Register		E0
0xC0		DC
Temporäre Variablen		D8
0xC0		D4
0xC0	6	D0
3		C8
Gespeicherter SP		C4
Gespeicherte Register		C0
🟥🟨🟩🟦🟦🟪		BC
Temporäre Variablen		5C

Warum geht das schief?

```
int main(){

    int rainbow[]= makeRainbow();
    printRainbow(rainbow, 6);
}

int *makeRainbow(){

     int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                     COLOR_GREEN, COLOR_CYAN,
                     COLOR_BLUE, COLOR_MAGENTA};

   return rainbow;
}

void printRainbow(int rainbow[], int rainbowLength){

     for(int i= 0; i<rainbowLength; i++){
         attron(COLOR_PAIR(rainbow[i]));
         addstr("\u2588");
         attroff(COLOR_PAIR(rainbow[i]));
    }
}
```

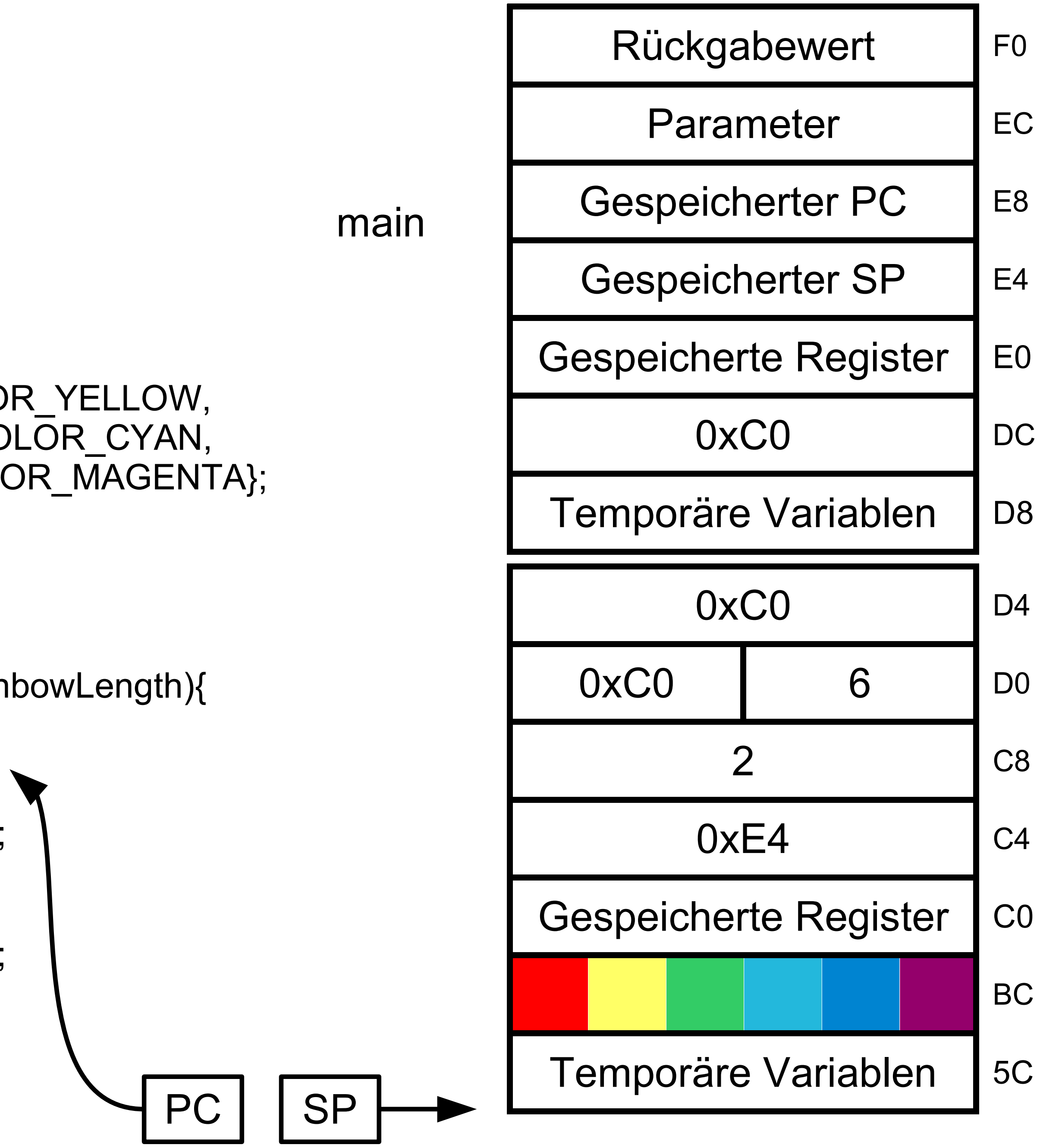


main

Rückgabewert		F0
Parameter		EC
Gespeicherter PC		E8
Gespeicherter SP		E4
Gespeicherte Register		E0
0xC0		DC
Temporäre Variablen		D8
0xC0		D4
0xC0	6	D0
2		C8
0xE4		C4
Gespeicherte Register		C0
🟥🟨🟩🟦🟦🟪		BC
Temporäre Variablen		5C

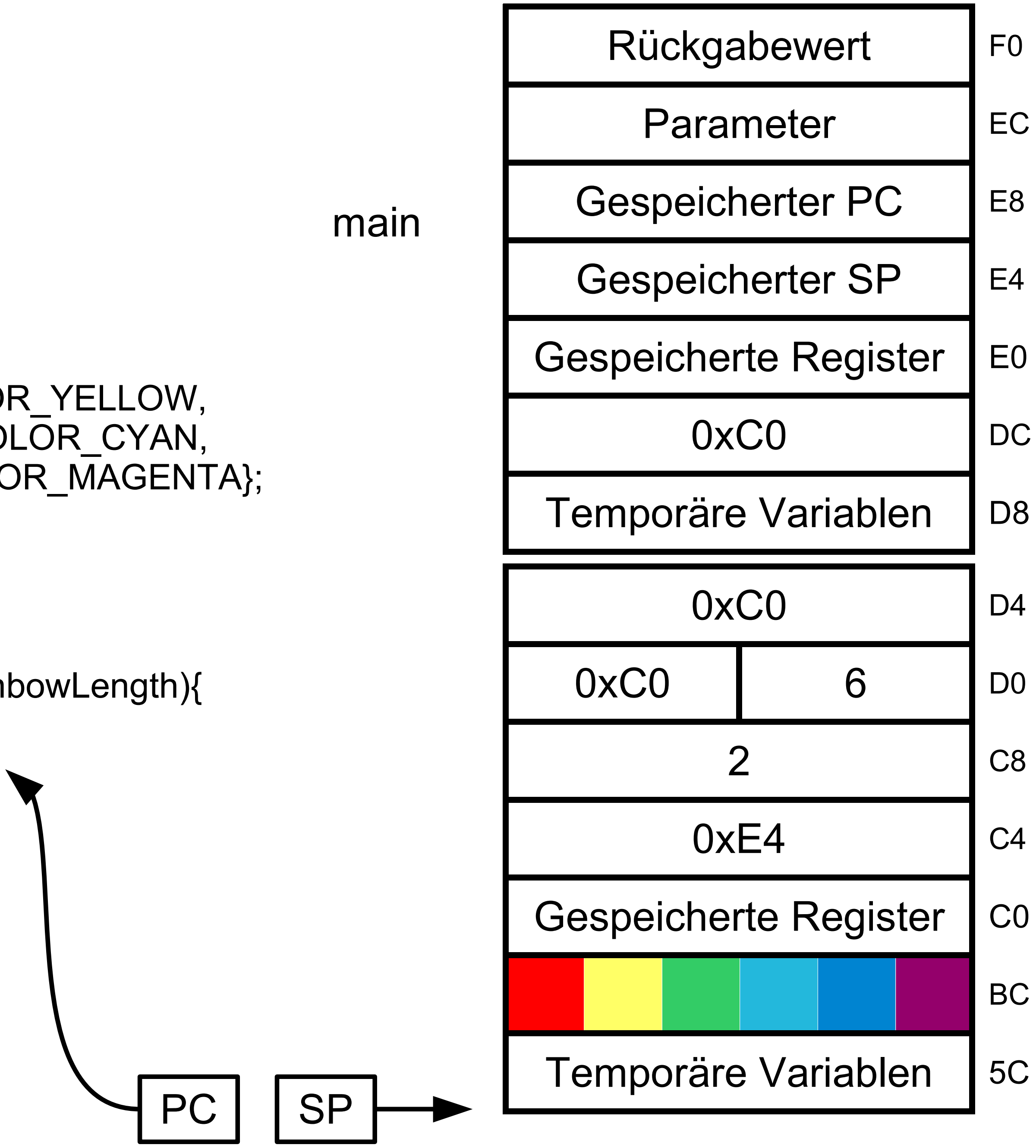
Warum geht das schief?

```
int main(){

    int rainbow[]= makeRainbow();
    printRainbow(rainbow, 6);
}

int *makeRainbow(){

     int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                     COLOR_GREEN, COLOR_CYAN,
                     COLOR_BLUE, COLOR_MAGENTA};
   return rainbow;
}

void printRainbow(int rainbow[], int rainbowLength){

     for(int i= 0; i<rainbowLength; i++){
         attron(COLOR_PAIR(rainbow[i]));
         addstr("\u2588");
         attroff(COLOR_PAIR(rainbow[i]));
    }
}
```



Warum geht das schief?

```
1   int main(){

2       int rainbow[]= makeRainbow();
3       printRainbow(rainbow, 6);
    }

4   int *makeRainbow(){

5        int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                         COLOR_GREEN, COLOR_CYAN,
                         COLOR_BLUE, COLOR_MAGENTA};
6      return rainbow;
    }

7   void printRainbow(int rainbow[], int rainbowLength){

8        for(int i= 0; i<rainbowLength; i++){
9            attron(COLOR_PAIR(rainbow[i]));
10           addstr("\u2588");
11           attroff(COLOR_PAIR(rainbow[i]));
         }
    }
```

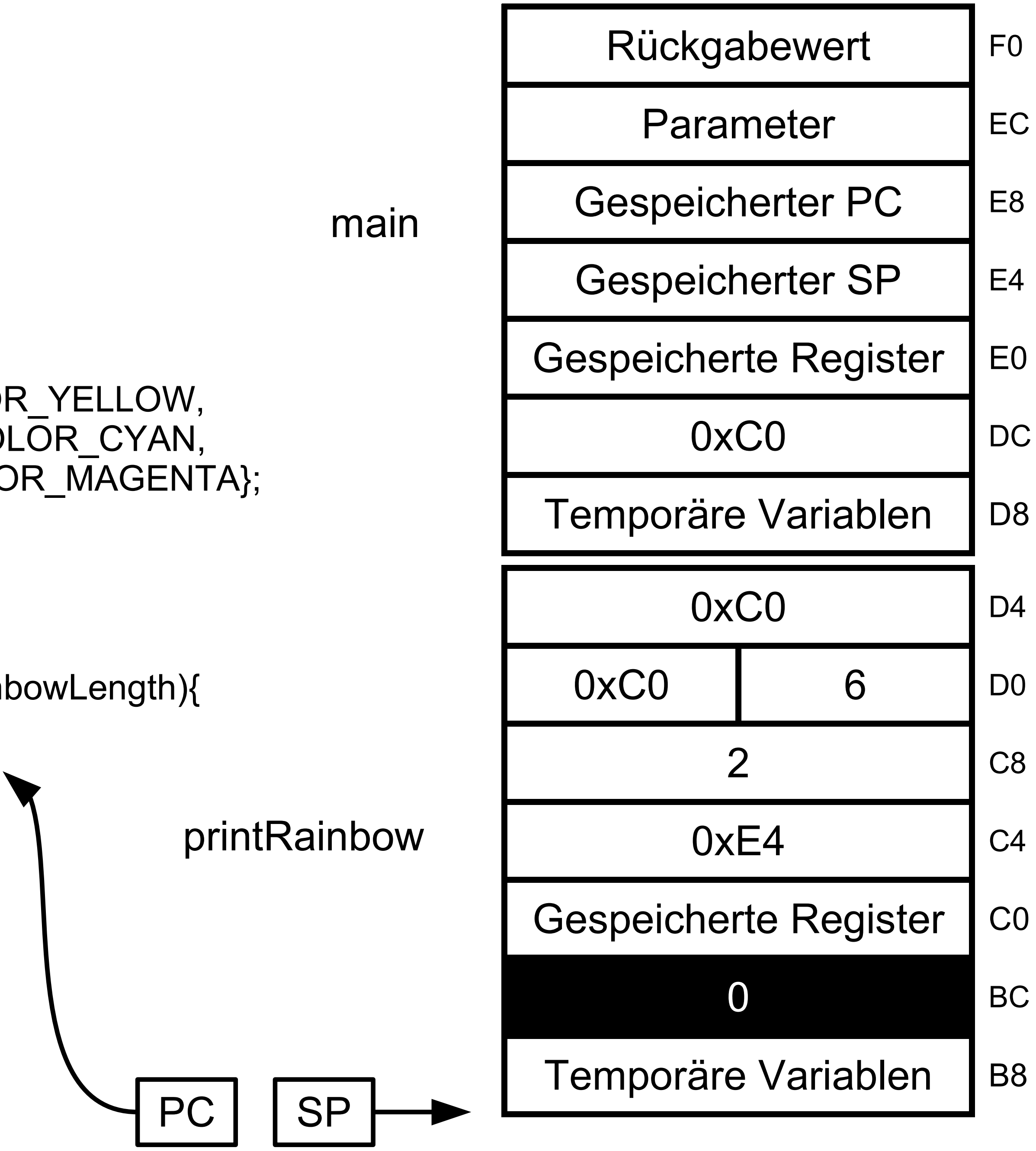


main

Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
0xC0	DC
Temporäre Variablen	D8
0xC0	D4
0xC0 | 6	D0
2	C8
0xE4	C4
Gespeicherte Register	C0
🟥 🟨 🟩 🟦 🟦 🟪	BC
Temporäre Variablen	5C

Warum geht das schief?

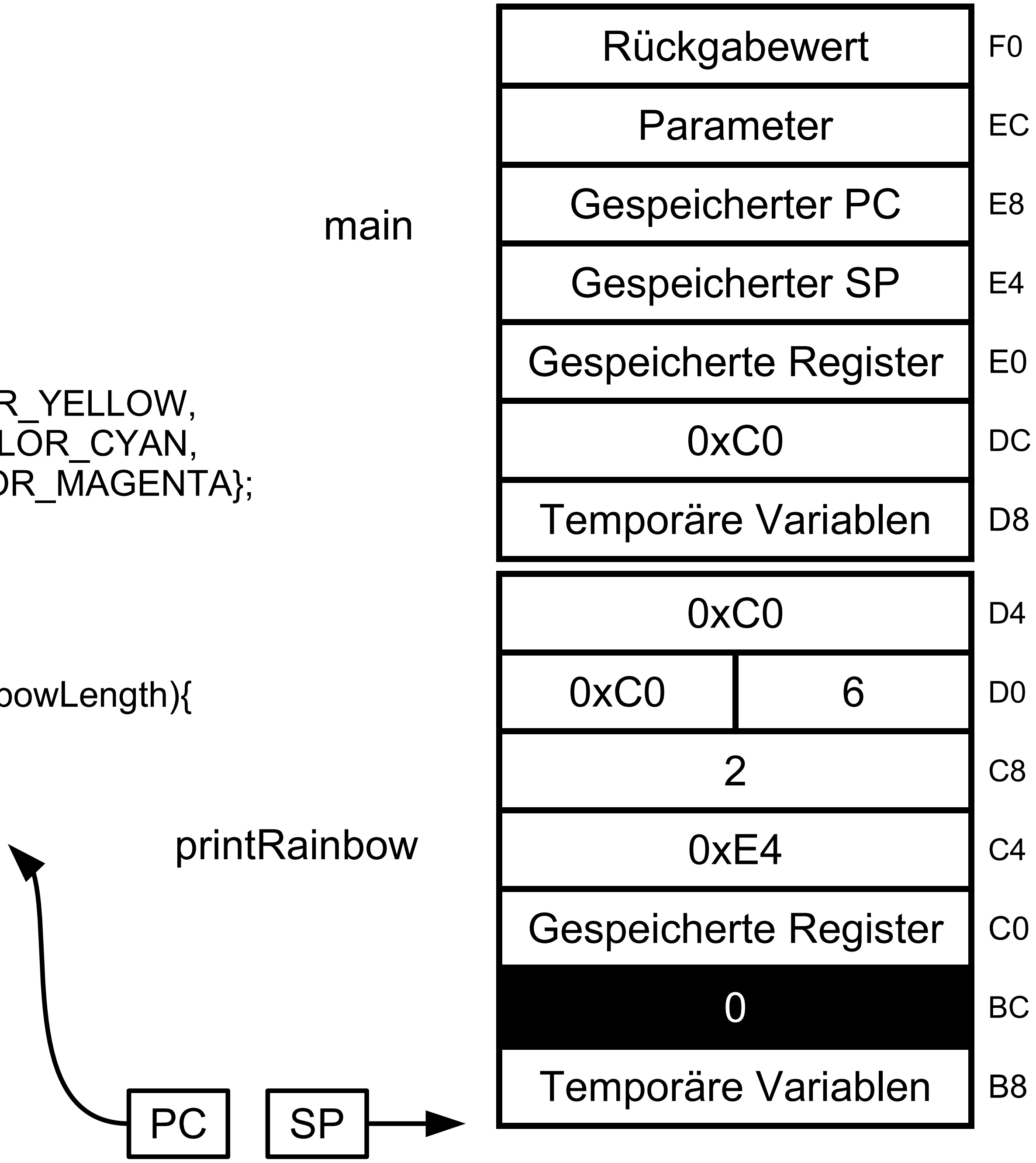
```
1  int main(){

2      int rainbow[]= makeRainbow();
3      printRainbow(rainbow, 6);
   }

4  int *makeRainbow(){

5       int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                        COLOR_GREEN, COLOR_CYAN,
                        COLOR_BLUE, COLOR_MAGENTA};
6     return rainbow;
   }

7  void printRainbow(int rainbow[], int rainbowLength){

8       for(int i= 0; i<rainbowLength; i++){
9           attron(COLOR_PAIR(rainbow[i]));
10          addstr("\u2588");
11          attroff(COLOR_PAIR(rainbow[i]));
        }
   }
```



Rückgabewert		F0
Parameter		EC
Gespeicherter PC		E8
Gespeicherter SP		E4
Gespeicherte Register		E0
0xC0		DC
Temporäre Variablen		D8
0xC0		D4
0xC0	6	D0
2		C8
0xE4		C4
Gespeicherte Register		C0
0		BC
Temporäre Variablen		B8

Warum geht das schief?

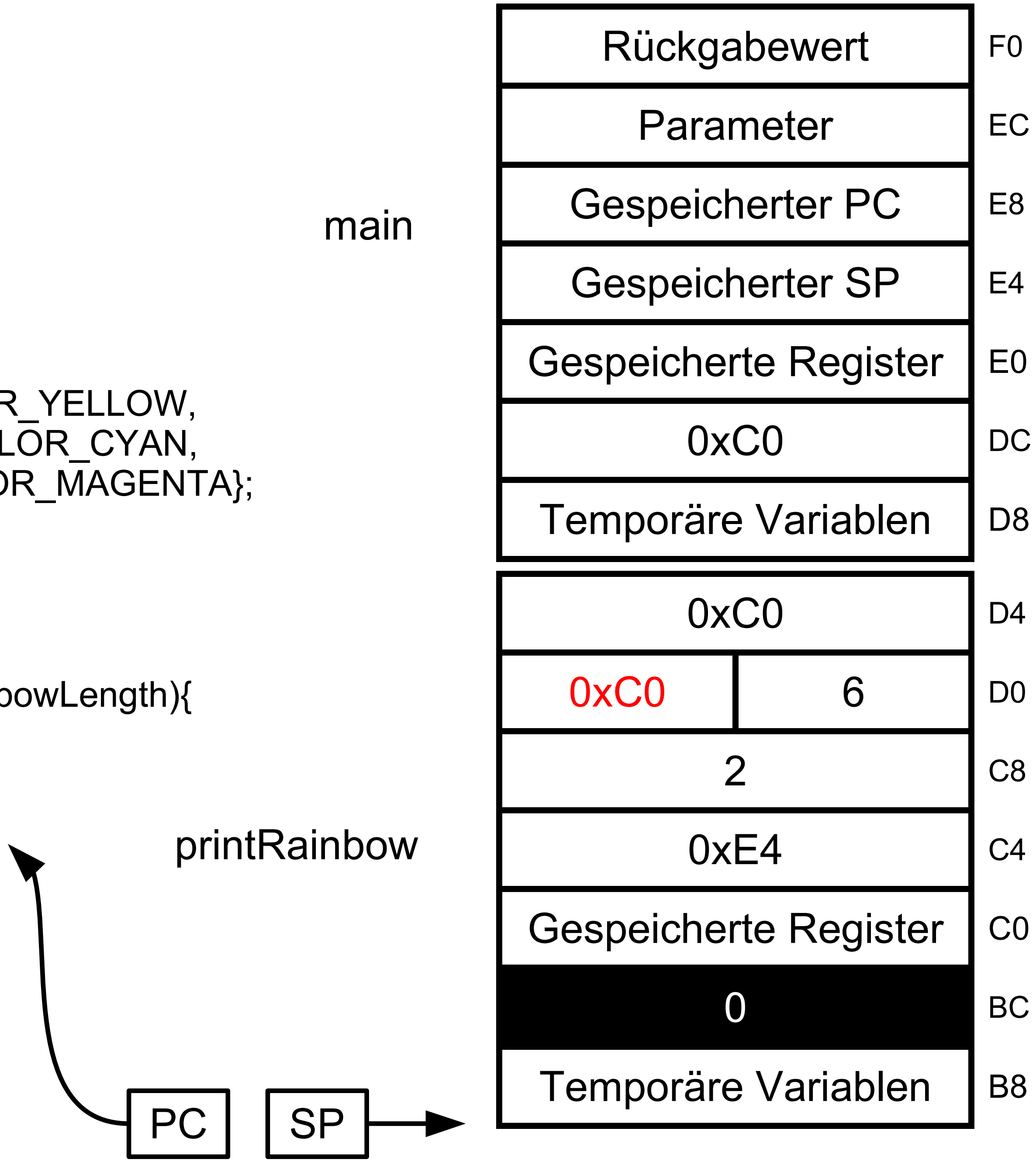
```
int main(){

    int rainbow[]= makeRainbow();
    printRainbow(rainbow, 6);
}

int *makeRainbow(){

     int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                     COLOR_GREEN, COLOR_CYAN,
                     COLOR_BLUE, COLOR_MAGENTA};
   return rainbow;
}

void printRainbow(int rainbow[], int rainbowLength){

     for(int i= 0; i<rainbowLength; i++){
         attron(COLOR_PAIR(rainbow[i]));
         addstr("\u2588");
         attroff(COLOR_PAIR(rainbow[i]));
    }
}
```



main	Rückgabewert		F0
	Parameter		EC
	Gespeicherter PC		E8
	Gespeicherter SP		E4
	Gespeicherte Register		E0
	0xC0		DC
	Temporäre Variablen		D8
printRainbow	0xC0		D4
	0xC0	6	D0
	2		C8
	0xE4		C4
	Gespeicherte Register		C0
	0		BC
	Temporäre Variablen		B8

Warum geht das schief?

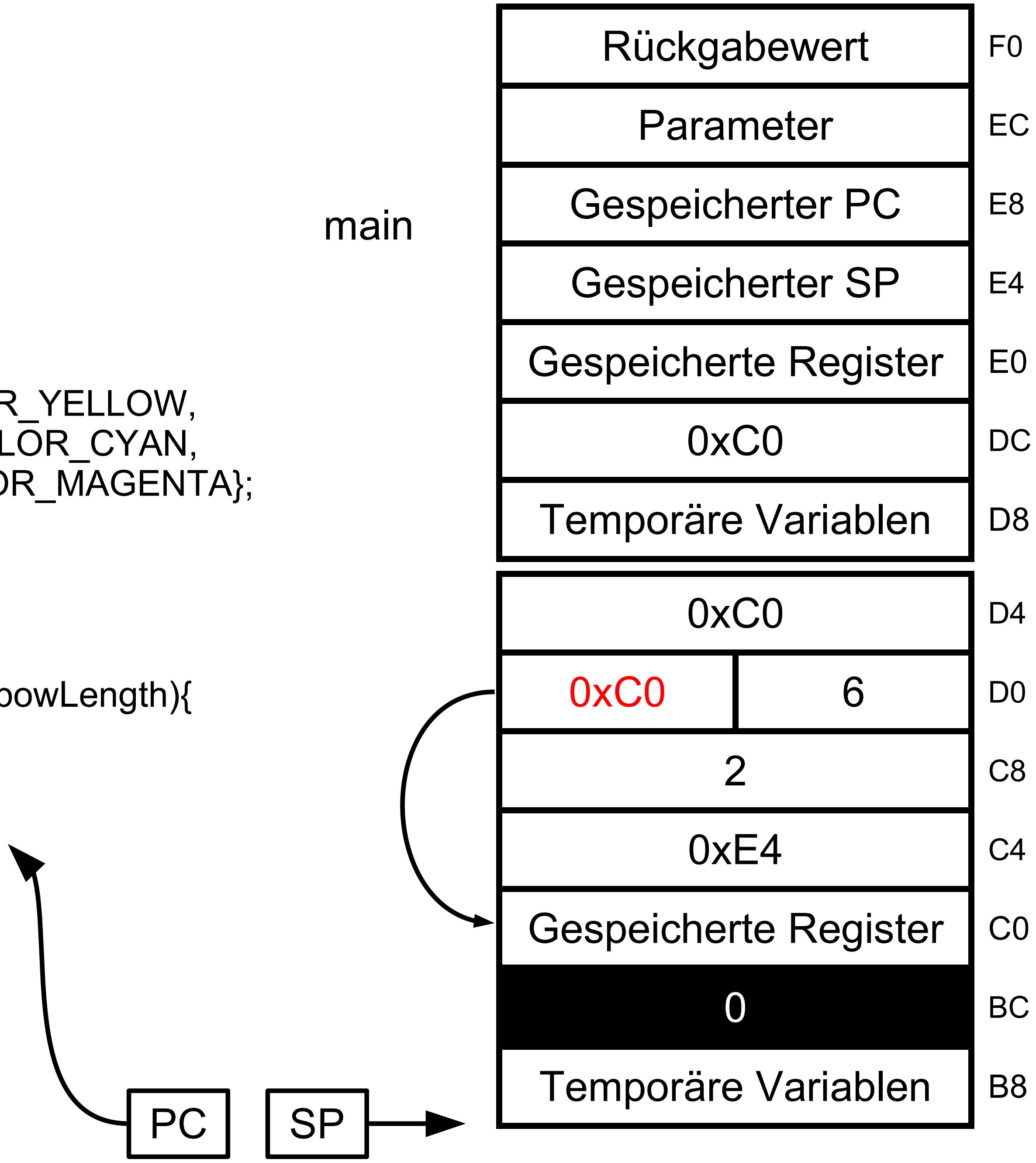
```
int main(){

    int rainbow[]= makeRainbow();
    printRainbow(rainbow, 6);
}

int *makeRainbow(){

     int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                     COLOR_GREEN, COLOR_CYAN,
                     COLOR_BLUE, COLOR_MAGENTA};
   return rainbow;
}

void printRainbow(int rainbow[], int rainbowLength){

     for(int i= 0; i<rainbowLength; i++){
         attron(COLOR_PAIR(rainbow[i]));
         addstr("\u2588");
         attroff(COLOR_PAIR(rainbow[i]));
    }
}
```



main	Rückgabewert	F0
	Parameter	EC
	Gespeicherter PC	E8
	Gespeicherter SP	E4
	Gespeicherte Register	E0
	0xC0	DC
	Temporäre Variablen	D8
printRainbow	0xC0	D4
	0xC0 | 6	D0
	2	C8
	0xE4	C4
	Gespeicherte Register	C0
	0	BC
	Temporäre Variablen	B8

Warum geht das schief?

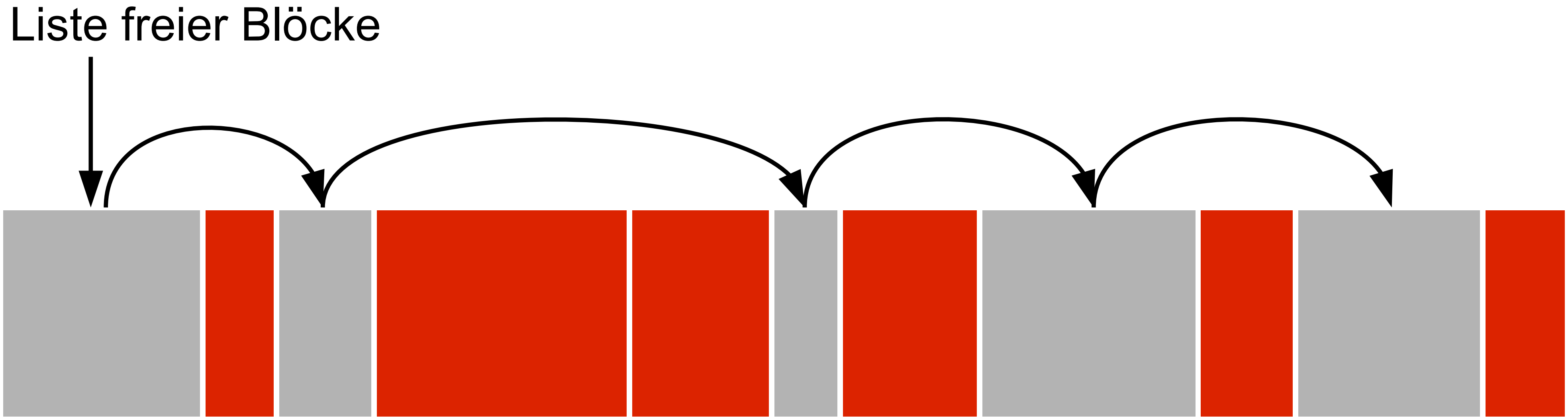
```
int main(){

    int rainbow[]= makeRainbow();
    printRainbow(rainbow, 6);
}

int *makeRainbow(){

     int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                     COLOR_GREEN, COLOR_CYAN,
                     COLOR_BLUE, COLOR_MAGENTA};
   return rainbow;
}

void printRainbow(int rainbow[], int rainbowLength){

     for(int i= 0; i<rainbowLength; i++){
          attron(COLOR_PAIR(rainbow[i]));
          addstr("\u2588");
          attroff(COLOR_PAIR(rainbow[i]));
    }
}
```



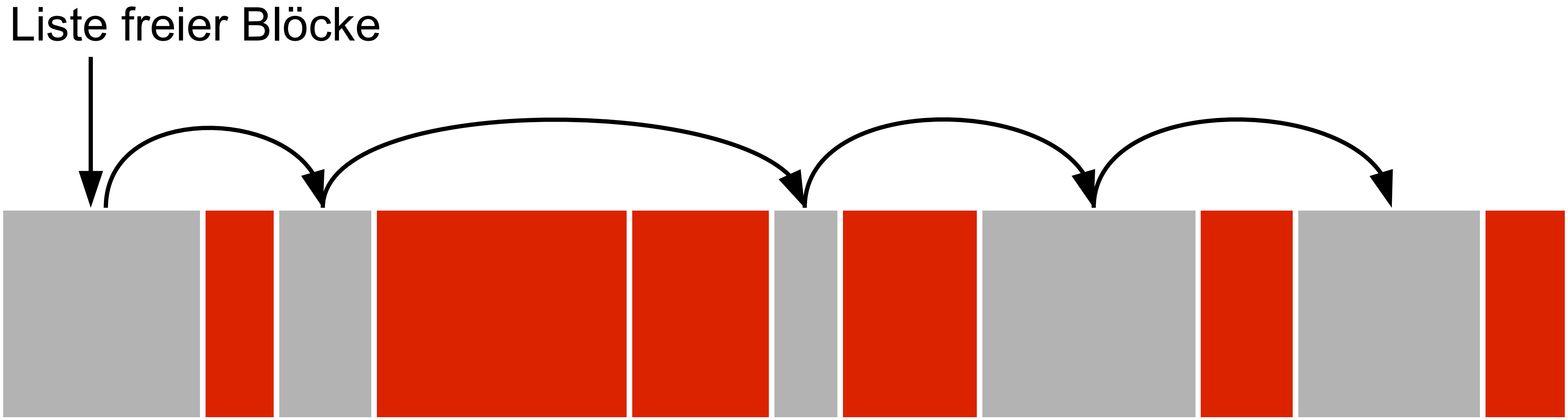
main

Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
0xC0	DC
Temporäre Variablen	D8
0xC0	D4
0xC0 | 6	D0
2	C8
0xE4	C4
Gespeicherte Register	C0
0	BC
Temporäre Variablen	B8

Warum geht das schief?

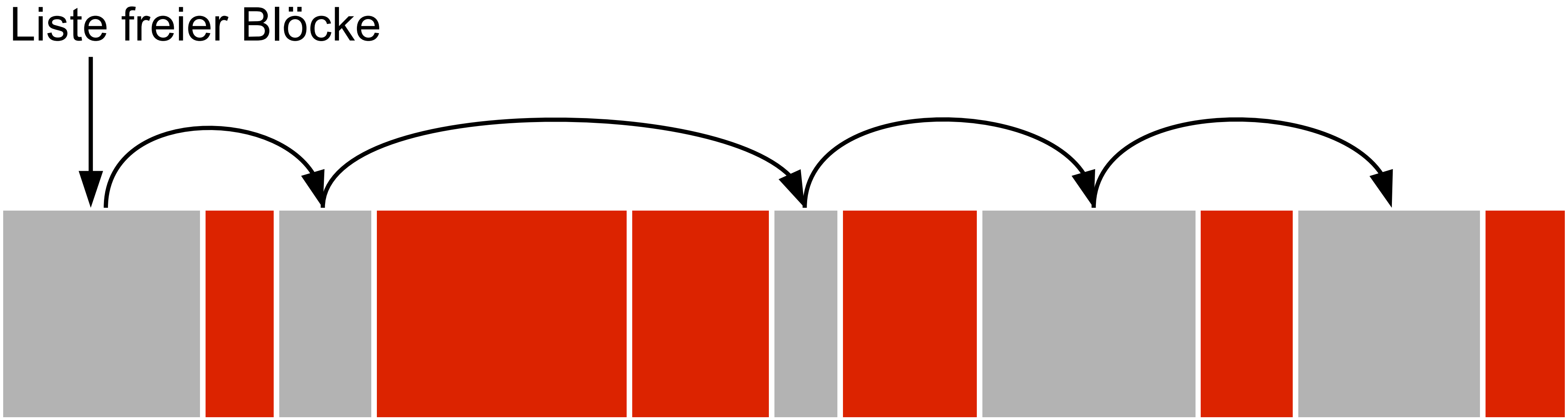


Liste freier Blöcke



Anfordern von Speicher:

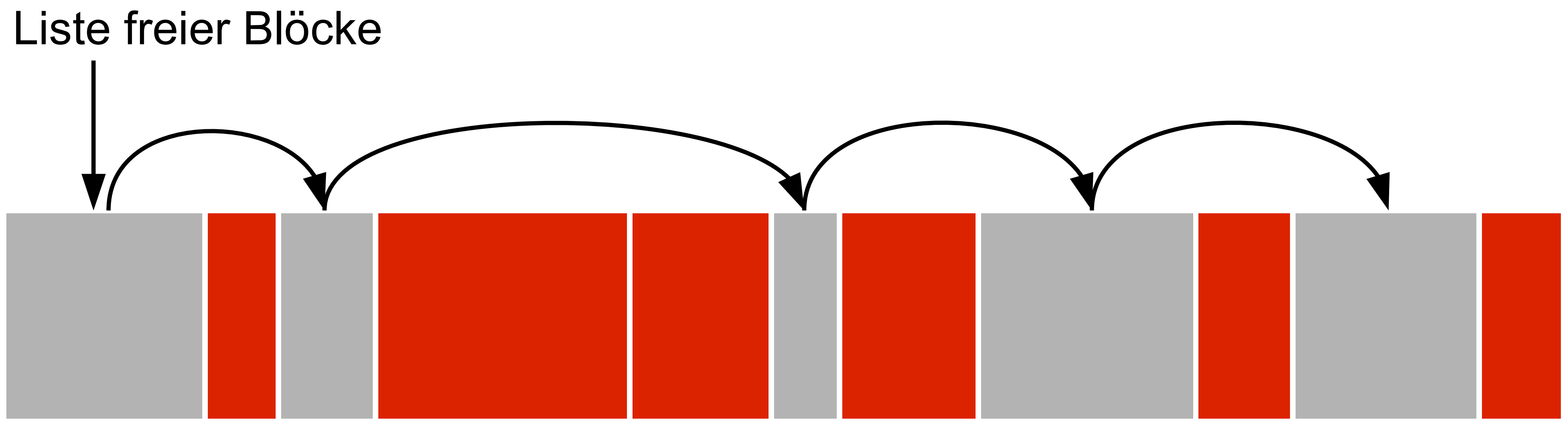
void *malloc(size_t size);
void *calloc(size_t nmemb, size_t size);



Anfordern von Speicher:

void *malloc(size_t size);
void *calloc(size_t nmemb, size_t size);

Entspricht dem **new** in Java



Anfordern von Speicher:

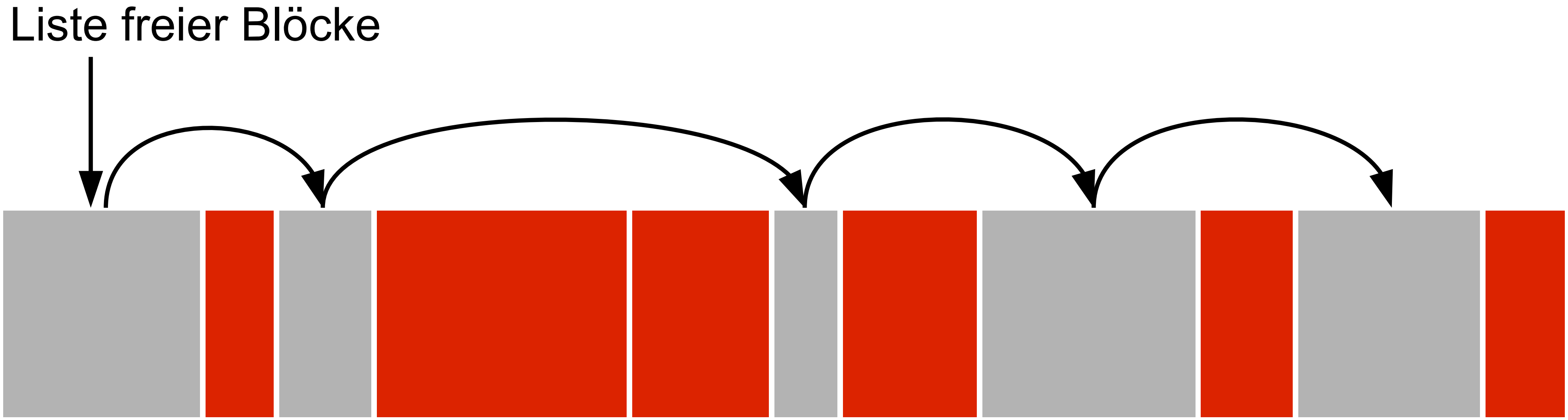
void *malloc(size_t size);
void *calloc(size_t nmemb, size_t size);

Entspricht dem **new** in Java

Freigeben von Speicher:

void free(**void** *ptr);

Der Heap



Anfordern von Speicher:

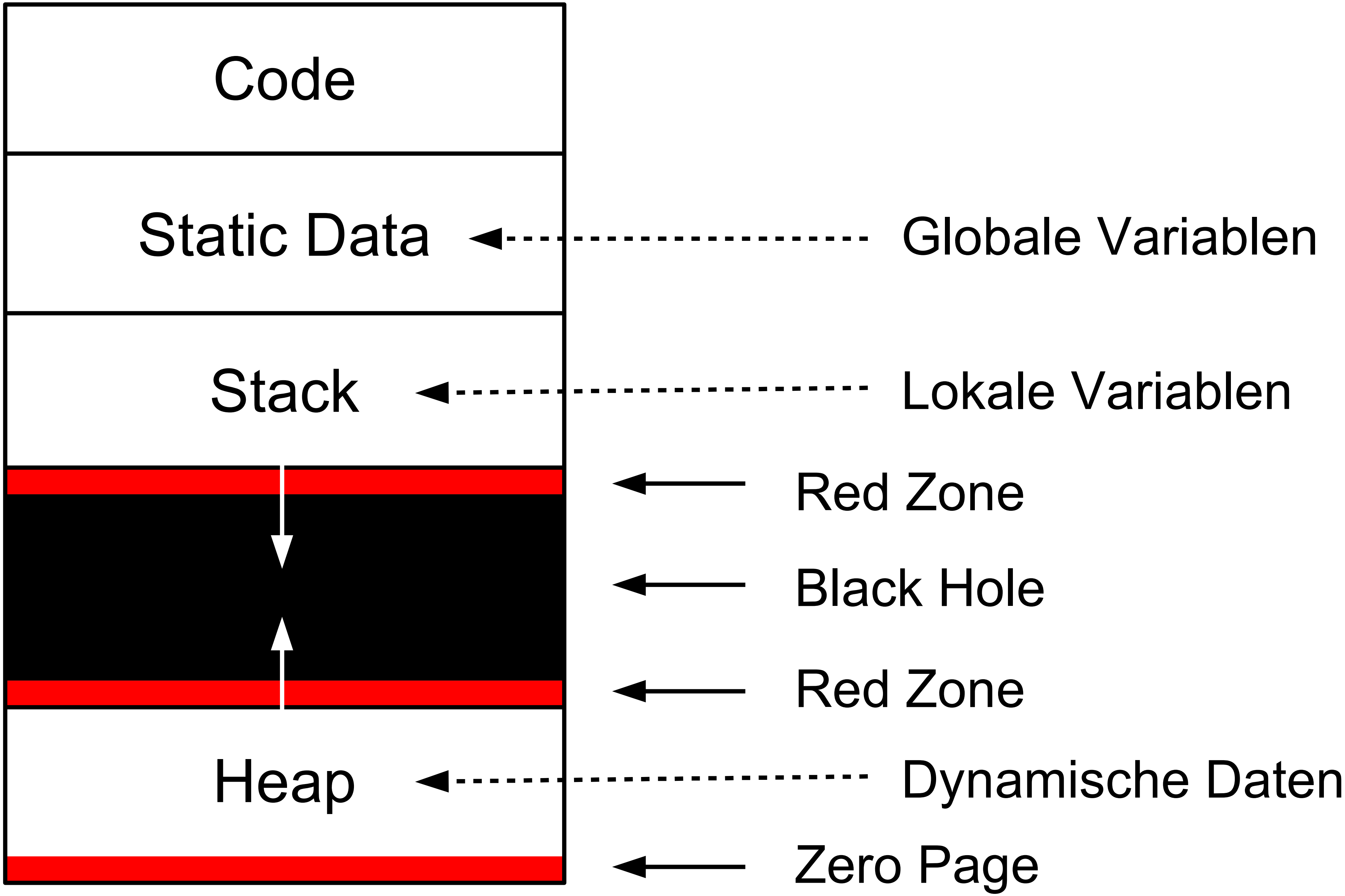
void *malloc(size_t size);
void *calloc(size_t nmemb, size_t size);

Entspricht dem **new** in Java

Freigeben von Speicher:

void free(**void** *ptr);

Anfordern und Freigeben kann in beliebiger Verzahnung geschehen und ist unabhängig von der Lebensdauer eines Stackframe


Code
Static Data
Globale Variablen
Stack
Lokale Variablen
Red Zone
Black Hole
Red Zone
Heap
Dynamische Daten
Zero Page

Datei field.h

```
#pragma once

#include "point.h"

typedef struct field Field;

Field * newField(int width, int height);
void freeField(Field *field);

int get(Field *field, Point point);

void increment(Field *field, Point point);
void decrement(Field *field, Point point);

Point size(Field *field);
```

Datei field.h

```
#pragma once

#include "point.h"

typedef struct field Field;

Field * newField(int width, int height);
void freeField(Field *field);

int get(Field *field, Point point);

void increment(Field *field, Point point);
void decrement(Field *field, Point point);

Point size(Field *field);
```

Datei field.h

```
#pragma once

#include "point.h"

typedef struct field Field;

Field * newField(int width, int height);
void freeField(Field *field);

int get(Field *field, Point point);

void increment(Field *field, Point point);
void decrement(Field *field, Point point);

Point size(Field *field);
```

Datei field.c

```
...

struct field{
    int **panel;
    int width;
    int height;
};

…
```

Datei field.c

```
…

Field * newField(int width, int height){

    Field *field= calloc(1, sizeof(Field));
    field->width= width;
    field->height= height;
    field->panel= (int **) calloc(height, sizeof(int*));
    for(int i=0; i<height; i++){
        field->panel[i]= (int*) calloc(width, sizeof(int));
    }
    return field;
}

…
```

Datei field.c

```
...

 void freeField(Field *field){

 for (int i=0; i<field->height; i++){
     free(field->panel[i]);
 }
     free(field->panel);
     free(field);
     field= NULL;
 }

 …
```

Datei Snake.h

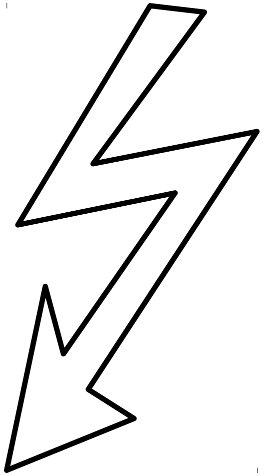
```
…

typedef struct snake Snake;

Snake *newSnake();
void freeSnake(Snake *snake);

int length(Snake *list);

void *getFirst(Snake *list);
void *getLast(Snake *list);

void pushFirst(Snake *list, void *content);
void popLast(Snake *snake);

…
```

Datei Snake.c

```
#include “node.h”
…

struct Snake{
    int length;
    Node *head, *tail;
};

Snake *newSnake(){
    Snake *snake= (Snake *) calloc(1, sizeof(Snake));
    return snake;
}

…
```

Datei Snake.c

```
…

void freeSnake(Snake * snake){

    Node head= snake->head;
    for(; head!=NULL; head= next(head)){
        freeNode(head);
    }
    snake= NULL;
}

…
```



Datei Snake.c

```
…

void freeSnake(Snake * snake){

    Node head= snake->head;
    Node temp;
    for(; head!=NULL; head= temp){
        temp= next(head);
        freeNode(head);
    }
    snake= NULL;
}

…
```

Danke!

Danke!
Fragen?

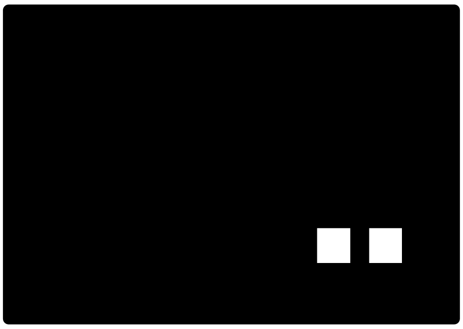
Freitagsrunde C-Kurs 2011

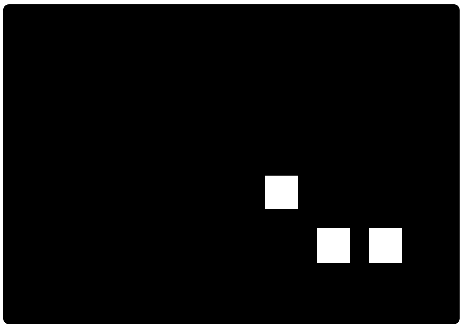
Speicherverwaltung in c

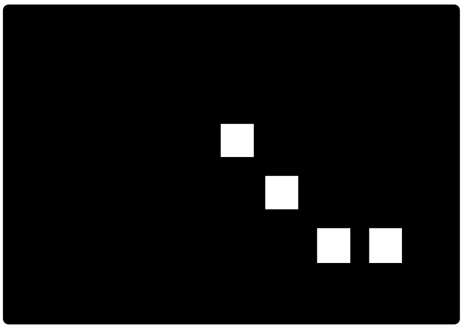
Vorlesung 2 – 14.09.2011
Katrin Lang
katrin.lang@tu-berlin.de

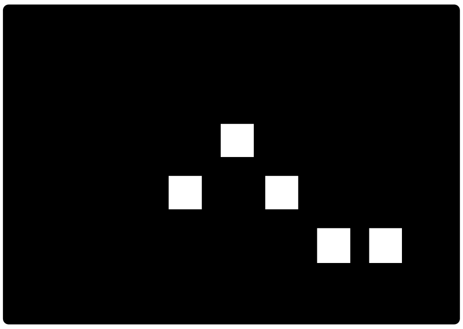
Simulation:
Spielfeld: 9x6 Kästchen
Länge der Schlange: 10

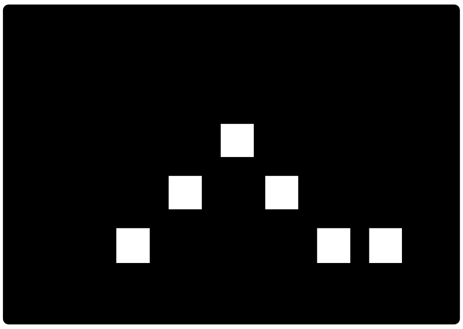
Datenstrukturen für den Screensaver

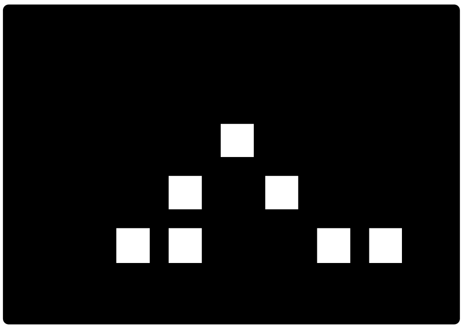


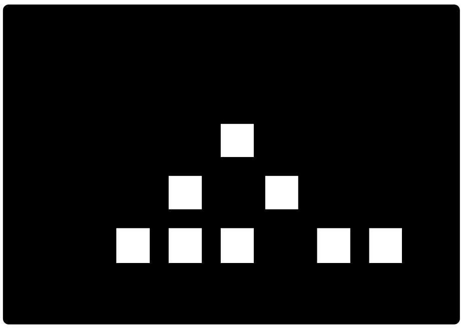


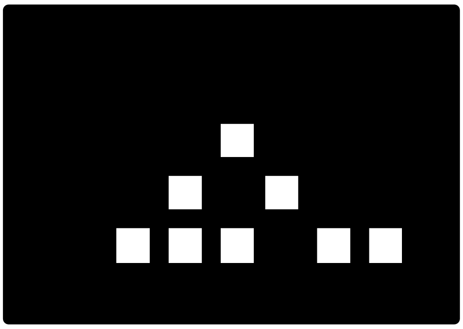


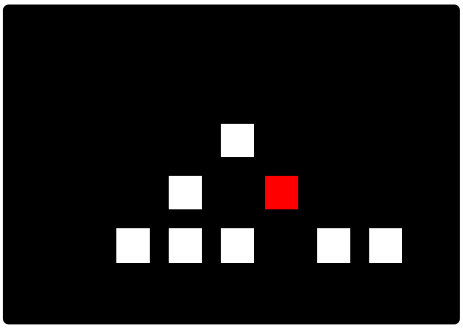


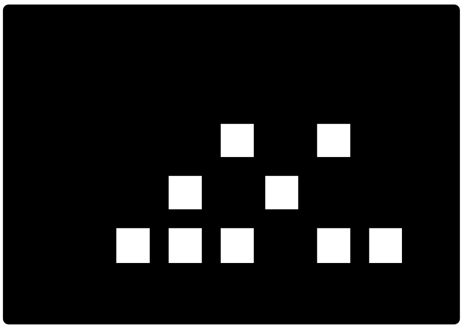


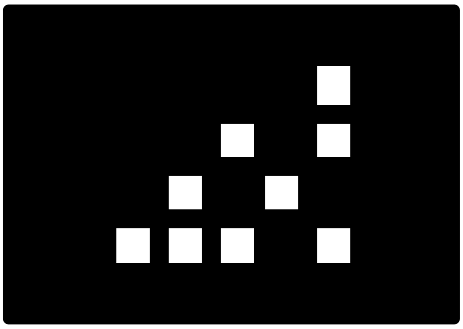


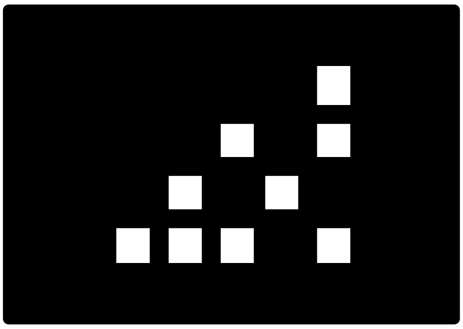












Meine Implementierung:

- Ein 2D-Array für das Spielfeld
- Eine doppelt verkettete Liste für die Schlange

Der Compiler kennt den Speicherbedarf unseres Spiels nicht, denn:

- Bildschirmgröße variiert
- Verkleinern/Vergrößern des Fensters möglich

Können wir den Speicherbedarf statisch ermitteln?

Der Compiler kennt den Speicherbedarf unseres Spiels nicht, denn:

- Bildschirmgröße variiert
- Verkleinern/Vergrößern des Fensters möglich

  ➜ Größe des Spielfelds muss zur Laufzeit vom Betriebssystem erfragt werden

Können wir den Speicherbedarf statisch ermitteln?

Der Compiler kennt den Speicherbedarf unseres Spiels nicht, denn:

- Bildschirmgröße variiert
- Verkleinern/Vergrößern des Fensters möglich

  ➜ Größe des Spielfelds muss zur Laufzeit vom Betriebssystem erfragt werden

- Die verkettete Liste erfordert ständiges Neuanlegen bzw. Wegwerfen von Listenelementen

Können wir den Speicherbedarf statisch ermitteln?

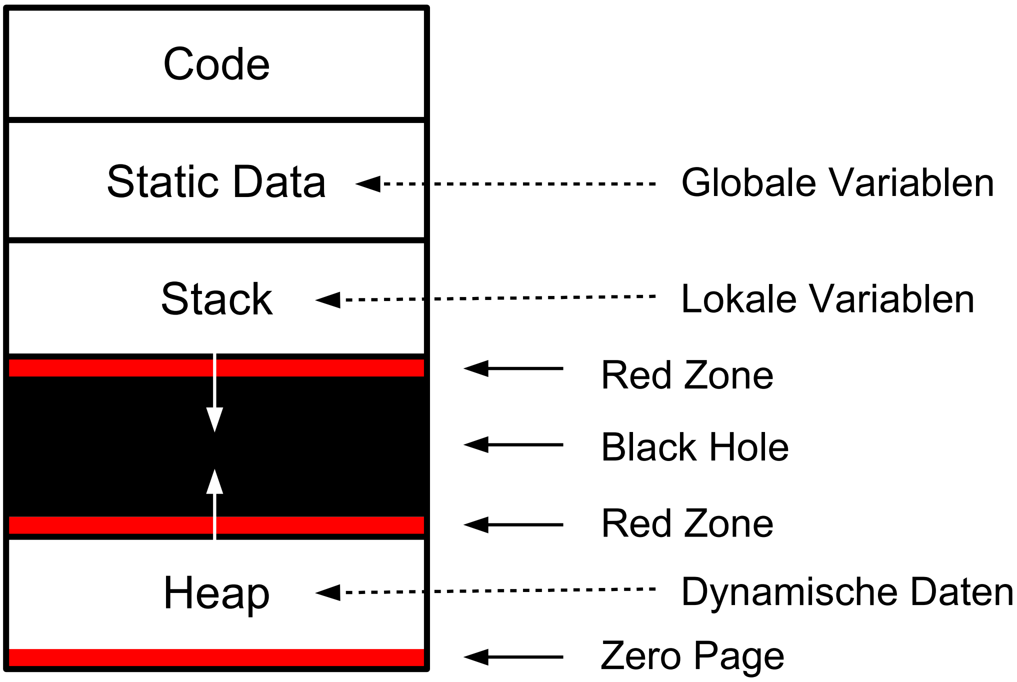
Der Compiler kennt den Speicherbedarf unseres Spiels nicht, denn:

- Bildschirmgröße variiert
- Verkleinern/Vergrößern des Fensters möglich

  ➜ Größe des Spielfelds muss zur Laufzeit vom Betriebssystem erfragt werden
- Die verkettete Liste erfordert ständiges Neuanlegen bzw. Wegwerfen von Listenelementen

  ➜ Das kann nur das Laufzeitsystem leisten

Können wir den Speicherbedarf statisch ermitteln?



Code

Static Data

Stack

Heap

Globale Variablen

Lokale Variablen

Red Zone

Black Hole

Red Zone

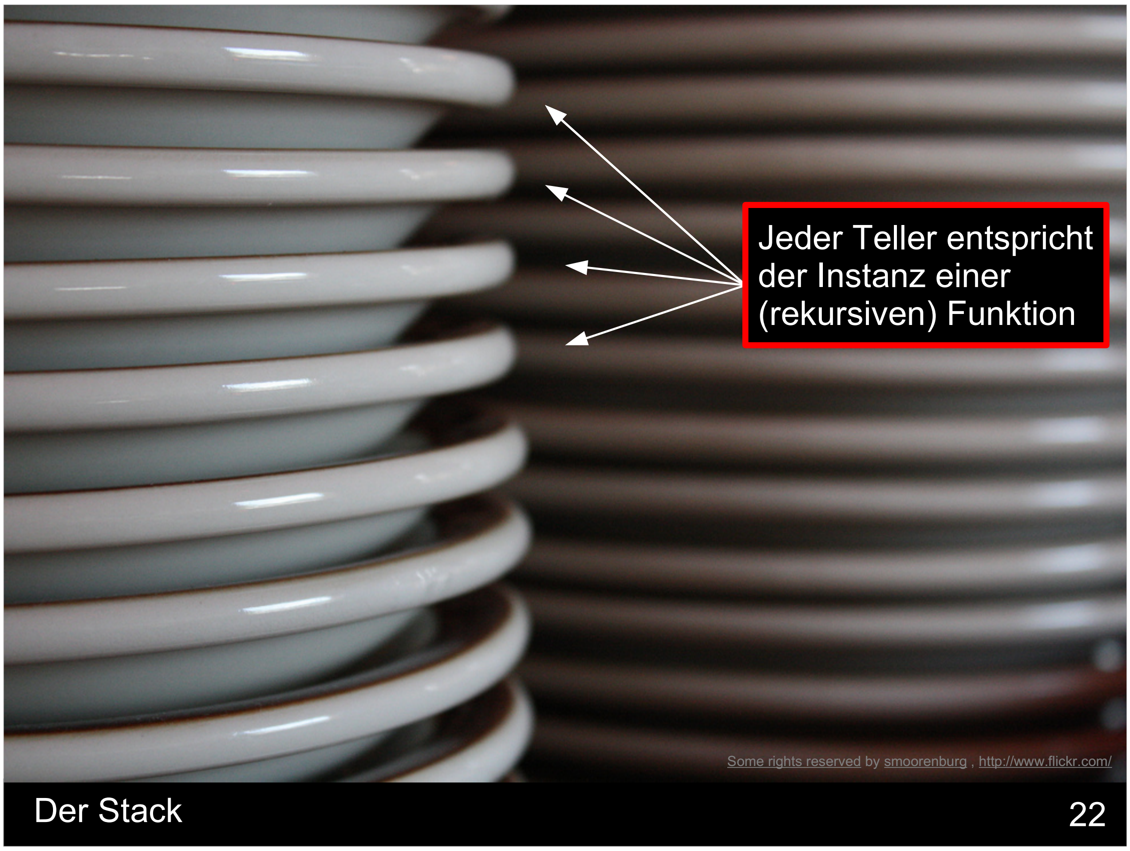
Dynamische Daten

Zero Page

Speicherorganisation eines Prozesses

Some rights reserved by smoorenburg , http://www.flickr.com/

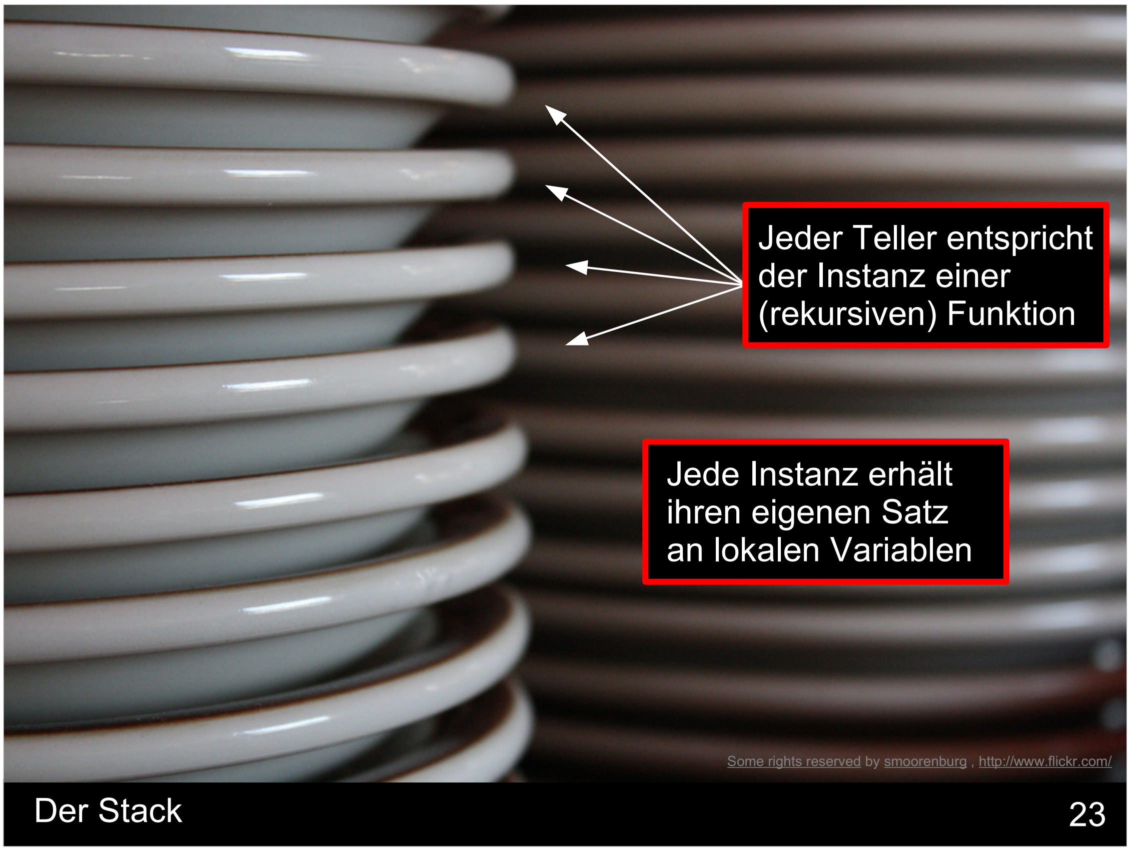
Der Stack



Jeder Teller entspricht der Instanz einer (rekursiven) Funktion

Some rights reserved by smoorenburg , http://www.flickr.com/

Der Stack

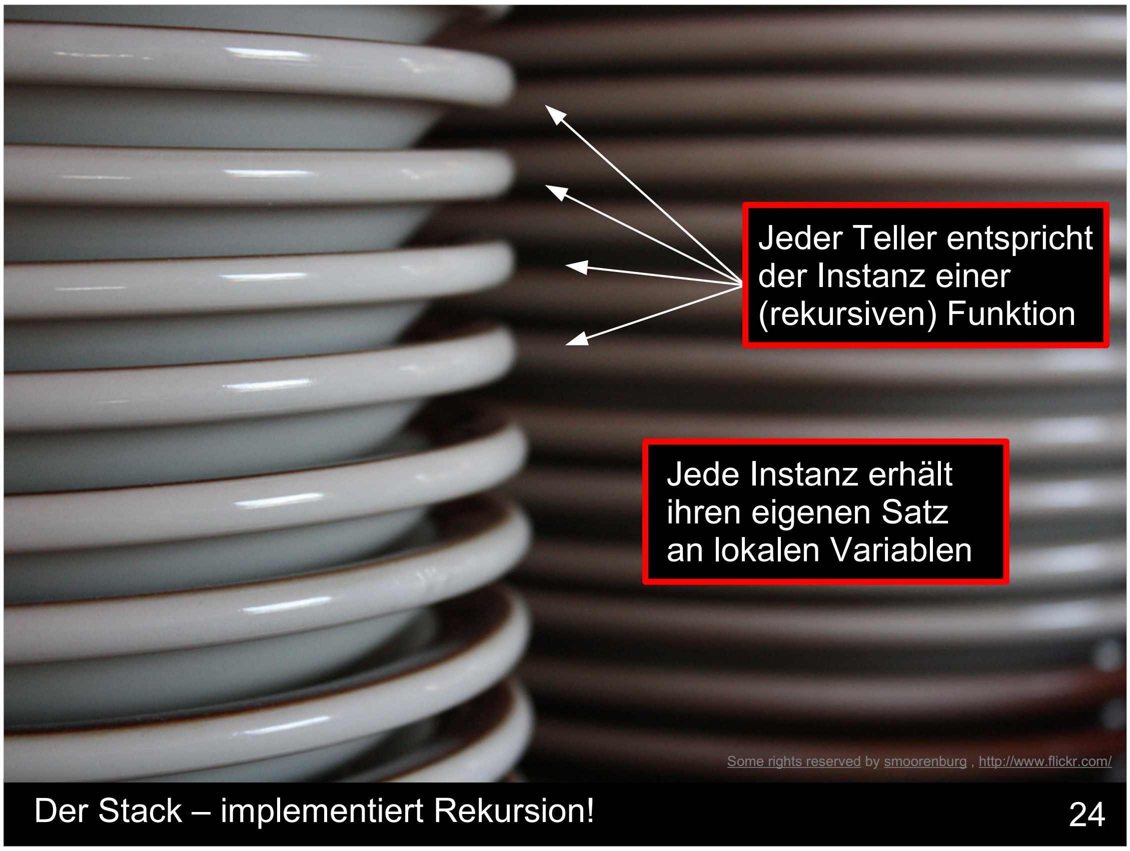


Jeder Teller entspricht der Instanz einer (rekursiven) Funktion

Jede Instanz erhält ihren eigenen Satz an lokalen Variablen

Some rights reserved by smoorenburg , http://www.flickr.com/

Der Stack



Jeder Teller entspricht der Instanz einer (rekursiven) Funktion

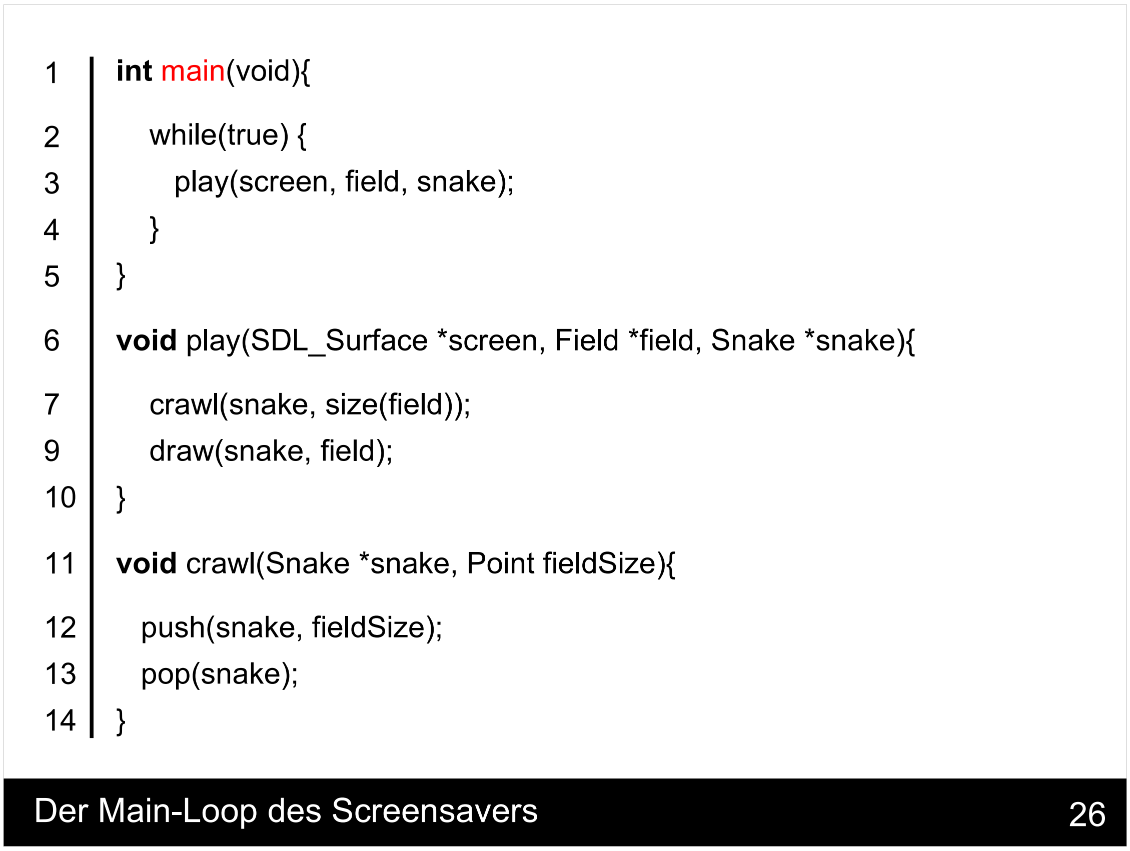
Jede Instanz erhält ihren eigenen Satz an lokalen Variablen

Some rights reserved by smoorenburg , http://www.flickr.com/

Der Stack – implementiert Rekursion!

```
int main(void){
   while(true) {
      play(screen, field, snake);
   }
}
void play(SDL_Surface *screen, Field *field, Snake *snake){
   crawl(snake, size(field));
   draw(snake, field);
}
void crawl(Snake *snake, Point fieldSize){
  push(snake, fieldSize);
  pop(snake);
}
```

Der Main-Loop des Screensavers



```
int main(void){
   while(true) {
     play(screen, field, snake);
   }
}
void play(SDL_Surface *screen, Field *field, Snake *snake){
   crawl(snake, size(field));
   draw(snake, field);
}
void crawl(Snake *snake, Point fieldSize){
  push(snake, fieldSize);
  pop(snake);
}
```

Der Main-Loop des Screensavers

```
int main(void){
    while(true) {
       play(screen, field, snake);
    }
}
void play(SDL_Surface *screen, Field *field, Snake *snake){
    crawl(snake, size(field));
    draw(snake, field);
}
void crawl(Snake *snake, Point fieldSize){
   push(snake, fieldSize);
   pop(snake);
}
```

Der Main-Loop des Screensavers

```
int main(void){
    while(true) {
       play(screen, field, snake);
    }
}
void play(SDL_Surface *screen, Field *field, Snake *snake){
    crawl(snake, size(field));
    draw(snake, field);
}
void crawl(Snake *snake, Point fieldSize){
   push(snake, fieldSize);
   pop(snake);
}
```

Der Main-Loop des Screensavers

```
int main(void){
   while(true) {
      play(screen, field, snake);
   }
}
void play(SDL_Surface *screen, Field *field, Snake *snake){
   crawl(snake, size(field));
   draw(snake, field);
}
void crawl(Snake *snake, Point fieldSize){
  push(snake, fieldSize);
  pop(snake);
}
```

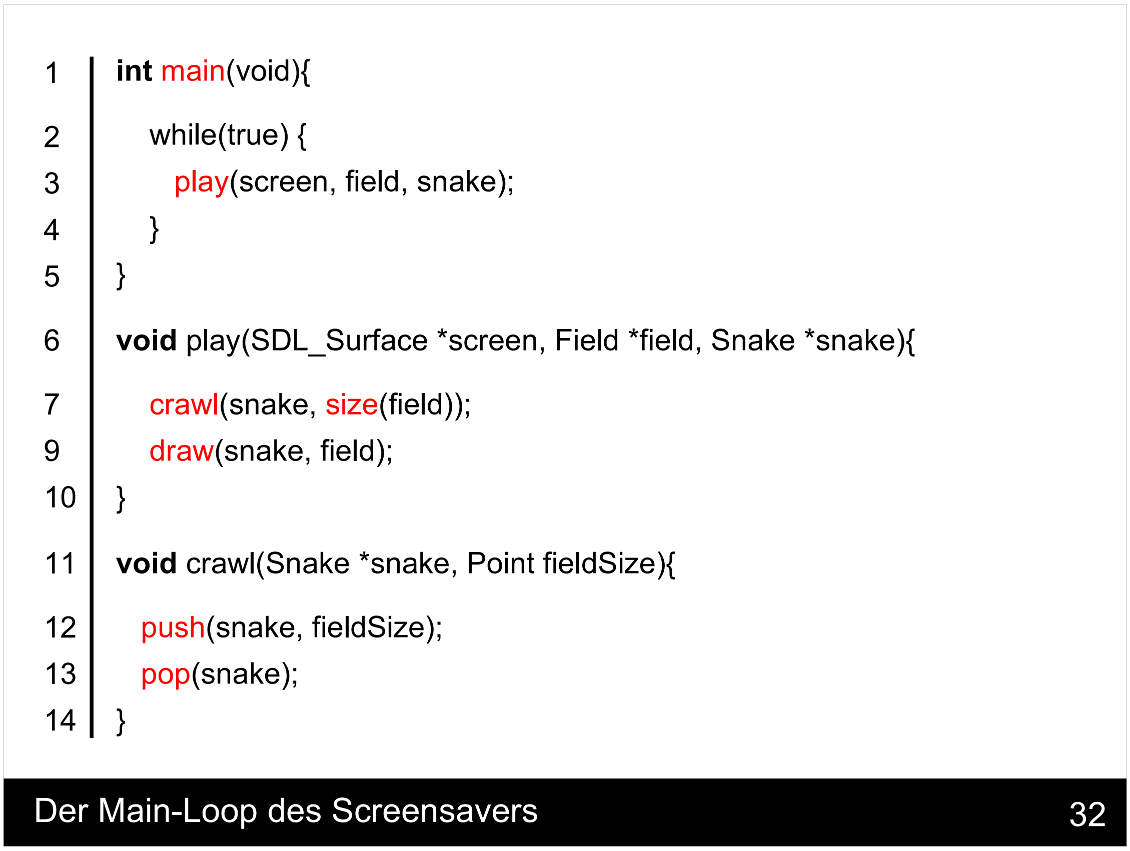
Der Main-Loop des Screensavers

```
int main(void){
   while(true) {
     play(screen, field, snake);
   }
}
void play(SDL_Surface *screen, Field *field, Snake *snake){
   crawl(snake, size(field));
   draw(snake, field);
}
void crawl(Snake *snake, Point fieldSize){
  push(snake, fieldSize);
  pop(snake);
}
```

Der Main-Loop des Screensavers

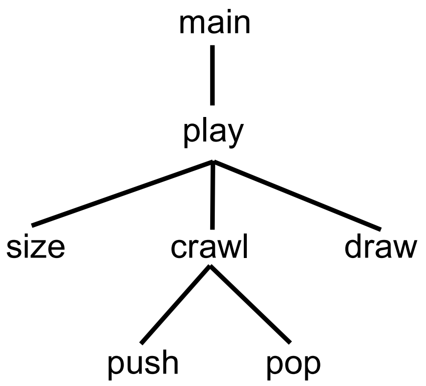
```
int main(void){
    while(true) {
       play(screen, field, snake);
    }
}
void play(SDL_Surface *screen, Field *field, Snake *snake){
    crawl(snake, size(field));
    draw(snake, field);
}
void crawl(Snake *snake, Point fieldSize){
   push(snake, fieldSize);
   pop(snake);
}
```

Der Main-Loop des Screensavers

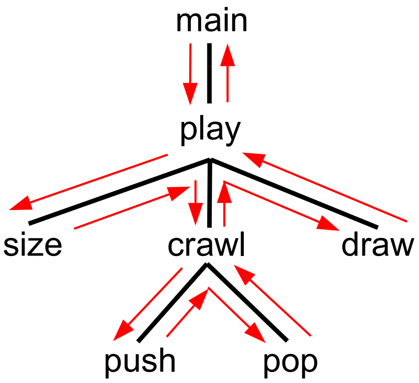


```
int main(void){
   while(true) {
      play(screen, field, snake);
   }
}
void play(SDL_Surface *screen, Field *field, Snake *snake){
   crawl(snake, size(field));
   draw(snake, field);
}
void crawl(Snake *snake, Point fieldSize){
  push(snake, fieldSize);
  pop(snake);
}
```

Der Main-Loop des Screensavers

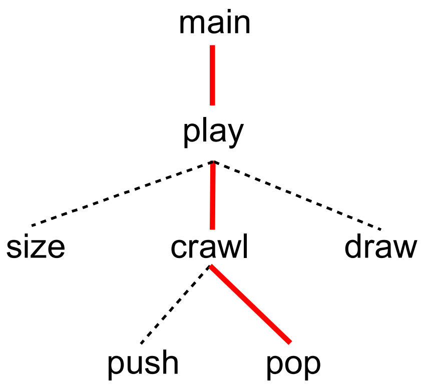


Der Call Tree des Screensavers



Preorder-Traversierung == Kontrollfluss

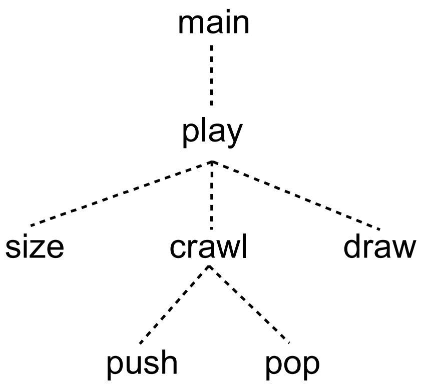
Der Call Tree des Screensavers



Pfad == Zustand des Stacks

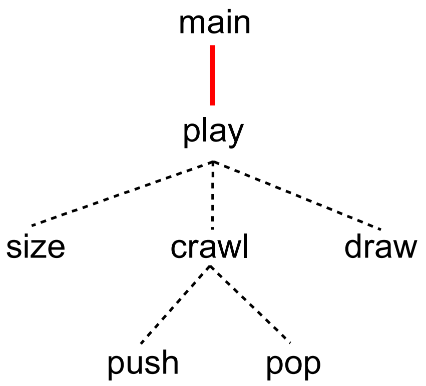
Preorder-Traversierung == Kontrollfluss

Der Call Tree des Screensavers



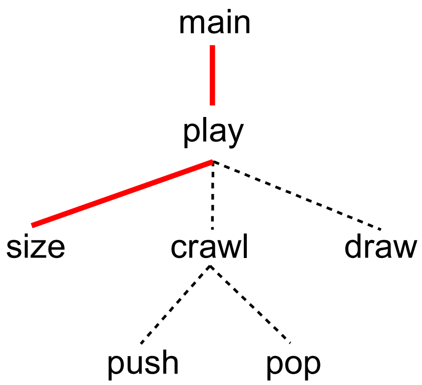
main

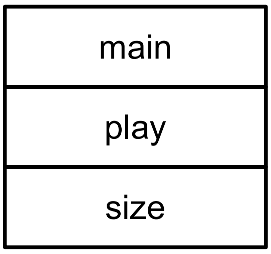
Der Call Tree des Screensavers



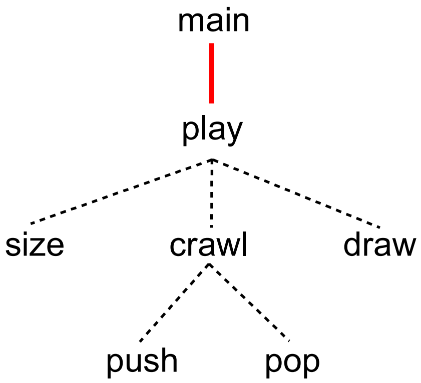
main
play

Der Call Tree des Screensavers



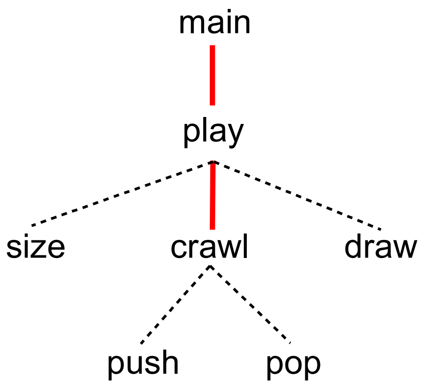


Der Call Tree des Screensavers



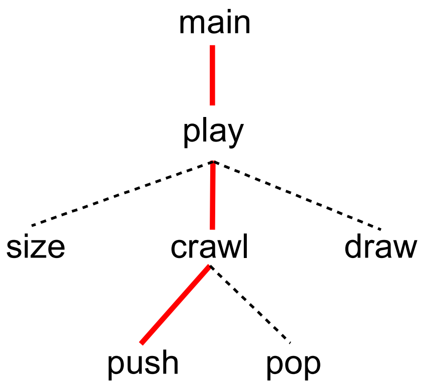
main
play

Der Call Tree des Screensavers



main
play
crawl

Der Call Tree des Screensavers



main

play

size

crawl

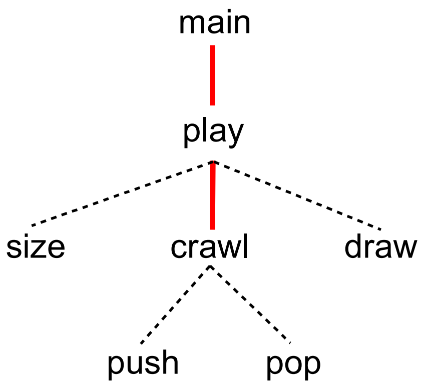
draw

push

pop

main
play
crawl
push

Der Call Tree des Screensavers

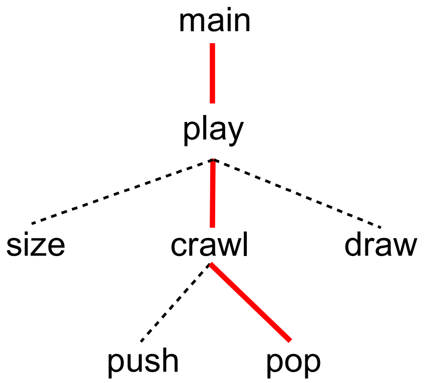


main

play

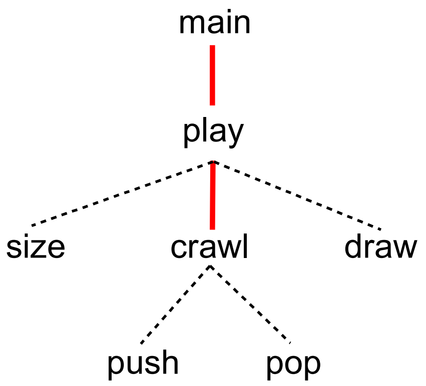
crawl

Der Call Tree des Screensavers



main
play
crawl
pop

Der Call Tree des Screensavers

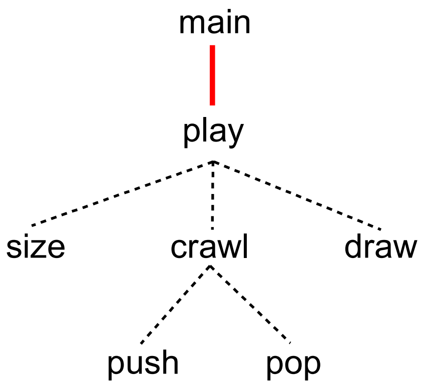


main

play

crawl

Der Call Tree des Screensavers



main

play

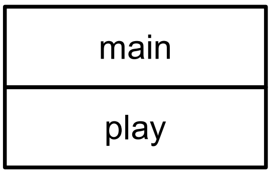
size

crawl

draw

push

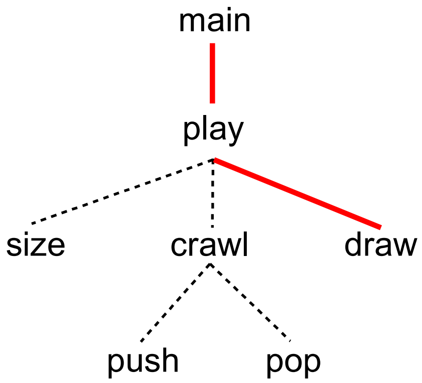
pop



main

play

Der Call Tree des Screensavers

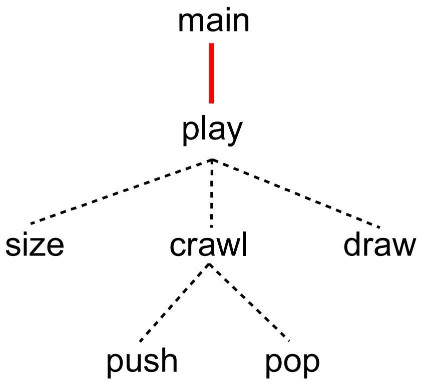


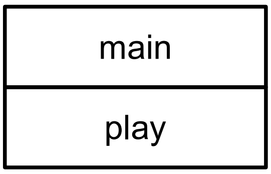
main

play

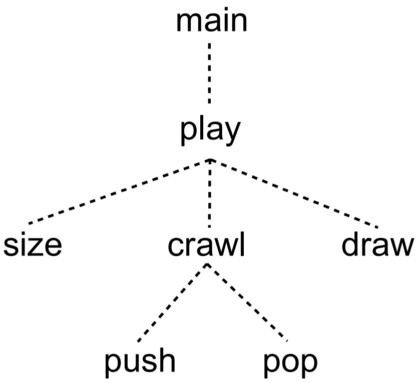
draw

Der Call Tree des Screensavers





Der Call Tree des Screensavers



main

play

size

crawl

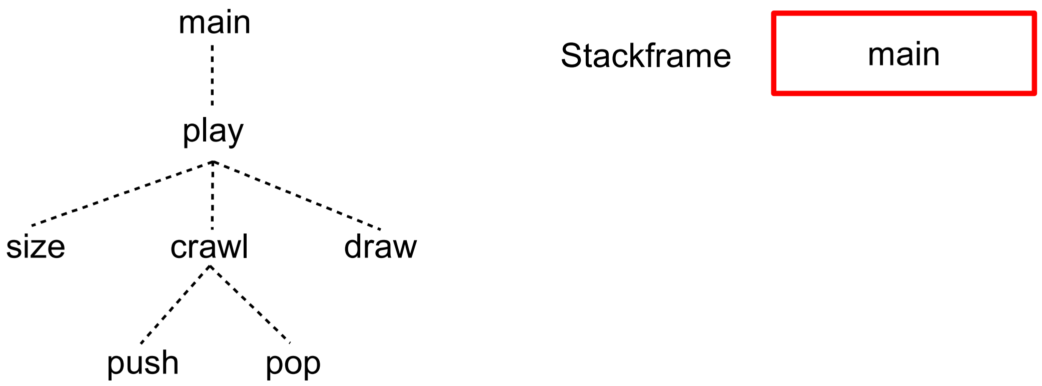
draw

push

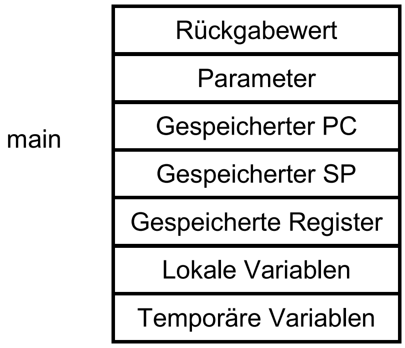
pop

main

Der Call Tree des Screensavers



Der Call Tree des Screensavers



main

Rückgabewert

Parameter

Gespeicherter PC

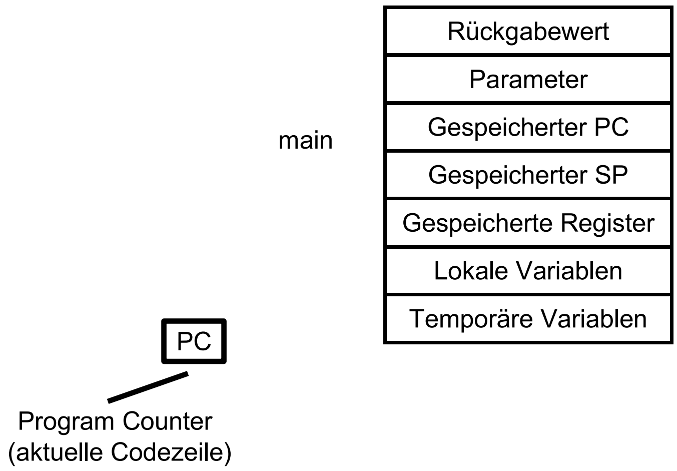
Gespeicherter SP

Gespeicherte Register

Lokale Variablen

Temporäre Variablen

Stack Frame



main

Rückgabewert

Parameter

Gespeicherter PC

Gespeicherter SP

Gespeicherte Register

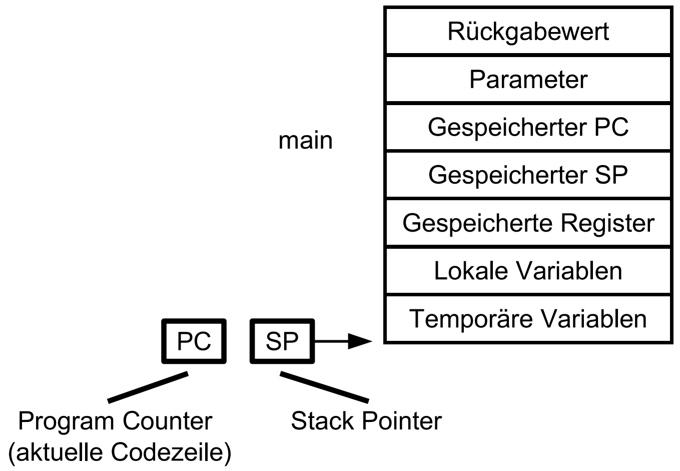
Lokale Variablen

Temporäre Variablen

PC

Program Counter
(aktuelle Codezeile)

Stack Frame



main

Rückgabewert

Parameter

Gespeicherter PC

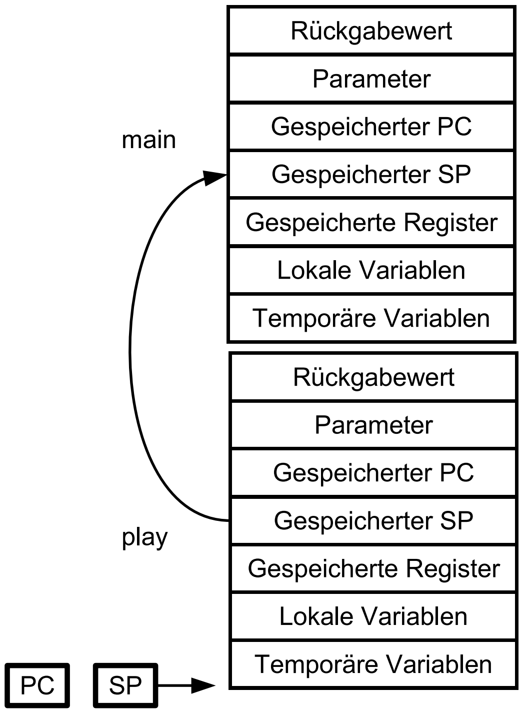
Gespeicherter SP

Gespeicherte Register

Lokale Variablen

Temporäre Variablen

Stack Frame



main

Rückgabewert

Parameter

Gespeicherter PC

Gespeicherter SP

Gespeicherte Register

Lokale Variablen

Temporäre Variablen

play

Rückgabewert

Parameter

Gespeicherter PC

Gespeicherter SP

Gespeicherte Register

Lokale Variablen

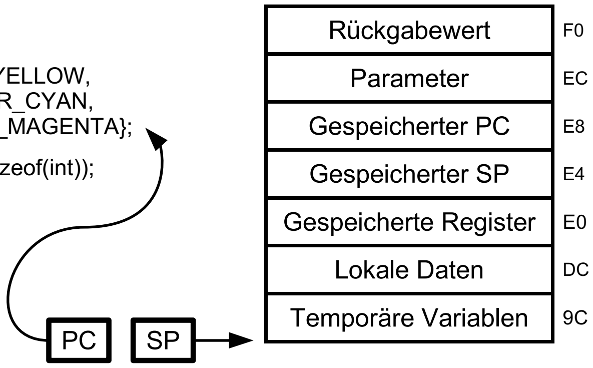
Temporäre Variablen

PC

SP

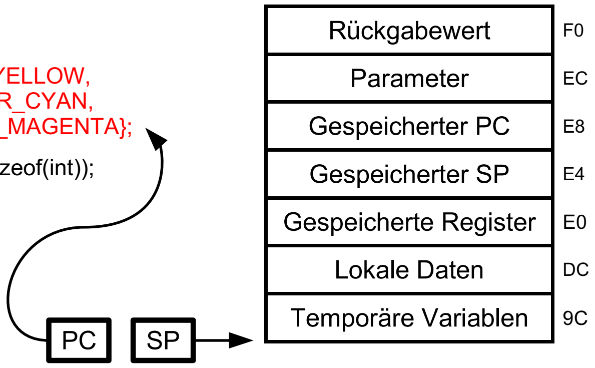
Stack Frame

```
int main(void){

   int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                   COLOR_GREEN, COLOR_CYAN,
                   COLOR_BLUE, COLOR_MAGENTA};
   printRainbow(rainbow, sizeof(rainbow)/sizeof(int));
}



void printRainbow(int rainbow[], int rainbowLength){

    for(int i= 0; i<rainbowLength; i++){
       attron(COLOR_PAIR(rainbow[i]));
       addstr("\u2588");
       attroff(COLOR_PAIR(rainbow[i]));
   }
}
```



Rainbow revisited

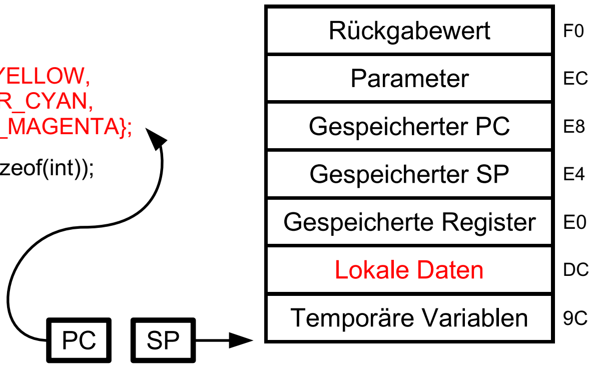
```
int main(void){

    int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                         COLOR_GREEN, COLOR_CYAN,
                         COLOR_BLUE, COLOR_MAGENTA};

    printRainbow(rainbow, sizeof(rainbow)/sizeof(int));
}




void printRainbow(int rainbow[], int rainbowLength){

     for(int i= 0; i<rainbowLength; i++){
         attron(COLOR_PAIR(rainbow[i]));
         addstr("\u2588");
         attroff(COLOR_PAIR(rainbow[i]));
     }
}
```



Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
Lokale Daten	DC
Temporäre Variablen	9C

Rainbow revisited

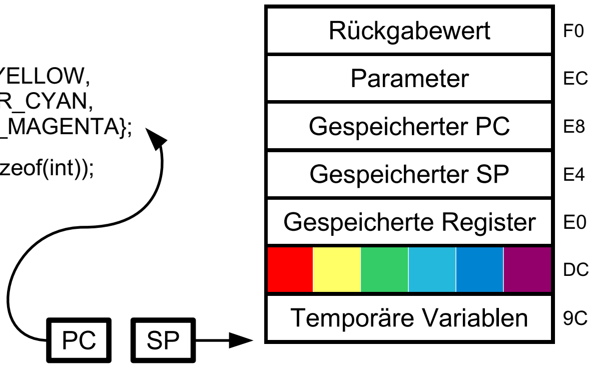
```
int main(void){

    int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                    COLOR_GREEN, COLOR_CYAN,
                    COLOR_BLUE, COLOR_MAGENTA};

    printRainbow(rainbow, sizeof(rainbow)/sizeof(int));
}


void printRainbow(int rainbow[], int rainbowLength){

    for(int i= 0; i<rainbowLength; i++){
        attron(COLOR_PAIR(rainbow[i]));
        addstr("\u2588");
        attroff(COLOR_PAIR(rainbow[i]));
    }
}
```



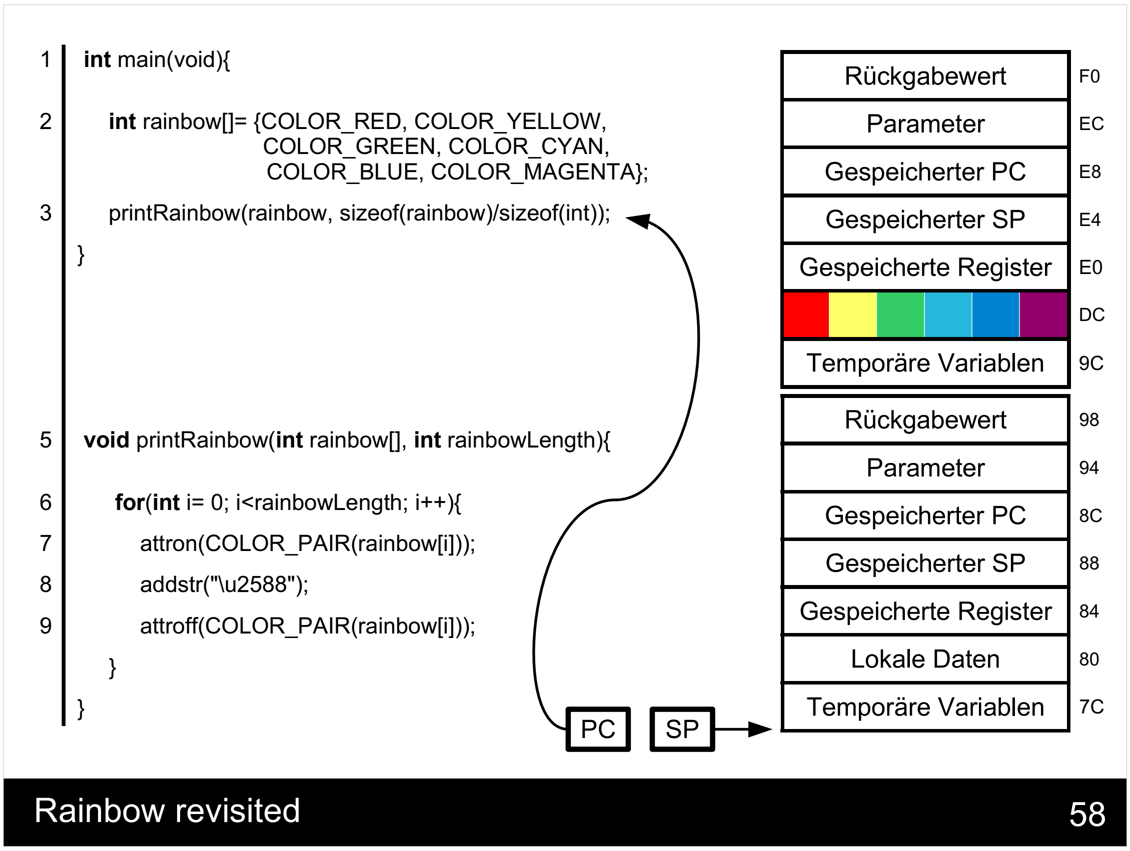
Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
Lokale Daten	DC
Temporäre Variablen	9C

Rainbow revisited

```
int main(void){

    int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                    COLOR_GREEN, COLOR_CYAN,
                    COLOR_BLUE, COLOR_MAGENTA};

    printRainbow(rainbow, sizeof(rainbow)/sizeof(int));
}


void printRainbow(int rainbow[], int rainbowLength){

    for(int i= 0; i<rainbowLength; i++){
        attron(COLOR_PAIR(rainbow[i]));
        addstr("\u2588");
        attroff(COLOR_PAIR(rainbow[i]));
    }
}
```



Rainbow revisited

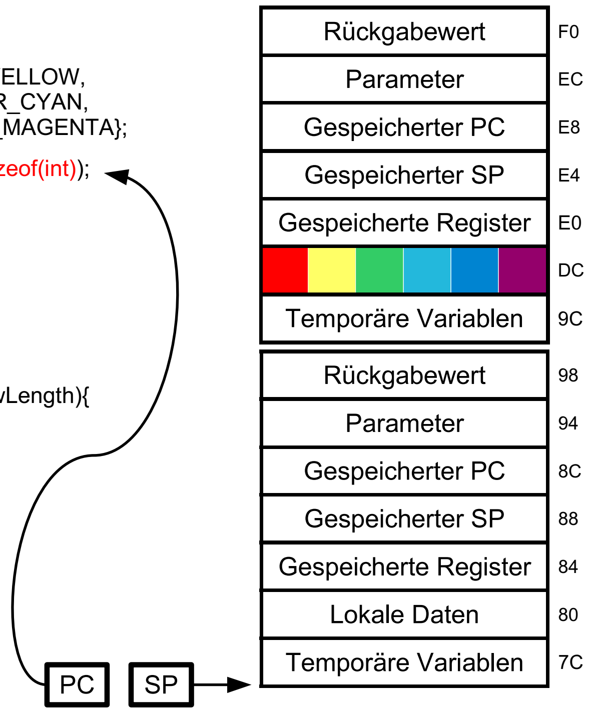


```
int main(void){

    int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                         COLOR_GREEN, COLOR_CYAN,
                         COLOR_BLUE, COLOR_MAGENTA};

    printRainbow(rainbow, sizeof(rainbow)/sizeof(int));
}



void printRainbow(int rainbow[], int rainbowLength){

    for(int i= 0; i<rainbowLength; i++){
        attron(COLOR_PAIR(rainbow[i]));
        addstr("\u2588");
        attroff(COLOR_PAIR(rainbow[i]));
    }
}
```

Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
🟥🟨🟩🟦🟦🟪	DC
Temporäre Variablen	9C
Rückgabewert	98
Parameter	94
Gespeicherter PC	8C
Gespeicherter SP	88
Gespeicherte Register	84
Lokale Daten	80
Temporäre Variablen	7C

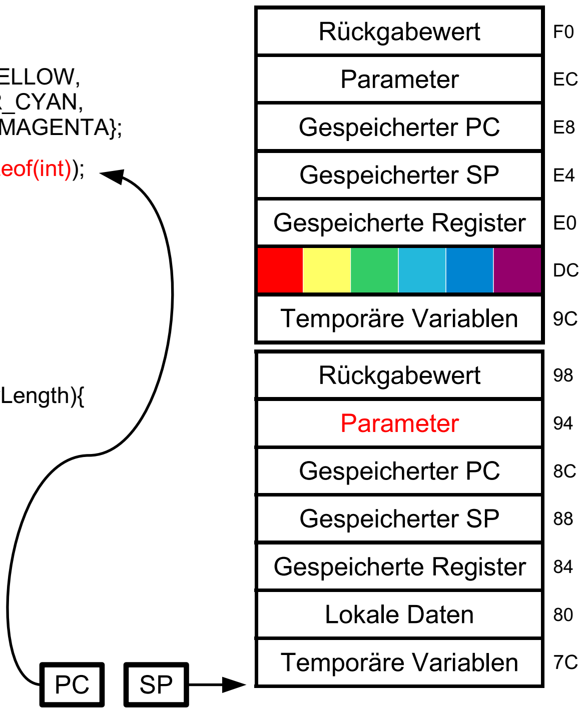
Rainbow revisited

```
int main(void){

    int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                    COLOR_GREEN, COLOR_CYAN,
                    COLOR_BLUE, COLOR_MAGENTA};

    printRainbow(rainbow, sizeof(rainbow)/sizeof(int));
}



void printRainbow(int rainbow[], int rainbowLength){

    for(int i= 0; i<rainbowLength; i++){
        attron(COLOR_PAIR(rainbow[i]));
        addstr("\u2588");
        attroff(COLOR_PAIR(rainbow[i]));
    }
}
```



Rainbow revisited

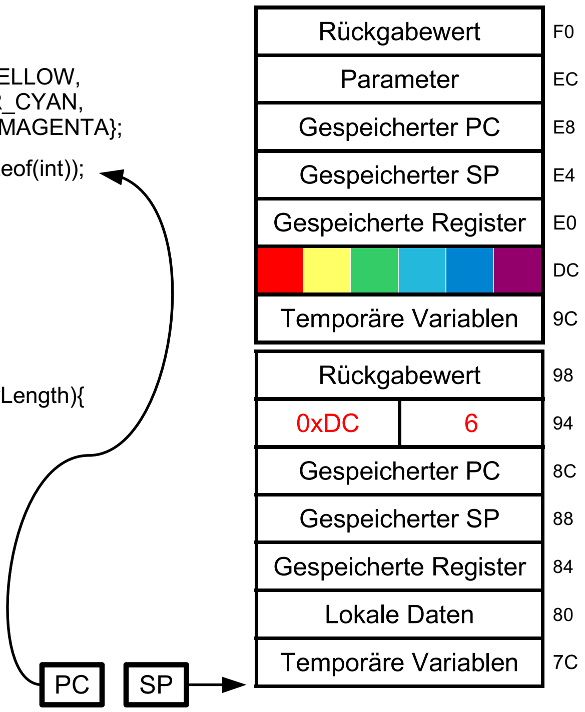
```
int main(void){

    int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                    COLOR_GREEN, COLOR_CYAN,
                    COLOR_BLUE, COLOR_MAGENTA};

    printRainbow(rainbow, sizeof(rainbow)/sizeof(int));
}


void printRainbow(int rainbow[], int rainbowLength){

    for(int i= 0; i<rainbowLength; i++){
        attron(COLOR_PAIR(rainbow[i]));
        addstr("\u2588");
        attroff(COLOR_PAIR(rainbow[i]));
    }
}
```



Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
🟥🟨🟩🟦🟦🟪	DC
Temporäre Variablen	9C
Rückgabewert	98
Parameter	94
Gespeicherter PC	8C
Gespeicherter SP	88
Gespeicherte Register	84
Lokale Daten	80
Temporäre Variablen	7C

Rainbow revisited

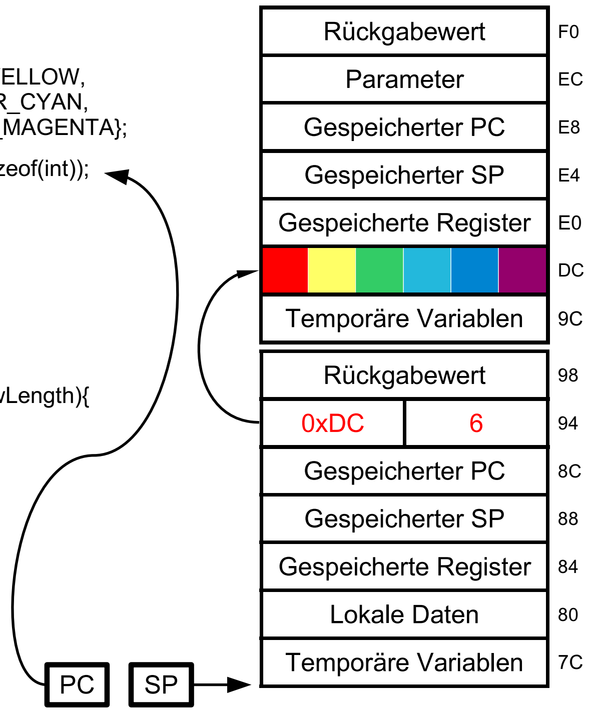
```
int main(void){

    int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                    COLOR_GREEN, COLOR_CYAN,
                    COLOR_BLUE, COLOR_MAGENTA};

    printRainbow(rainbow, sizeof(rainbow)/sizeof(int));
}




void printRainbow(int rainbow[], int rainbowLength){

     for(int i= 0; i<rainbowLength; i++){
         attron(COLOR_PAIR(rainbow[i]));
         addstr("\u2588");
         attroff(COLOR_PAIR(rainbow[i]));
     }
}
```



Rückgabewert		F0
Parameter		EC
Gespeicherter PC		E8
Gespeicherter SP		E4
Gespeicherte Register		E0
🟥🟨🟩🟦🟦🟪		DC
Temporäre Variablen		9C
Rückgabewert		98
0xDC	6	94
Gespeicherter PC		8C
Gespeicherter SP		88
Gespeicherte Register		84
Lokale Daten		80
Temporäre Variablen		7C

Rainbow revisited

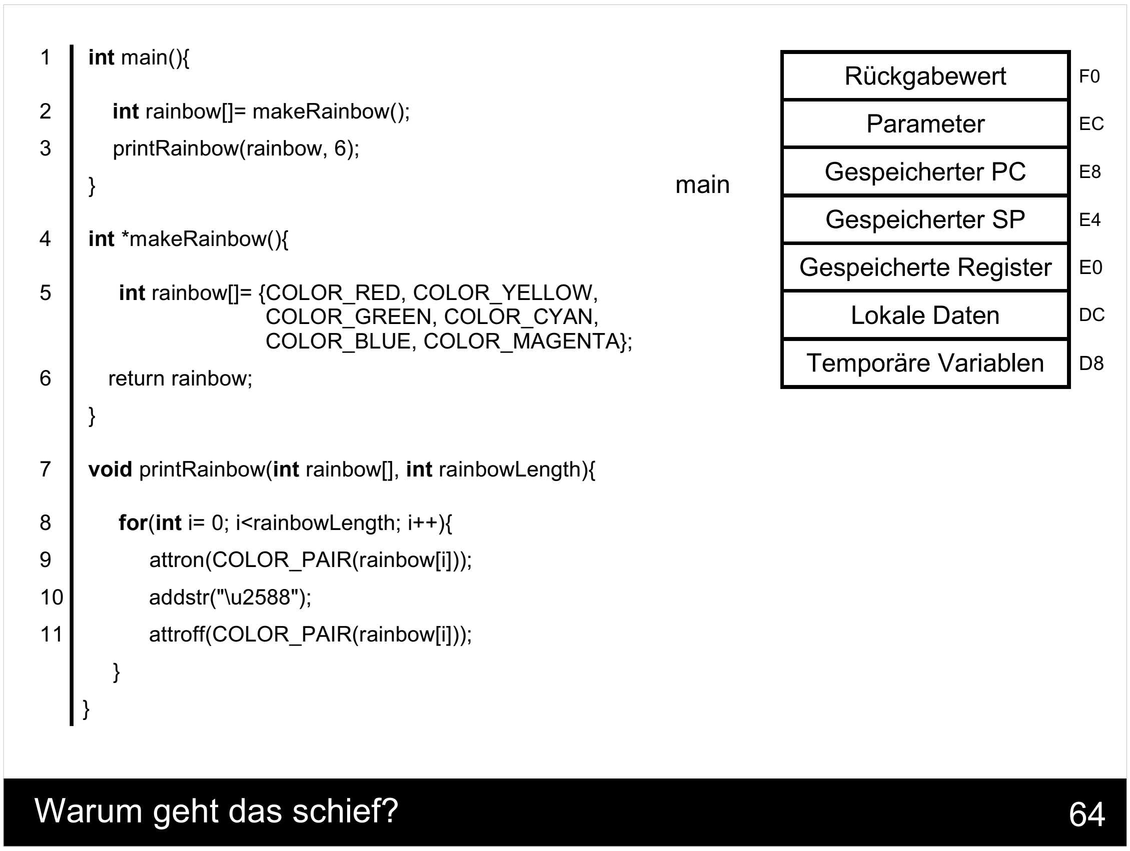
```
int main(void){

   int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                   COLOR_GREEN, COLOR_CYAN,
                   COLOR_BLUE, COLOR_MAGENTA};

   printRainbow(rainbow, sizeof(rainbow)/sizeof(int));
}



void printRainbow(int rainbow[], int rainbowLength){

    for(int i= 0; i<rainbowLength; i++){
       attron(COLOR_PAIR(rainbow[i]));
       addstr("\u2588");
       attroff(COLOR_PAIR(rainbow[i]));
    }
}
```



Rainbow revisited

```
int main(){
    int rainbow[]= makeRainbow();
    printRainbow(rainbow, 6);
}
int *makeRainbow(){
     int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                     COLOR_GREEN, COLOR_CYAN,
                     COLOR_BLUE, COLOR_MAGENTA};
   return rainbow;
}
void printRainbow(int rainbow[], int rainbowLength){
     for(int i= 0; i<rainbowLength; i++){
          attron(COLOR_PAIR(rainbow[i]));
          addstr("\u2588");
          attroff(COLOR_PAIR(rainbow[i]));
     }
}
```

Warum geht das schief?

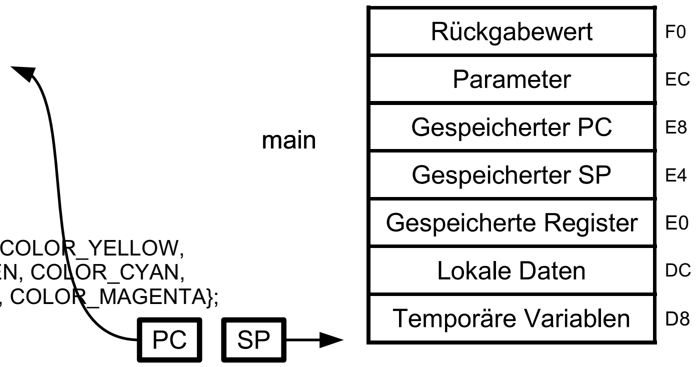


```
1   int main(){
2      int rainbow[]= makeRainbow();
3      printRainbow(rainbow, 6);
    }
4   int *makeRainbow(){
5       int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                        COLOR_GREEN, COLOR_CYAN,
                        COLOR_BLUE, COLOR_MAGENTA};
6     return rainbow;
    }
7   void printRainbow(int rainbow[], int rainbowLength){
8       for(int i= 0; i<rainbowLength; i++){
9           attron(COLOR_PAIR(rainbow[i]));
10          addstr("\u2588");
11          attroff(COLOR_PAIR(rainbow[i]));
       }
   }
```

	Rückgabewert	F0
	Parameter	EC
main	Gespeicherter PC	E8
	Gespeicherter SP	E4
	Gespeicherte Register	E0
	Lokale Daten	DC
	Temporäre Variablen	D8

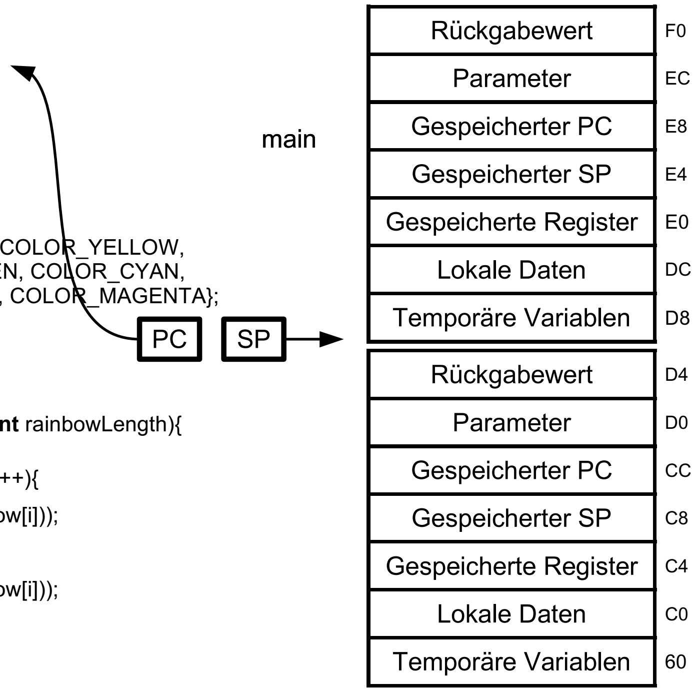
Warum geht das schief?

```
1   int main(){

2       int rainbow[]= makeRainbow();
3       printRainbow(rainbow, 6);
    }

4   int *makeRainbow(){

5        int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                         COLOR_GREEN, COLOR_CYAN,
                         COLOR_BLUE, COLOR_MAGENTA};

6      return rainbow;
    }

7   void printRainbow(int rainbow[], int rainbowLength){

8        for(int i= 0; i<rainbowLength; i++){
9            attron(COLOR_PAIR(rainbow[i]));
10           addstr("\u2588");
11           attroff(COLOR_PAIR(rainbow[i]));
         }
    }
```



Warum geht das schief?

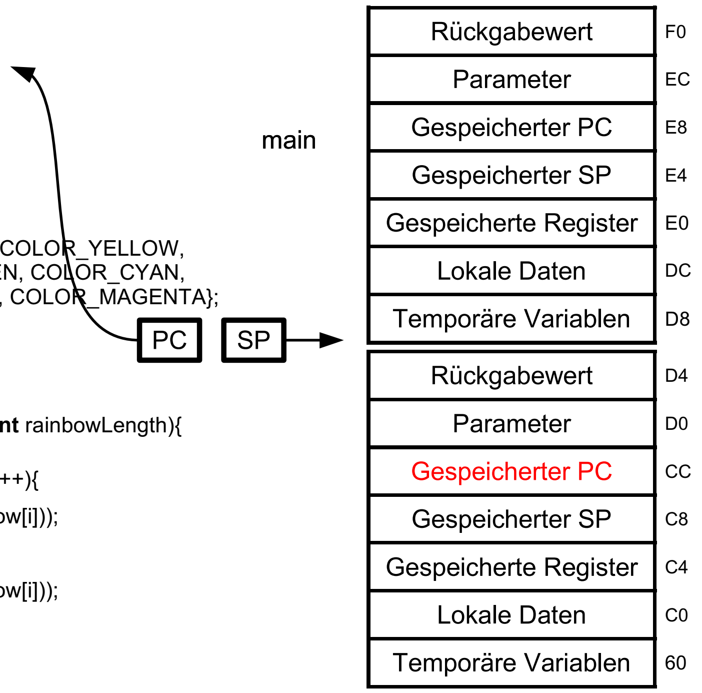
```
1    int main(){

2        int rainbow[]= makeRainbow();
3        printRainbow(rainbow, 6);
     }

4    int *makeRainbow(){

5         int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                          COLOR_GREEN, COLOR_CYAN,
                          COLOR_BLUE, COLOR_MAGENTA};
6       return rainbow;
     }

7    void printRainbow(int rainbow[], int rainbowLength){

8         for(int i= 0; i<rainbowLength; i++){
9             attron(COLOR_PAIR(rainbow[i]));
10            addstr("\u2588");
11            attroff(COLOR_PAIR(rainbow[i]));
          }
     }
```



Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
Lokale Daten	DC
Temporäre Variablen	D8
Rückgabewert	D4
Parameter	D0
Gespeicherter PC	CC
Gespeicherter SP	C8
Gespeicherte Register	C4
Lokale Daten	C0
Temporäre Variablen	60

Warum geht das schief?

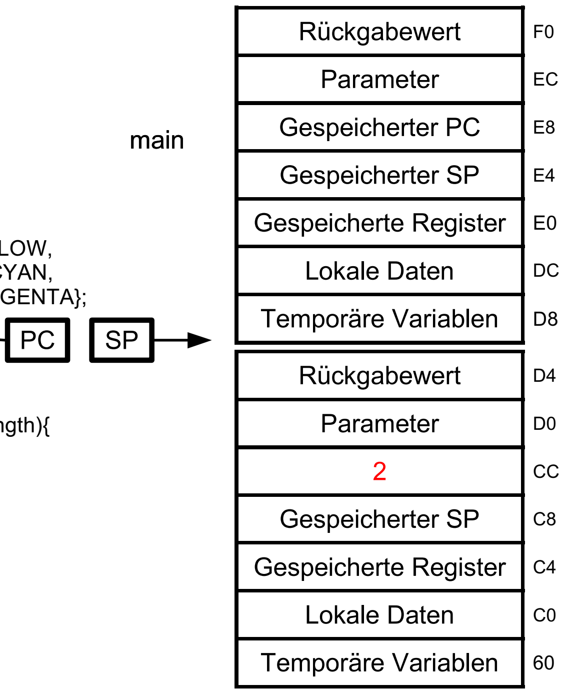
```
1   int main(){
2       int rainbow[]= makeRainbow();
3       printRainbow(rainbow, 6);
    }
4   int *makeRainbow(){
5       int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                        COLOR_GREEN, COLOR_CYAN,
                        COLOR_BLUE, COLOR_MAGENTA};
6       return rainbow;
    }
7   void printRainbow(int rainbow[], int rainbowLength){
8       for(int i= 0; i<rainbowLength; i++){
9           attron(COLOR_PAIR(rainbow[i]));
10          addstr("\u2588");
11          attroff(COLOR_PAIR(rainbow[i]));
        }
    }
```



Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
Lokale Daten	DC
Temporäre Variablen	D8
Rückgabewert	D4
Parameter	D0
Gespeicherter PC	CC
Gespeicherter SP	C8
Gespeicherte Register	C4
Lokale Daten	C0
Temporäre Variablen	60

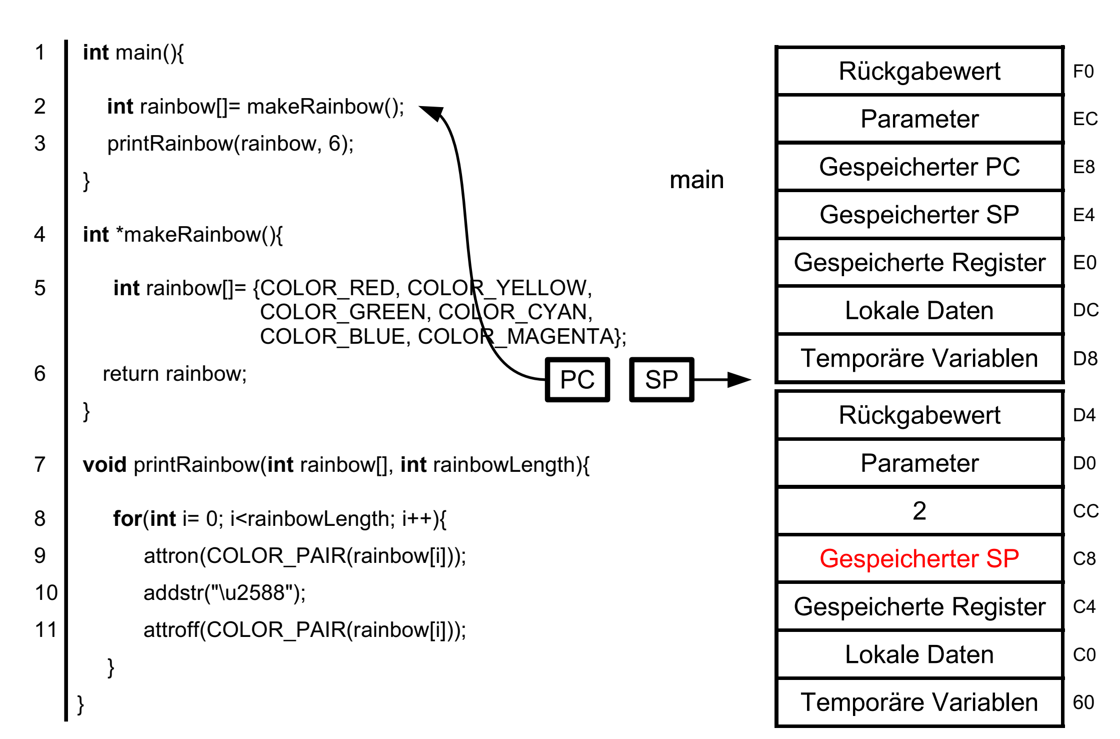
Warum geht das schief?

```
1   int main(){

2       int rainbow[]= makeRainbow();
3       printRainbow(rainbow, 6);
    }

4   int *makeRainbow(){

5       int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                        COLOR_GREEN, COLOR_CYAN,
                        COLOR_BLUE, COLOR_MAGENTA};

6      return rainbow;
    }

7   void printRainbow(int rainbow[], int rainbowLength){

8       for(int i= 0; i<rainbowLength; i++){
9           attron(COLOR_PAIR(rainbow[i]));
10          addstr("\u2588");
11          attroff(COLOR_PAIR(rainbow[i]));
        }
   }
```



Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
Lokale Daten	DC
Temporäre Variablen	D8
Rückgabewert	D4
Parameter	D0
2	CC
Gespeicherter SP	C8
Gespeicherte Register	C4
Lokale Daten	C0
Temporäre Variablen	60

Warum geht das schief?

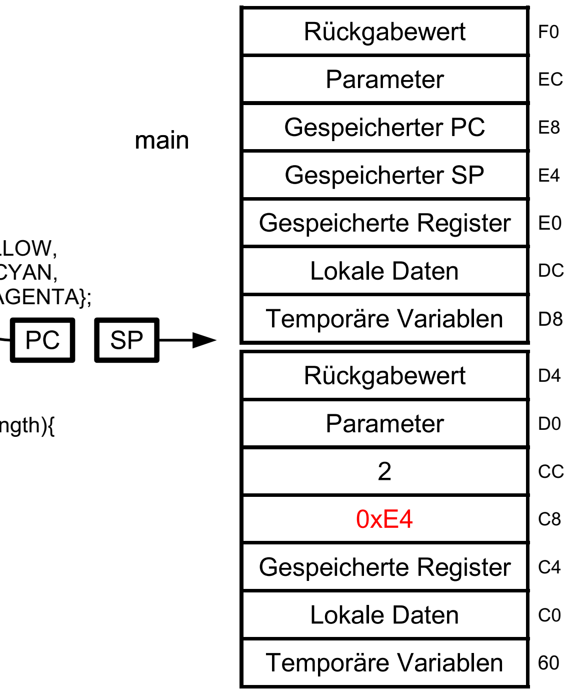


```
1   int main(){
2       int rainbow[]= makeRainbow();
3       printRainbow(rainbow, 6);
    }
4   int *makeRainbow(){
5        int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                         COLOR_GREEN, COLOR_CYAN,
                         COLOR_BLUE, COLOR_MAGENTA};
6       return rainbow;
    }
7   void printRainbow(int rainbow[], int rainbowLength){
8        for(int i= 0; i<rainbowLength; i++){
9            attron(COLOR_PAIR(rainbow[i]));
10           addstr("\u2588");
11           attroff(COLOR_PAIR(rainbow[i]));
        }
    }
```

Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
Lokale Daten	DC
Temporäre Variablen	D8
Rückgabewert	D4
Parameter	D0
2	CC
Gespeicherter SP	C8
Gespeicherte Register	C4
Lokale Daten	C0
Temporäre Variablen	60

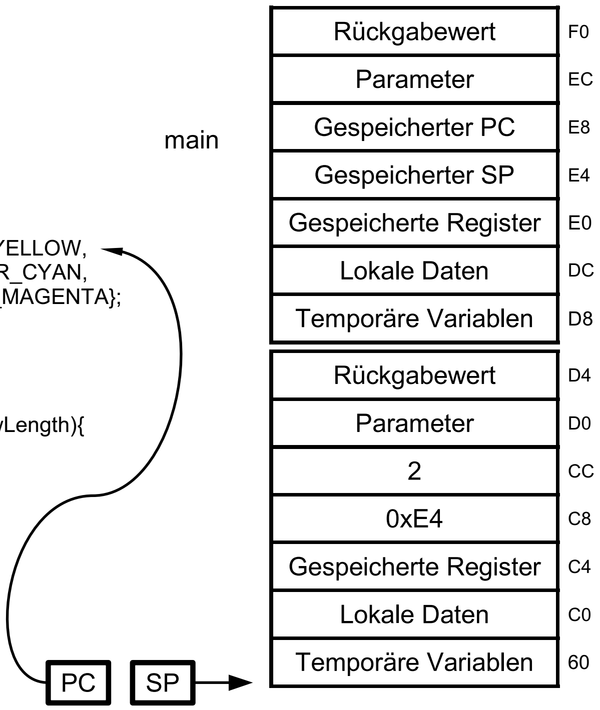
Warum geht das schief?

```
1   int main(){

2       int rainbow[]= makeRainbow();
3       printRainbow(rainbow, 6);
    }

4   int *makeRainbow(){

5        int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                         COLOR_GREEN, COLOR_CYAN,
                         COLOR_BLUE, COLOR_MAGENTA};

6       return rainbow;
    }

7   void printRainbow(int rainbow[], int rainbowLength){

8        for(int i= 0; i<rainbowLength; i++){
9            attron(COLOR_PAIR(rainbow[i]));
10           addstr("\u2588");
11           attroff(COLOR_PAIR(rainbow[i]));
         }
    }
```



Warum geht das schief?

```
1   int main(){

2      int rainbow[]= makeRainbow();
3      printRainbow(rainbow, 6);
    }

4   int *makeRainbow(){

5       int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                        COLOR_GREEN, COLOR_CYAN,
                        COLOR_BLUE, COLOR_MAGENTA};
6      return rainbow;
    }

7   void printRainbow(int rainbow[], int rainbowLength){

8       for(int i= 0; i<rainbowLength; i++){
9           attron(COLOR_PAIR(rainbow[i]));
10          addstr("\u2588");
11          attroff(COLOR_PAIR(rainbow[i]));
        }
    }
```

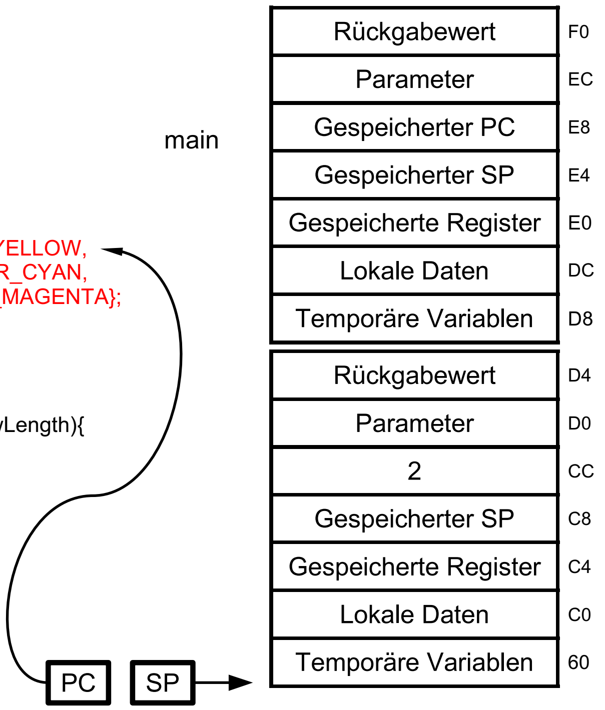


main

Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
Lokale Daten	DC
Temporäre Variablen	D8
Rückgabewert	D4
Parameter	D0
2	CC
0xE4	C8
Gespeicherte Register	C4
Lokale Daten	C0
Temporäre Variablen	60

Warum geht das schief?

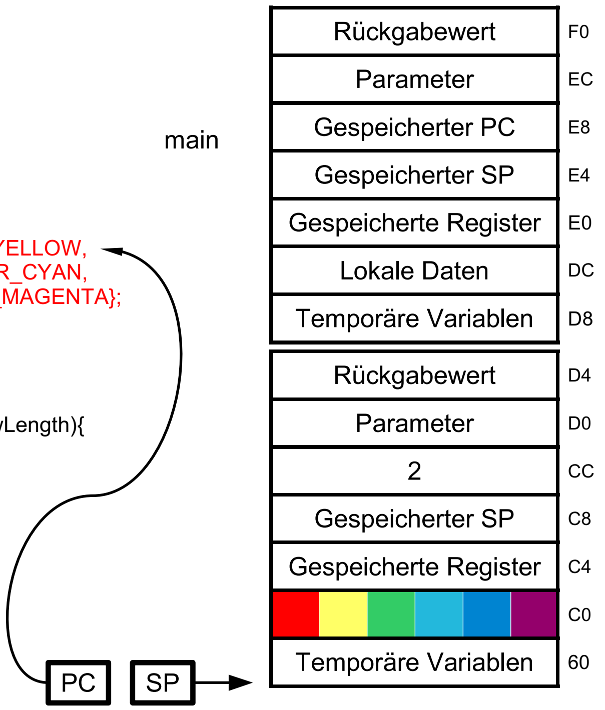
```
1   int main(){

2       int rainbow[]= makeRainbow();
3       printRainbow(rainbow, 6);
    }

4   int *makeRainbow(){

5        int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                         COLOR_GREEN, COLOR_CYAN,
                         COLOR_BLUE, COLOR_MAGENTA};

6       return rainbow;
    }

7   void printRainbow(int rainbow[], int rainbowLength){

8        for(int i= 0; i<rainbowLength; i++){
9            attron(COLOR_PAIR(rainbow[i]));
10           addstr("\u2588");
11           attroff(COLOR_PAIR(rainbow[i]));
         }
    }
```



Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
Lokale Daten	DC
Temporäre Variablen	D8
Rückgabewert	D4
Parameter	D0
2	CC
Gespeicherter SP	C8
Gespeicherte Register	C4
Lokale Daten	C0
Temporäre Variablen	60

Warum geht das schief?

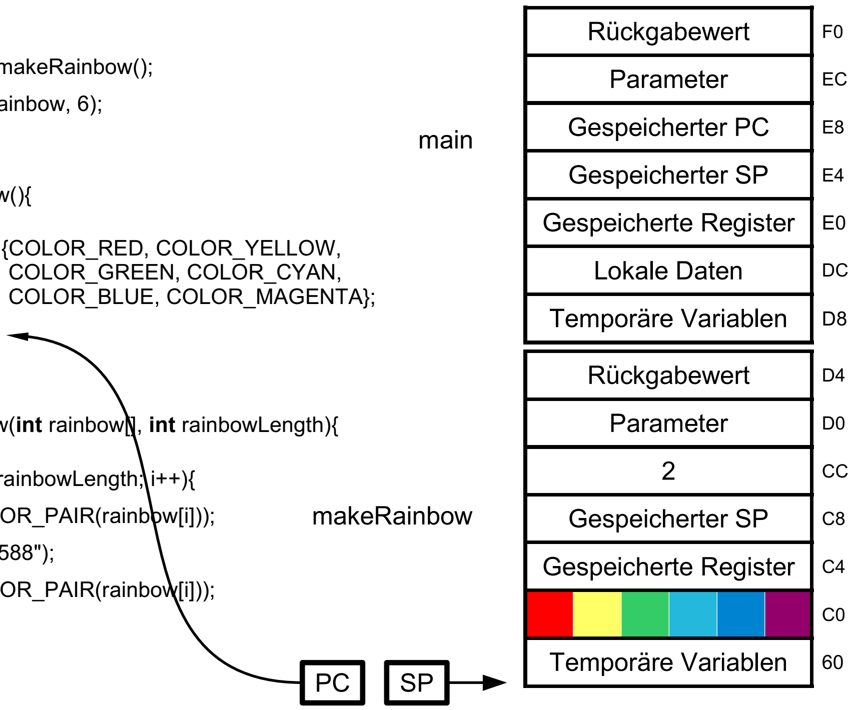
```
1   int main(){

2       int rainbow[]= makeRainbow();
3       printRainbow(rainbow, 6);
    }

4   int *makeRainbow(){

5        int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                         COLOR_GREEN, COLOR_CYAN,
                         COLOR_BLUE, COLOR_MAGENTA};

6       return rainbow;
    }

7   void printRainbow(int rainbow[], int rainbowLength){

8        for(int i= 0; i<rainbowLength; i++){
9            attron(COLOR_PAIR(rainbow[i]));
10           addstr("\u2588");
11           attroff(COLOR_PAIR(rainbow[i]));
        }
    }
```



Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
Lokale Daten	DC
Temporäre Variablen	D8
Rückgabewert	D4
Parameter	D0
2	CC
Gespeicherter SP	C8
Gespeicherte Register	C4
🟥🟨🟩🟦🟦🟪	C0
Temporäre Variablen	60

Warum geht das schief?

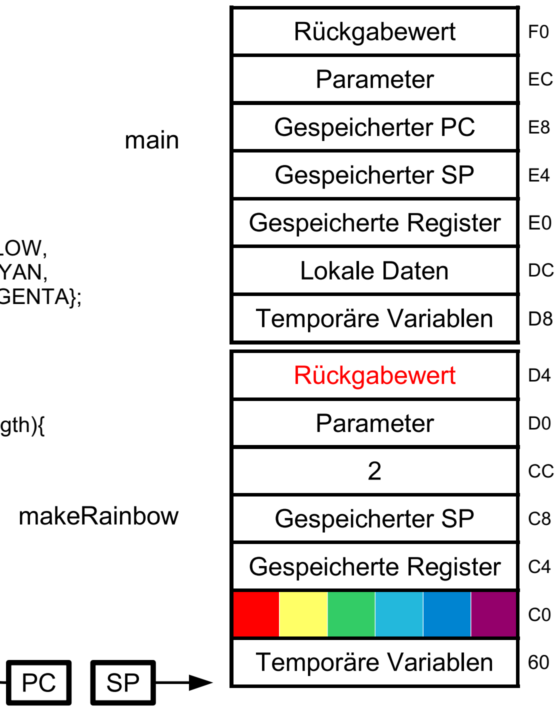
```
1   int main(){

2       int rainbow[]= makeRainbow();
3       printRainbow(rainbow, 6);
    }

4   int *makeRainbow(){

5       int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                        COLOR_GREEN, COLOR_CYAN,
                        COLOR_BLUE, COLOR_MAGENTA};

6       return rainbow;
    }

7   void printRainbow(int rainbow[], int rainbowLength){

8       for(int i= 0; i<rainbowLength; i++){
9           attron(COLOR_PAIR(rainbow[i]));
10          addstr("\u2588");
11          attroff(COLOR_PAIR(rainbow[i]));
        }
    }
```



Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
Lokale Daten	DC
Temporäre Variablen	D8
Rückgabewert	D4
Parameter	D0
2	CC
Gespeicherter SP	C8
Gespeicherte Register	C4
🟥🟨🟩🟦🟦🟪	C0
Temporäre Variablen	60

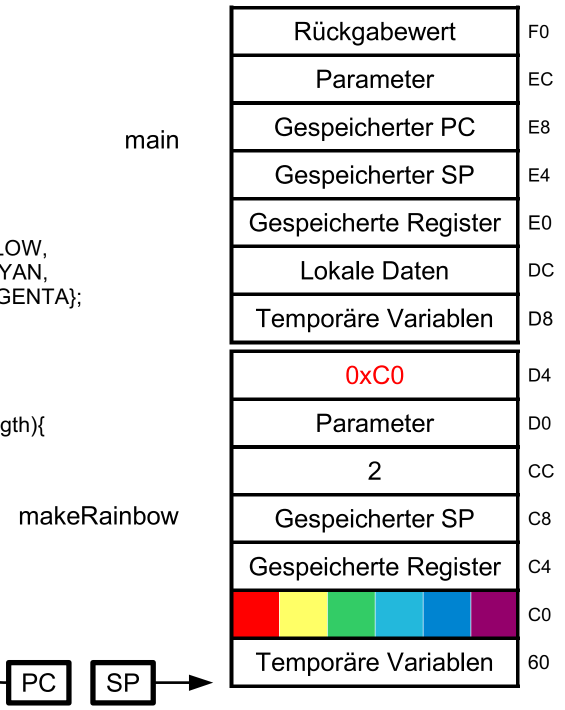
Warum geht das schief?

```
1   int main(){

2       int rainbow[]= makeRainbow();
3       printRainbow(rainbow, 6);
    }

4   int *makeRainbow(){

5        int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                         COLOR_GREEN, COLOR_CYAN,
                         COLOR_BLUE, COLOR_MAGENTA};

6      return rainbow;
    }

7   void printRainbow(int rainbow[], int rainbowLength){

8        for(int i= 0; i<rainbowLength; i++){
9            attron(COLOR_PAIR(rainbow[i]));
10           addstr("\u2588");
11           attroff(COLOR_PAIR(rainbow[i]));
        }
   }
```



Warum geht das schief?

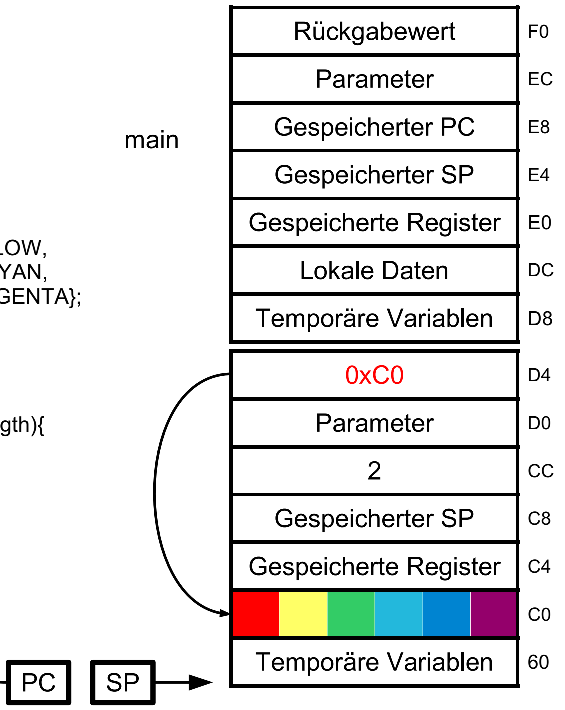
```
int main(){

    int rainbow[]= makeRainbow();
    printRainbow(rainbow, 6);
}

int *makeRainbow(){

     int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                     COLOR_GREEN, COLOR_CYAN,
                     COLOR_BLUE, COLOR_MAGENTA};

   return rainbow;
}

void printRainbow(int rainbow[], int rainbowLength){

     for(int i= 0; i<rainbowLength; i++){
         attron(COLOR_PAIR(rainbow[i]));
         addstr("\u2588");
         attroff(COLOR_PAIR(rainbow[i]));
    }
}
```



main	Rückgabewert	F0
	Parameter	EC
	Gespeicherter PC	E8
	Gespeicherter SP	E4
	Gespeicherte Register	E0
	Lokale Daten	DC
	Temporäre Variablen	D8
makeRainbow	0xC0	D4
	Parameter	D0
	2	CC
	Gespeicherter SP	C8
	Gespeicherte Register	C4
	🟥🟨🟩🟦🟦🟪	C0
	Temporäre Variablen	60

Warum geht das schief?

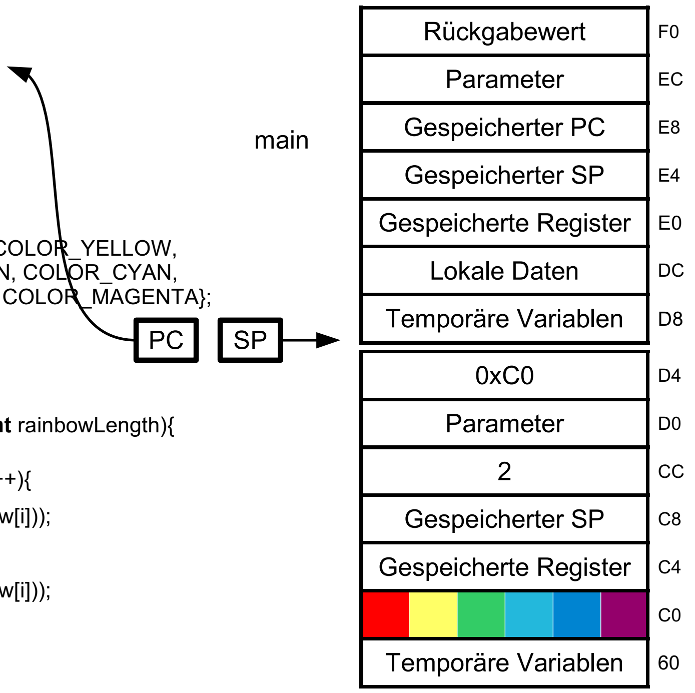
```
1   int main(){

2       int rainbow[]= makeRainbow();
3       printRainbow(rainbow, 6);
    }

4   int *makeRainbow(){

5        int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                         COLOR_GREEN, COLOR_CYAN,
                         COLOR_BLUE, COLOR_MAGENTA};

6       return rainbow;
    }

7   void printRainbow(int rainbow[], int rainbowLength){

8        for(int i= 0; i<rainbowLength; i++){
9            attron(COLOR_PAIR(rainbow[i]));
10           addstr("\u2588");
11           attroff(COLOR_PAIR(rainbow[i]));
         }
    }
```



Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
Lokale Daten	DC
Temporäre Variablen	D8
0xC0	D4
Parameter	D0
2	CC
Gespeicherter SP	C8
Gespeicherte Register	C4
🟥🟨🟩🟦🟦🟪	C0
Temporäre Variablen	60

Warum geht das schief?

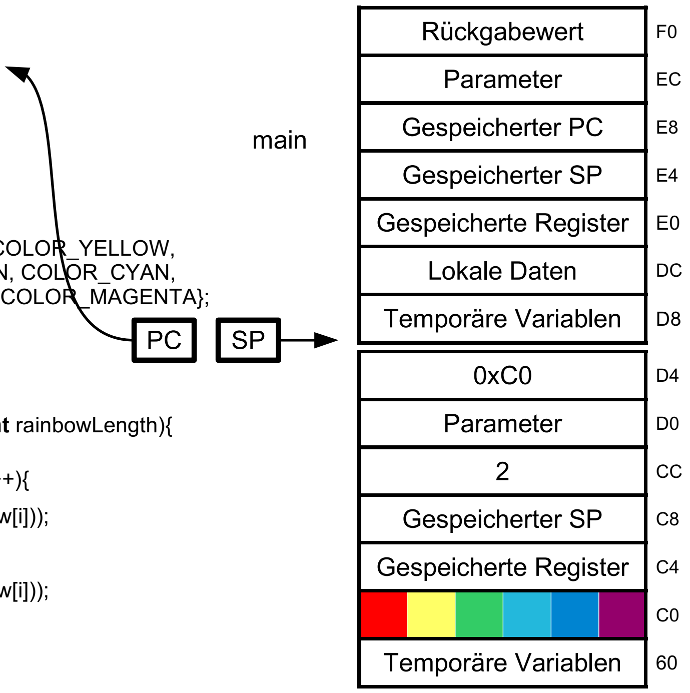
```
1   int main(){

2       int rainbow[]= makeRainbow();
3       printRainbow(rainbow, 6);
    }

4   int *makeRainbow(){

5        int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                         COLOR_GREEN, COLOR_CYAN,
                         COLOR_BLUE, COLOR_MAGENTA};

6      return rainbow;
    }

7   void printRainbow(int rainbow[], int rainbowLength){

8        for(int i= 0; i<rainbowLength; i++){
9            attron(COLOR_PAIR(rainbow[i]));
10           addstr("\u2588");
11           attroff(COLOR_PAIR(rainbow[i]));
        }
   }
```



Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
Lokale Daten	DC
Temporäre Variablen	D8
0xC0	D4
Parameter	D0
2	CC
Gespeicherter SP	C8
Gespeicherte Register	C4
🟥🟨🟩🟦🟦🟪	C0
Temporäre Variablen	60

Warum geht das schief?

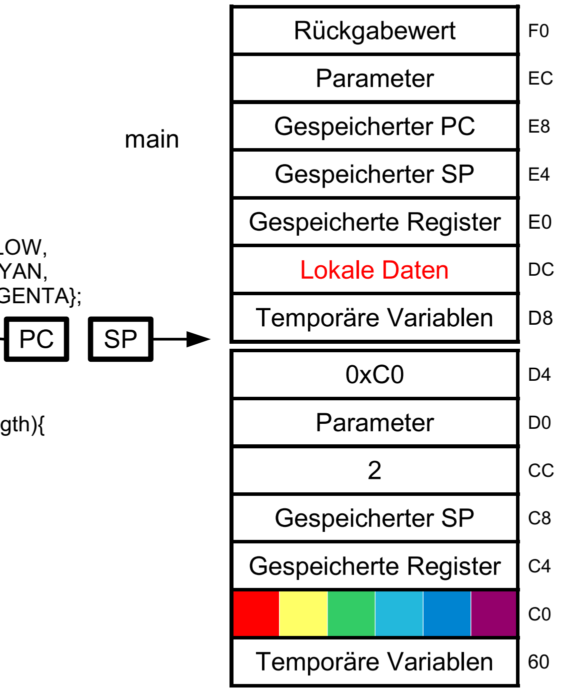
```
int main(){

    int rainbow[]= makeRainbow();
    printRainbow(rainbow, 6);
}

int *makeRainbow(){

     int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                         COLOR_GREEN, COLOR_CYAN,
                         COLOR_BLUE, COLOR_MAGENTA};

    return rainbow;
}

void printRainbow(int rainbow[], int rainbowLength){

     for(int i= 0; i<rainbowLength; i++){
          attron(COLOR_PAIR(rainbow[i]));
          addstr("\u2588");
          attroff(COLOR_PAIR(rainbow[i]));
    }
}
```



Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
Lokale Daten	DC
Temporäre Variablen	D8
0xC0	D4
Parameter	D0
2	CC
Gespeicherter SP	C8
Gespeicherte Register	C4
🟥🟨🟩🟦🟦🟪	C0
Temporäre Variablen	60

Warum geht das schief?

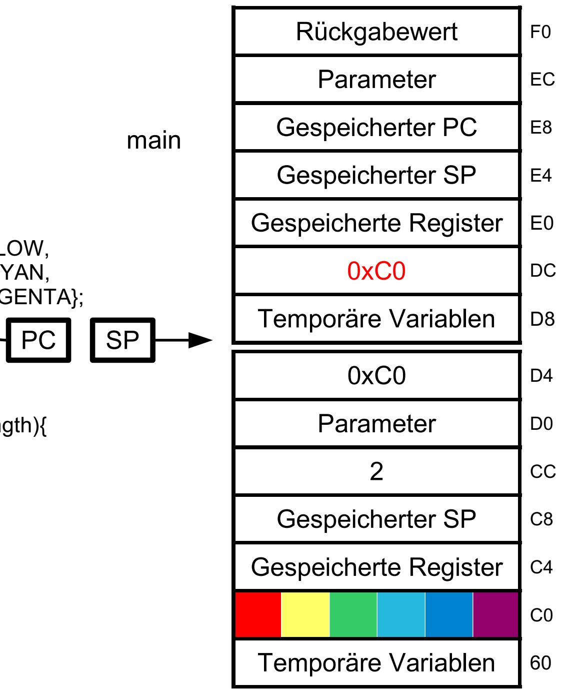
```
int main(){

    int rainbow[]= makeRainbow();
    printRainbow(rainbow, 6);
}

int *makeRainbow(){

     int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                     COLOR_GREEN, COLOR_CYAN,
                     COLOR_BLUE, COLOR_MAGENTA};

    return rainbow;
}

void printRainbow(int rainbow[], int rainbowLength){

     for(int i= 0; i<rainbowLength; i++){
          attron(COLOR_PAIR(rainbow[i]));
          addstr("\u2588");
          attroff(COLOR_PAIR(rainbow[i]));
    }
}
```



Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
Lokale Daten	DC
Temporäre Variablen	D8
0xC0	D4
Parameter	D0
2	CC
Gespeicherter SP	C8
Gespeicherte Register	C4
🟥🟨🟩🟦🟦🟪	C0
Temporäre Variablen	60

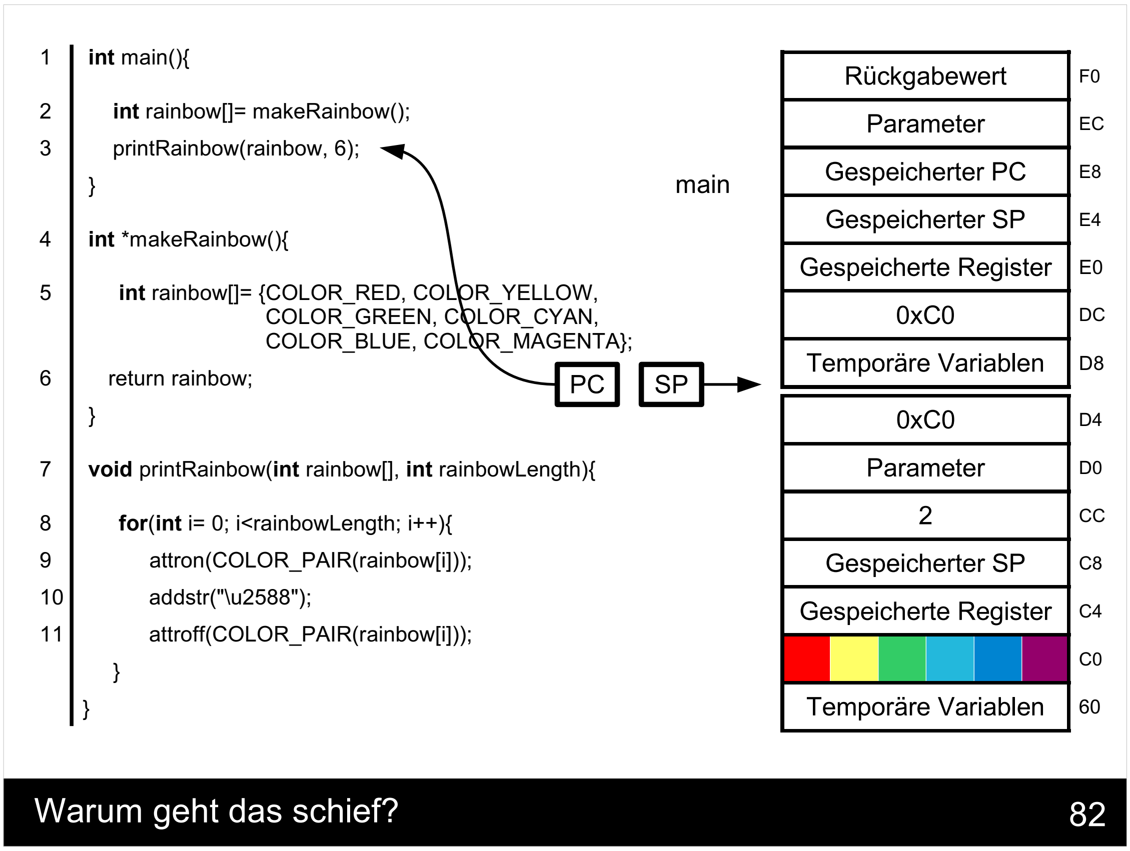
Warum geht das schief?

```
int main(){

    int rainbow[]= makeRainbow();
    printRainbow(rainbow, 6);
}

int *makeRainbow(){

    int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                    COLOR_GREEN, COLOR_CYAN,
                    COLOR_BLUE, COLOR_MAGENTA};

    return rainbow;
}

void printRainbow(int rainbow[], int rainbowLength){

    for(int i= 0; i<rainbowLength; i++){
        attron(COLOR_PAIR(rainbow[i]));
        addstr("\u2588");
        attroff(COLOR_PAIR(rainbow[i]));
    }
}
```



Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
0xC0	DC
Temporäre Variablen	D8
0xC0	D4
Parameter	D0
2	CC
Gespeicherter SP	C8
Gespeicherte Register	C4
🟥🟨🟩🟦🟦🟪	C0
Temporäre Variablen	60

Warum geht das schief?

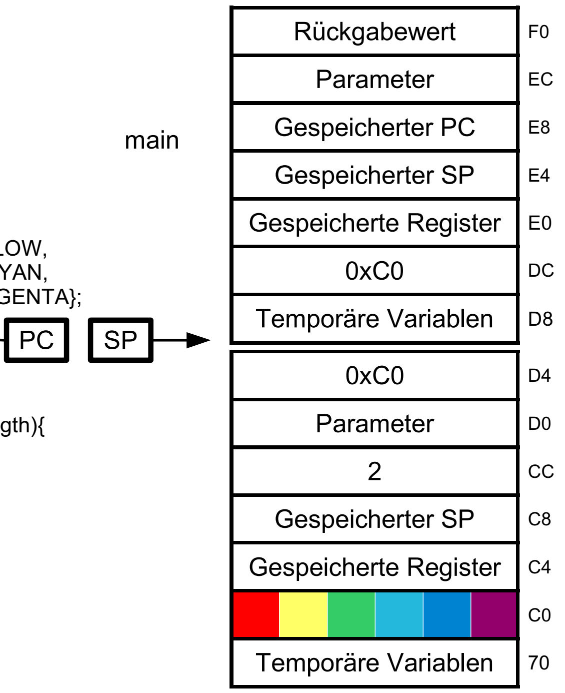


```
int main(){

    int rainbow[]= makeRainbow();
    printRainbow(rainbow, 6);
}

int *makeRainbow(){

    int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                    COLOR_GREEN, COLOR_CYAN,
                    COLOR_BLUE, COLOR_MAGENTA};

    return rainbow;
}

void printRainbow(int rainbow[], int rainbowLength){

    for(int i= 0; i<rainbowLength; i++){
        attron(COLOR_PAIR(rainbow[i]));
        addstr("\u2588");
        attroff(COLOR_PAIR(rainbow[i]));
    }
}
```

Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
0xC0	DC
Temporäre Variablen	D8
0xC0	D4
Parameter	D0
2	CC
Gespeicherter SP	C8
Gespeicherte Register	C4
🟥🟨🟩🟦🟦🟪	C0
Temporäre Variablen	60

Warum geht das schief?

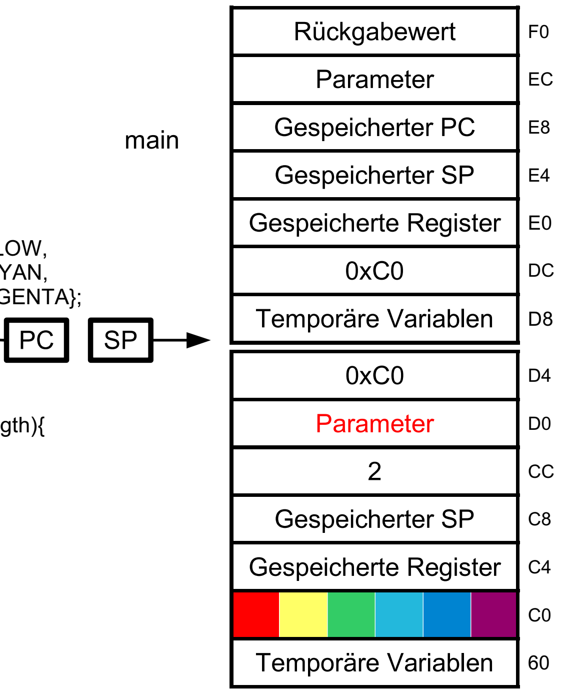
```
1  int main(){

2      int rainbow[]= makeRainbow();
3      printRainbow(rainbow, 6);
   }

4  int *makeRainbow(){

5       int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                        COLOR_GREEN, COLOR_CYAN,
                        COLOR_BLUE, COLOR_MAGENTA};

6     return rainbow;
   }

7  void printRainbow(int rainbow[], int rainbowLength){

8       for(int i= 0; i<rainbowLength; i++){
9           attron(COLOR_PAIR(rainbow[i]));
10          addstr("\u2588");
11          attroff(COLOR_PAIR(rainbow[i]));
       }
  }
```



Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
0xC0	DC
Temporäre Variablen	D8
0xC0	D4
Parameter	D0
2	CC
Gespeicherter SP	C8
Gespeicherte Register	C4
🟥🟨🟩🟦🟦🟪	C0
Temporäre Variablen	70

Warum geht das schief?

```
int main(){

    int rainbow[]= makeRainbow();
    printRainbow(rainbow, 6);
}

int *makeRainbow(){

     int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                     COLOR_GREEN, COLOR_CYAN,
                     COLOR_BLUE, COLOR_MAGENTA};

    return rainbow;
}

void printRainbow(int rainbow[], int rainbowLength){

     for(int i= 0; i<rainbowLength; i++){
          attron(COLOR_PAIR(rainbow[i]));
          addstr("\u2588");
          attroff(COLOR_PAIR(rainbow[i]));
     }
}
```



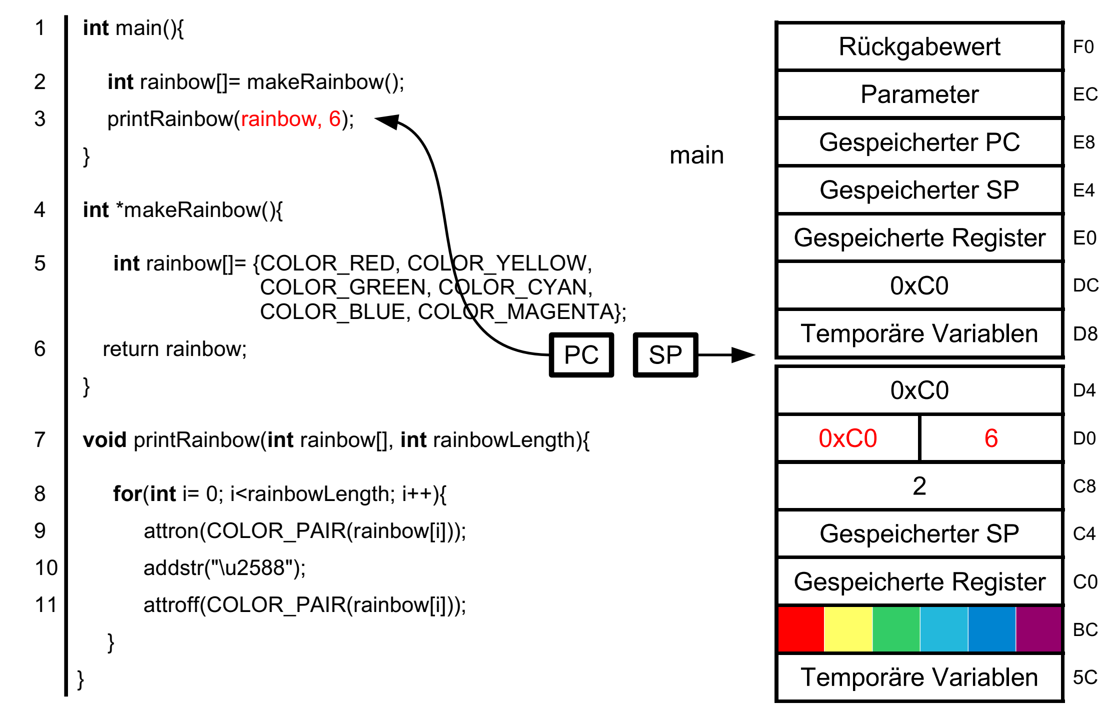
main

PC

SP

Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
0xC0	DC
Temporäre Variablen	D8
0xC0	D4
Parameter	D0
2	CC
Gespeicherter SP	C8
Gespeicherte Register	C4
🟥🟨🟩🟦🟦🟪	C0
Temporäre Variablen	60

Warum geht das schief?

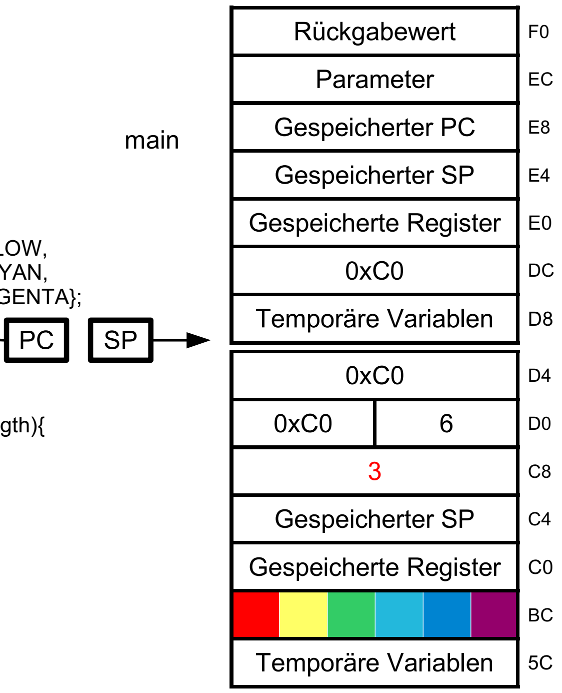


```
int main(){
    int rainbow[]= makeRainbow();
    printRainbow(rainbow, 6);
}
int *makeRainbow(){
     int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                         COLOR_GREEN, COLOR_CYAN,
                         COLOR_BLUE, COLOR_MAGENTA};
   return rainbow;
}
void printRainbow(int rainbow[], int rainbowLength){
     for(int i= 0; i<rainbowLength; i++){
          attron(COLOR_PAIR(rainbow[i]));
          addstr("\u2588");
          attroff(COLOR_PAIR(rainbow[i]));
    }
}
```

Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
0xC0	DC
Temporäre Variablen	D8
0xC0	D4
0xC0 | 6	D0
2	C8
Gespeicherter SP	C4
Gespeicherte Register	C0
🟥🟨🟩🟦🟦🟪	BC
Temporäre Variablen	5C

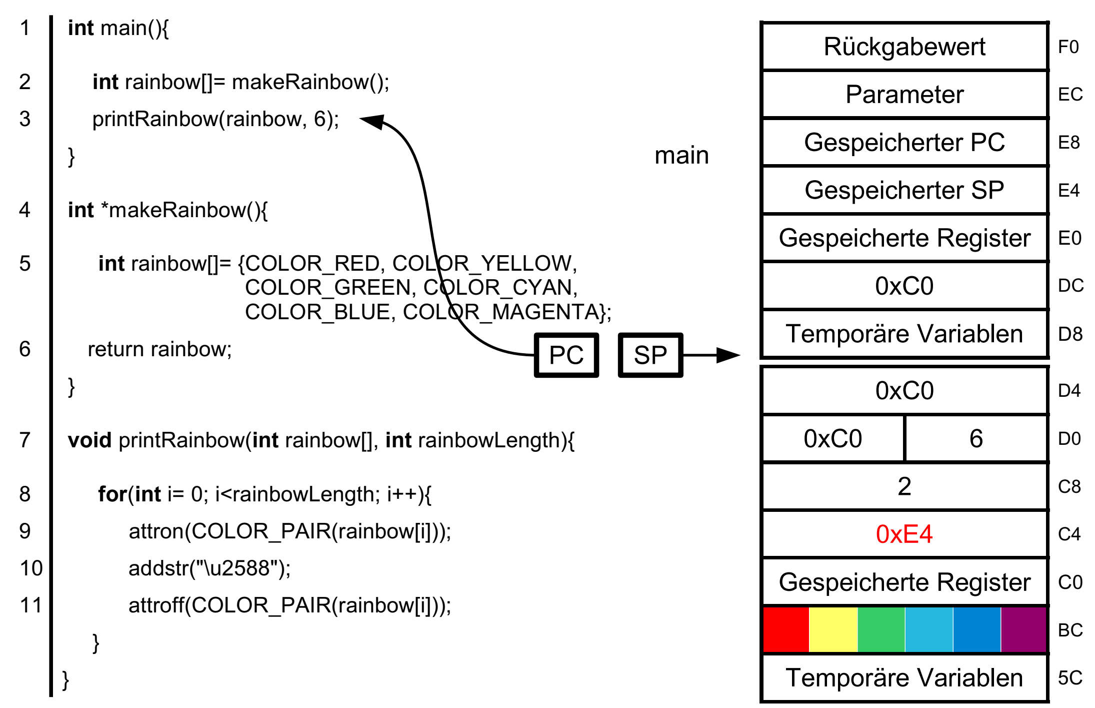
Warum geht das schief?

```
1   int main(){

2       int rainbow[]= makeRainbow();
3       printRainbow(rainbow, 6);
    }

4   int *makeRainbow(){

5       int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                       COLOR_GREEN, COLOR_CYAN,
                       COLOR_BLUE, COLOR_MAGENTA};

6       return rainbow;
    }

7   void printRainbow(int rainbow[], int rainbowLength){

8       for(int i= 0; i<rainbowLength; i++){
9           attron(COLOR_PAIR(rainbow[i]));
10          addstr("\u2588");
11          attroff(COLOR_PAIR(rainbow[i]));
        }
    }
```



Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
0xC0	DC
Temporäre Variablen	D8
0xC0	D4
0xC0 | 6	D0
3	C8
Gespeicherter SP	C4
Gespeicherte Register	C0
🟥🟨🟩🟦🟦🟪	BC
Temporäre Variablen	5C

Warum geht das schief?

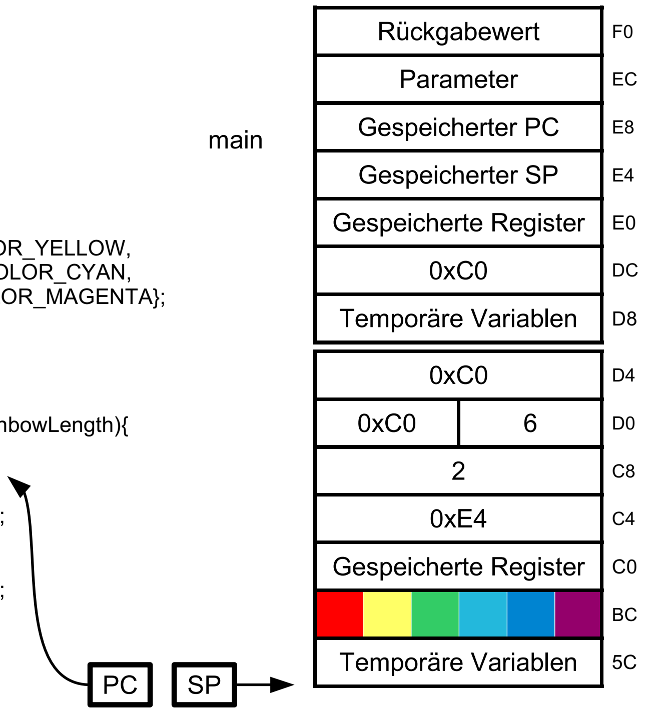


```
int main(){

    int rainbow[]= makeRainbow();
    printRainbow(rainbow, 6);
}

int *makeRainbow(){

     int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                     COLOR_GREEN, COLOR_CYAN,
                     COLOR_BLUE, COLOR_MAGENTA};

   return rainbow;
}

void printRainbow(int rainbow[], int rainbowLength){

     for(int i= 0; i<rainbowLength; i++){
         attron(COLOR_PAIR(rainbow[i]));
         addstr("\u2588");
         attroff(COLOR_PAIR(rainbow[i]));
     }
}
```

Rückgabewert		F0
Parameter		EC
Gespeicherter PC		E8
Gespeicherter SP		E4
Gespeicherte Register		E0
0xC0		DC
Temporäre Variablen		D8
0xC0		D4
0xC0	6	D0
2		C8
0xE4		C4
Gespeicherte Register		C0
🟥🟨🟩🟦🟦🟪		BC
Temporäre Variablen		5C

Warum geht das schief?

```
1   int main(){

2       int rainbow[]= makeRainbow();
3       printRainbow(rainbow, 6);
    }

4   int *makeRainbow(){

5        int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                         COLOR_GREEN, COLOR_CYAN,
                         COLOR_BLUE, COLOR_MAGENTA};

6       return rainbow;
    }

7   void printRainbow(int rainbow[], int rainbowLength){

8        for(int i= 0; i<rainbowLength; i++){
9            attron(COLOR_PAIR(rainbow[i]));
10           addstr("\u2588");
11           attroff(COLOR_PAIR(rainbow[i]));
         }
    }
```

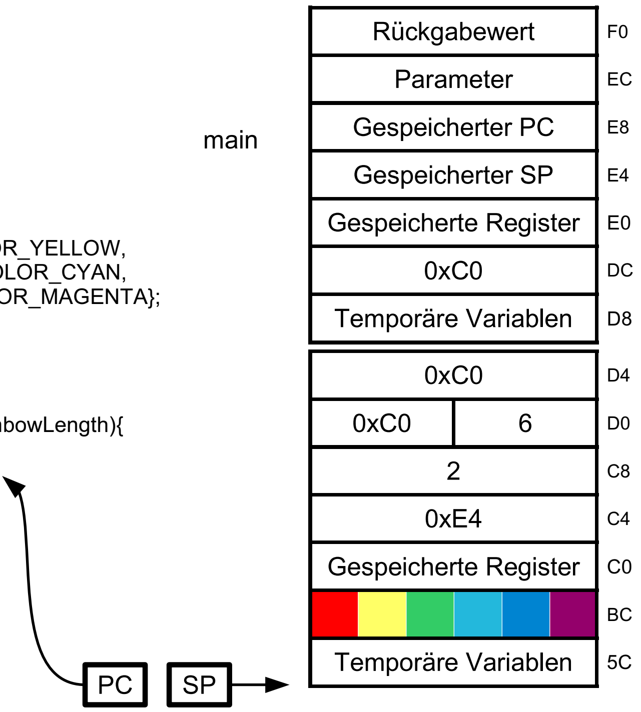


main

Rückgabewert		F0
Parameter		EC
Gespeicherter PC		E8
Gespeicherter SP		E4
Gespeicherte Register		E0
0xC0		DC
Temporäre Variablen		D8
0xC0		D4
0xC0	6	D0
2		C8
0xE4		C4
Gespeicherte Register		C0
🟥 🟨 🟩 🟦 🟦 🟪		BC
Temporäre Variablen		5C

Warum geht das schief?

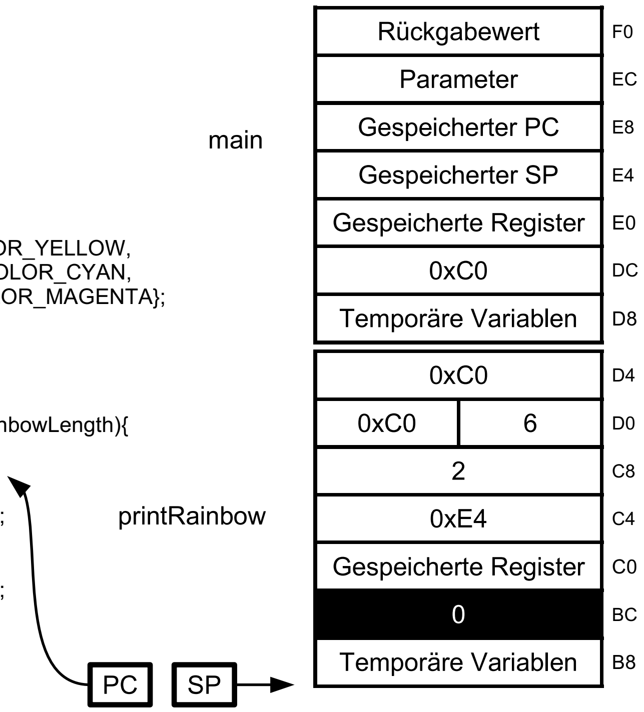
```
1   int main(){

2      int rainbow[]= makeRainbow();
3      printRainbow(rainbow, 6);
    }

4   int *makeRainbow(){

5       int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                        COLOR_GREEN, COLOR_CYAN,
                        COLOR_BLUE, COLOR_MAGENTA};

6      return rainbow;
    }

7   void printRainbow(int rainbow[], int rainbowLength){

8       for(int i= 0; i<rainbowLength; i++){
9           attron(COLOR_PAIR(rainbow[i]));
10          addstr("\u2588");
11          attroff(COLOR_PAIR(rainbow[i]));
        }
    }
```



Rückgabewert		F0
Parameter		EC
Gespeicherter PC		E8
Gespeicherter SP		E4
Gespeicherte Register		E0
0xC0		DC
Temporäre Variablen		D8
0xC0		D4
0xC0	6	D0
2		C8
0xE4		C4
Gespeicherte Register		C0
🟥🟨🟩🟦🟦🟪		BC
Temporäre Variablen		5C

Warum geht das schief?

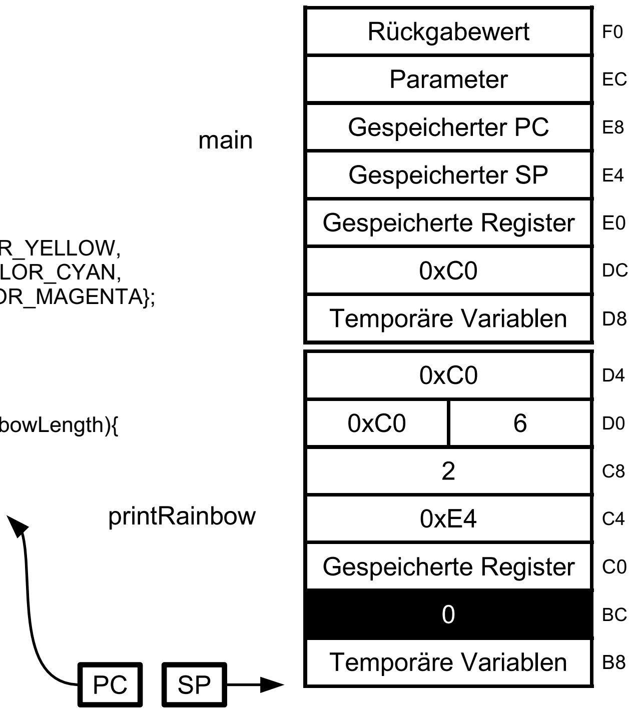
```
1   int main(){

2       int rainbow[]= makeRainbow();
3       printRainbow(rainbow, 6);
    }

4   int *makeRainbow(){

5       int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                        COLOR_GREEN, COLOR_CYAN,
                        COLOR_BLUE, COLOR_MAGENTA};

6       return rainbow;
    }

7   void printRainbow(int rainbow[], int rainbowLength){

8       for(int i= 0; i<rainbowLength; i++){
9           attron(COLOR_PAIR(rainbow[i]));
10          addstr("\u2588");
11          attroff(COLOR_PAIR(rainbow[i]));
        }
    }
```



Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
0xC0	DC
Temporäre Variablen	D8
0xC0	D4
0xC0 | 6	D0
2	C8
0xE4	C4
Gespeicherte Register	C0
0	BC
Temporäre Variablen	B8

Warum geht das schief?

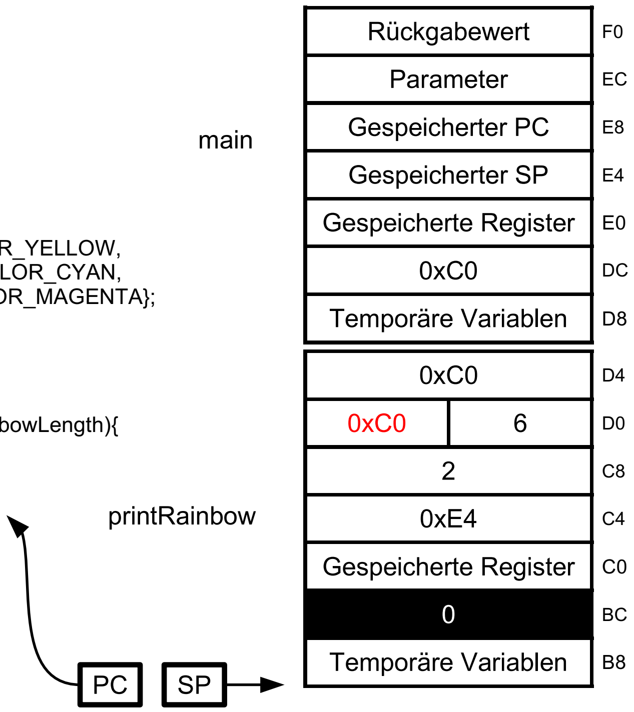
```
int main(){

    int rainbow[]= makeRainbow();
    printRainbow(rainbow, 6);
}

int *makeRainbow(){

     int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                     COLOR_GREEN, COLOR_CYAN,
                     COLOR_BLUE, COLOR_MAGENTA};

    return rainbow;
}

void printRainbow(int rainbow[], int rainbowLength){

     for(int i= 0; i<rainbowLength; i++){
          attron(COLOR_PAIR(rainbow[i]));
          addstr("\u2588");
          attroff(COLOR_PAIR(rainbow[i]));
     }
}
```



Rückgabewert		F0
Parameter		EC
Gespeicherter PC		E8
Gespeicherter SP		E4
Gespeicherte Register		E0
0xC0		DC
Temporäre Variablen		D8
0xC0		D4
0xC0	6	D0
2		C8
0xE4		C4
Gespeicherte Register		C0
0		BC
Temporäre Variablen		B8

Warum geht das schief?

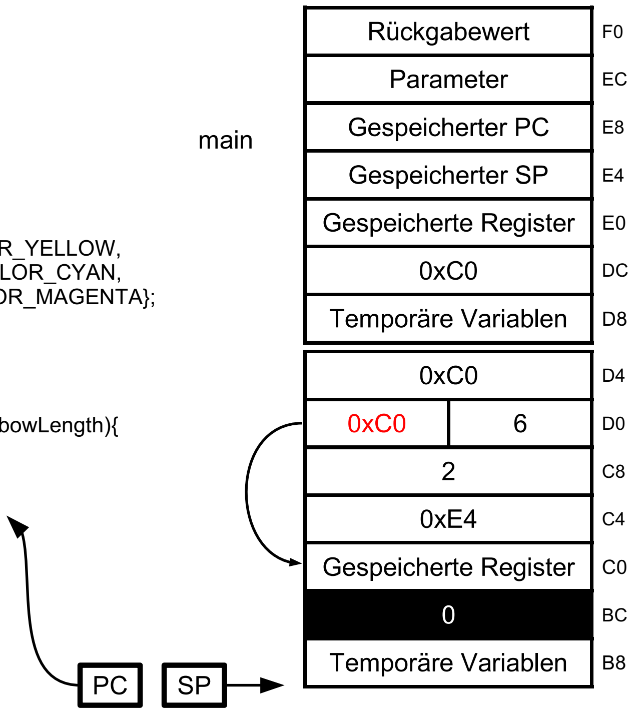
```
int main(){

    int rainbow[]= makeRainbow();
    printRainbow(rainbow, 6);
}

int *makeRainbow(){

     int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                     COLOR_GREEN, COLOR_CYAN,
                     COLOR_BLUE, COLOR_MAGENTA};

    return rainbow;
}

void printRainbow(int rainbow[], int rainbowLength){

     for(int i= 0; i<rainbowLength; i++){
         attron(COLOR_PAIR(rainbow[i]));
         addstr("\u2588");
         attroff(COLOR_PAIR(rainbow[i]));
    }
}
```



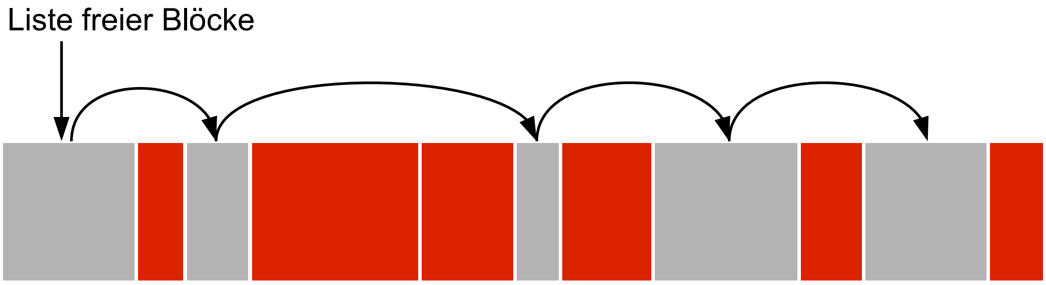
Rückgabewert	F0
Parameter	EC
Gespeicherter PC	E8
Gespeicherter SP	E4
Gespeicherte Register	E0
0xC0	DC
Temporäre Variablen	D8
0xC0	D4
0xC0 | 6	D0
2	C8
0xE4	C4
Gespeicherte Register	C0
0	BC
Temporäre Variablen	B8

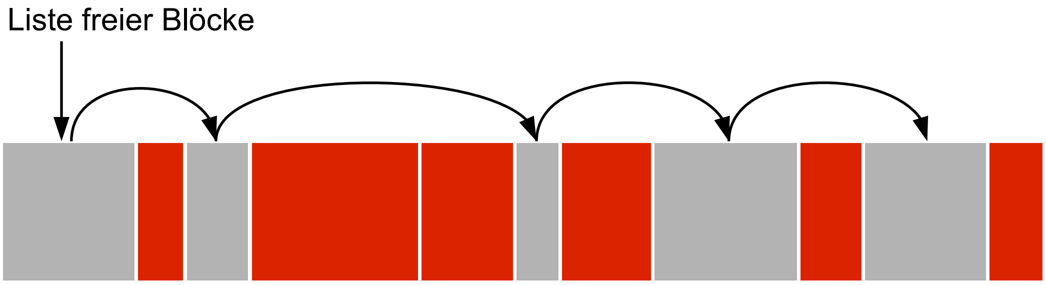
Warum geht das schief?

```
1   int main(){

2       int rainbow[]= makeRainbow();
3       printRainbow(rainbow, 6);
    }

4   int *makeRainbow(){

5       int rainbow[]= {COLOR_RED, COLOR_YELLOW,
                        COLOR_GREEN, COLOR_CYAN,
                        COLOR_BLUE, COLOR_MAGENTA};

6       return rainbow;
    }

7   void printRainbow(int rainbow[], int rainbowLength){

8       for(int i= 0; i<rainbowLength; i++){
9           attron(COLOR_PAIR(rainbow[i]));
10          addstr("\u2588");
11          attroff(COLOR_PAIR(rainbow[i]));
        }
    }
```



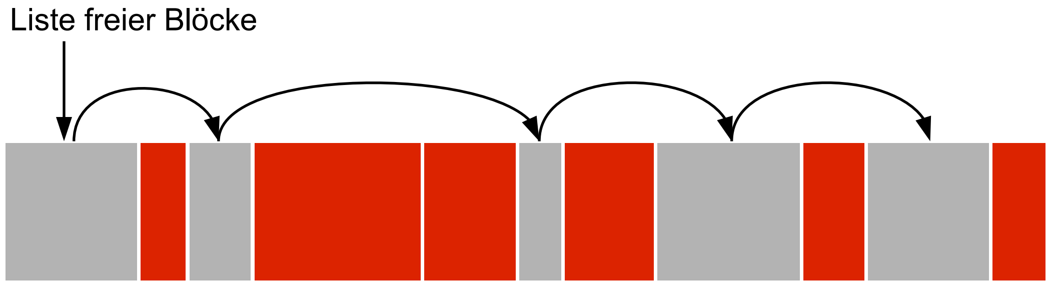
Warum geht das schief?


Liste freier Blöcke



Anfordern von Speicher:

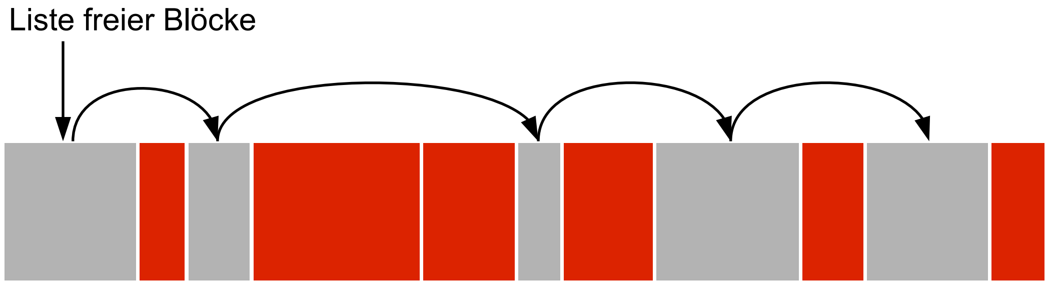
void *malloc(size_t size);
void *calloc(size_t nmemb, size_t size);



Anfordern von Speicher:

void *malloc(size_t size);
void *calloc(size_t nmemb, size_t size);

Entspricht dem **new** in Java



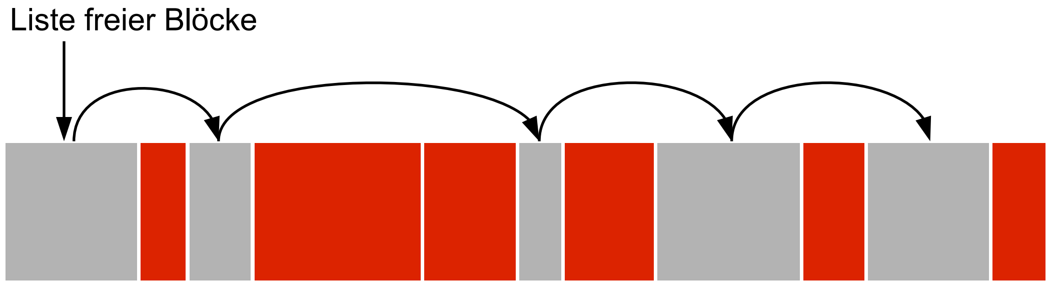
Anfordern von Speicher:

void *malloc(size_t size);
void *calloc(size_t nmemb, size_t size);

Entspricht dem **new** in Java

Freigeben von Speicher:

void free(**void** *ptr);



Anfordern von Speicher:

```
void *malloc(size_t size);
void *calloc(size_t nmemb, size_t size);
```

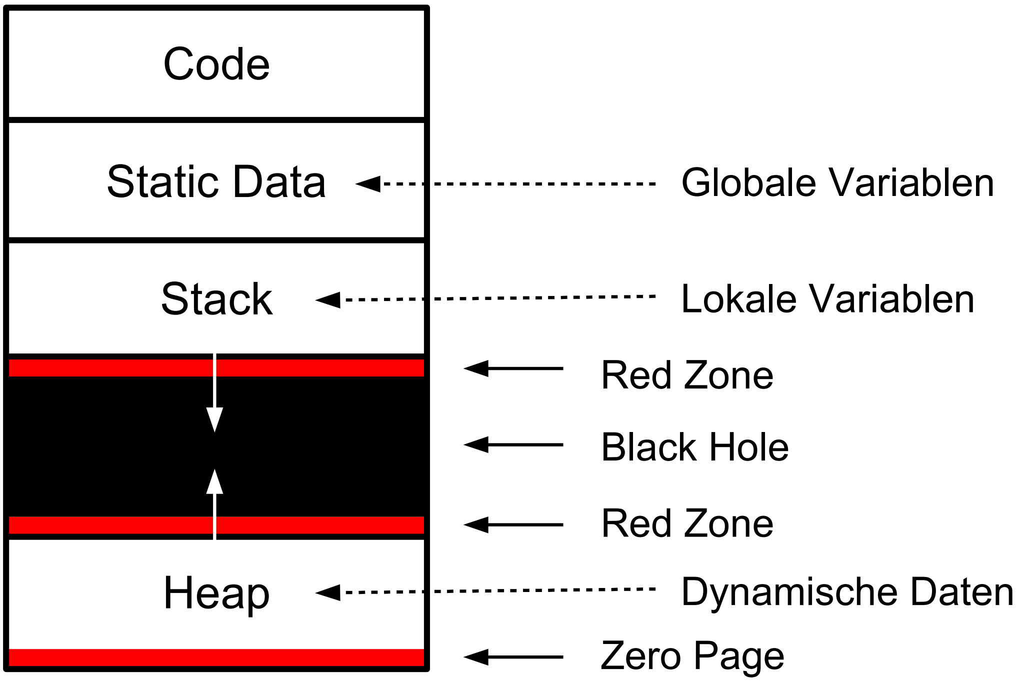
Entspricht dem **new** in Java

Freigeben von Speicher:

```
void free(void *ptr);
```

Anfordern und Freigeben kann in beliebiger Verzahnung geschehen und ist unabhängig von der Lebensdauer eines Stackframe

Der Heap



Code

Static Data

Stack

Heap

Globale Variablen

Lokale Variablen

Red Zone

Black Hole

Red Zone

Dynamische Daten

Zero Page

Datei field.h

```
#pragma once

#include "point.h"

typedef struct field Field;

Field * newField(int width, int height);
void freeField(Field *field);

int get(Field *field, Point point);

void increment(Field *field, Point point);
void decrement(Field *field, Point point);

Point size(Field *field);
```

Anwendung von calloc() und free()

Datei field.h

```
#pragma once

#include "point.h"

typedef struct field Field;

Field * newField(int width, int height);
void freeField(Field *field);

int get(Field *field, Point point);

void increment(Field *field, Point point);
void decrement(Field *field, Point point);

Point size(Field *field);
```

Anwendung von calloc() und free()

Datei field.h

```
#pragma once

#include "point.h"

typedef struct field Field;

Field * newField(int width, int height);
void freeField(Field *field);

int get(Field *field, Point point);

void increment(Field *field, Point point);
void decrement(Field *field, Point point);

Point size(Field *field);
```

Anwendung von calloc() und free()

Datei field.c

```
...

struct field{
    int **panel;
    int width;
    int height;
};

…
```

Anwendung von calloc() und free()

Datei field.c

```
…

Field * newField(int width, int height){

    Field *field= calloc(1, sizeof(Field));
    field->width= width;
    field->height= height;
    field->panel= (int **) calloc(height, sizeof(int*));
    for(int i=0; i<height; i++){
        field->panel[i]= (int*) calloc(width, sizeof(int));
    }
    return field;
}

…
```

Anwendung von calloc() und free()

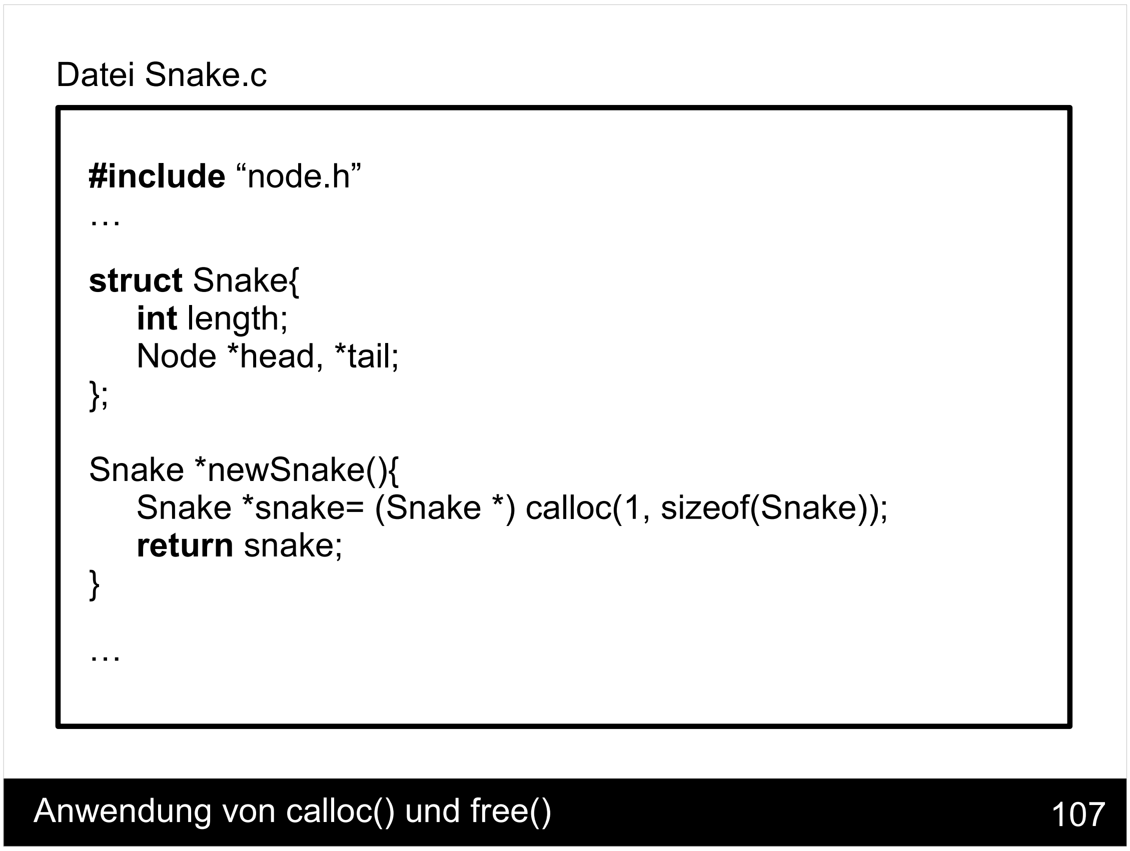
Datei field.c

```
...

 void freeField(Field *field){

 for (int i=0; i<field->height; i++){
    free(field->panel[i]);
 }
    free(field->panel);
    free(field);
    field= NULL;
 }

 …
```

Datei Snake.h

```
…

typedef struct snake Snake;

Snake *newSnake();
void freeSnake(Snake *snake);

int length(Snake *list);

void *getFirst(Snake *list);
void *getLast(Snake *list);

void pushFirst(Snake *list, void *content);
void popLast(Snake *snake);

…
```

Anwendung von calloc() und free()



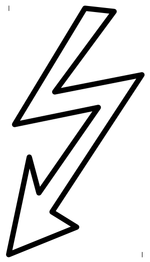
Datei Snake.c

```
#include “node.h”
…

struct Snake{
    int length;
    Node *head, *tail;
};

Snake *newSnake(){
    Snake *snake= (Snake *) calloc(1, sizeof(Snake));
    return snake;
}

…
```

Anwendung von calloc() und free()

Datei Snake.c

```
…

void freeSnake(Snake * snake){

    Node head= snake->head;
    for(; head!=NULL; head= next(head)){
        freeNode(head);
    }
    snake= NULL;
}

…
```



Datei Snake.c

```
…

void freeSnake(Snake * snake){

    Node head= snake->head;
    Node temp;
    for(; head!=NULL; head= temp){
        temp= next(head);
        freeNode(head);
    }
    snake= NULL;
}

…
```

Danke!

Danke!
Fragen?