Dynamische allocatie

Juist?

```
int *f() {
  int a = 10;
  return &a;
}
```

Juist?

```
int *f() {
  int array[] = {1, 2, 3};
  return array;
}
```

Eerder:

```
void f() {
   int a = 1;
   int b = 2;
   g();
}

void g() {
   int c = 3;
   int *d = h();
   printf("%i\n", *d);
}

int *h() {
   int e = 4;
   return &e;
}
```

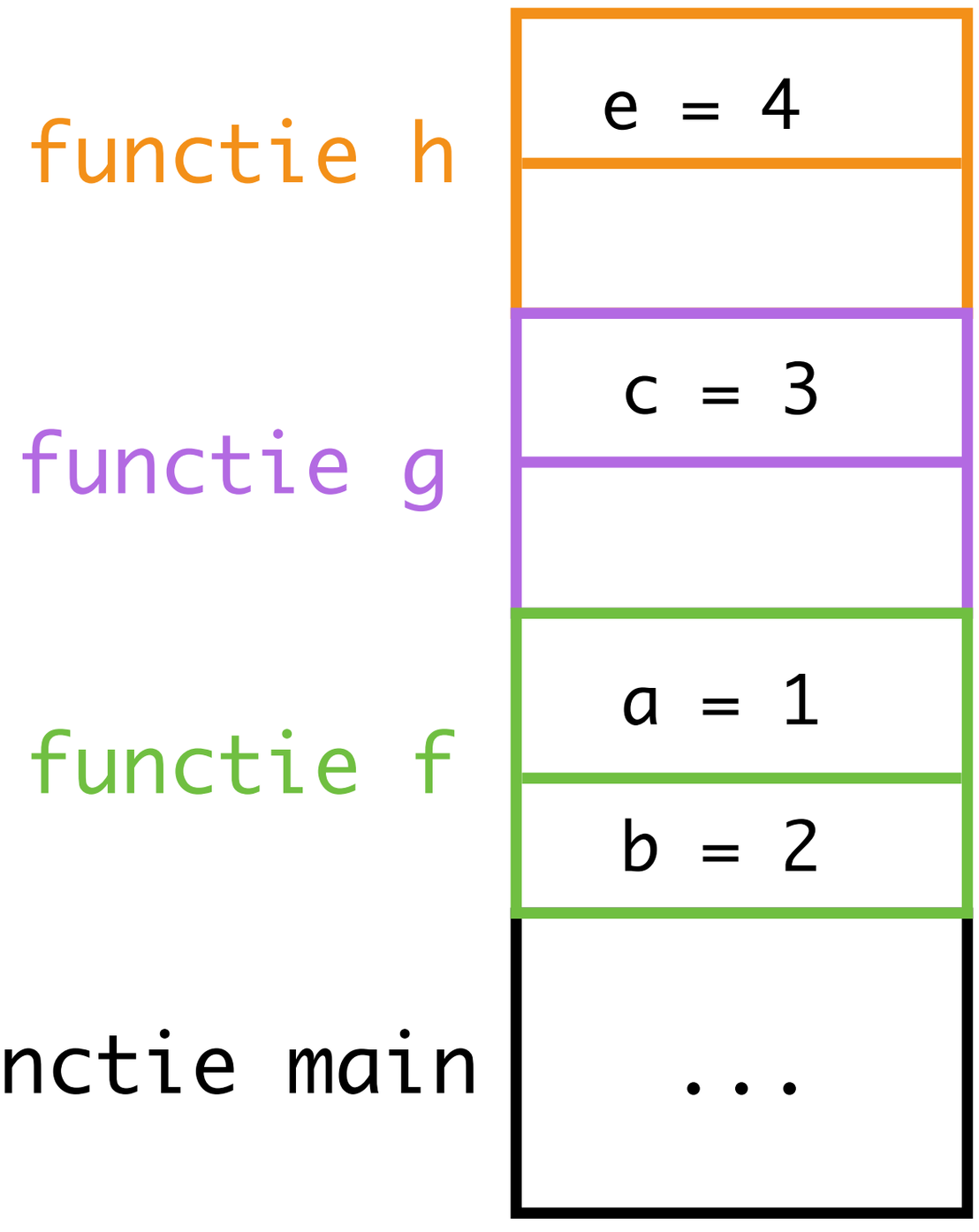
Eerder:

```
void f() {
   int a = 1;
   int b = 2;
   g();
}

void g() {
   int c = 3;
   int *d = h();
   printf("%i\n", *d);
}

int *h() {
   int e = 4;
   return &e;
}
```

Call stack
(functie stack)

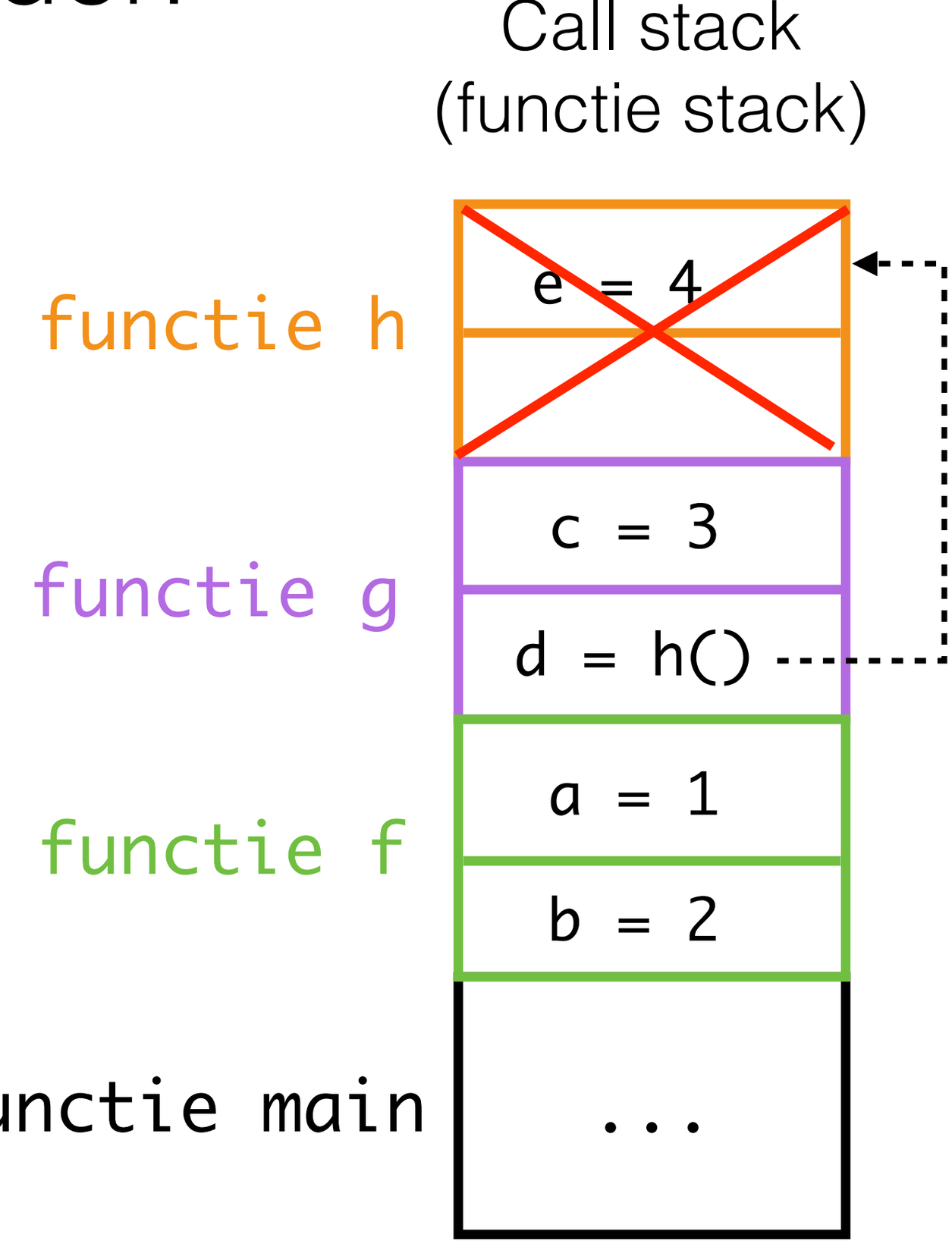


Eerder:

```
void f() {
   int a = 1;
   int b = 2;
   g();
}

void g() {
   int c = 3;
   int *d = h();
   printf("%i\n", *d);
}

int *h() {
   int e = 4;
   return &e;
}
```



malloc

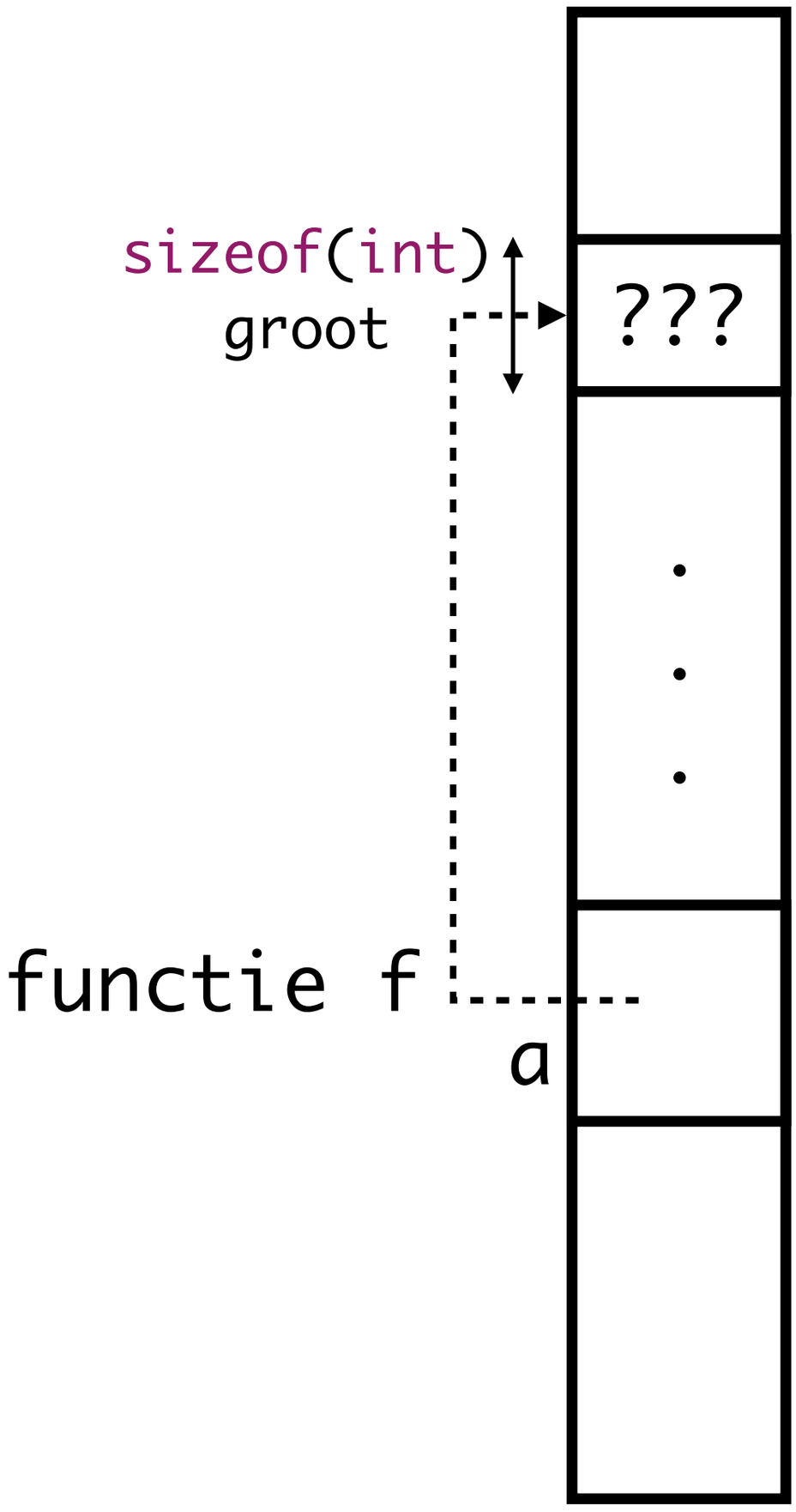
```c
int *f() {
   int *a = malloc(sizeof(int));
   if (a != NULL) {
      *a = 42;
   }
   return a;
}
```

malloc(n) maakt **op de heap** een stuk geheugen van n bytes aan en returnt een pointer naar dit stuk geheugen **of** returnt **NULL** als er geen geheugen meer beschikbaar is

`sizeof(int)`: geeft terug hoeveel bytes er nodig zijn om een integer op te slaan

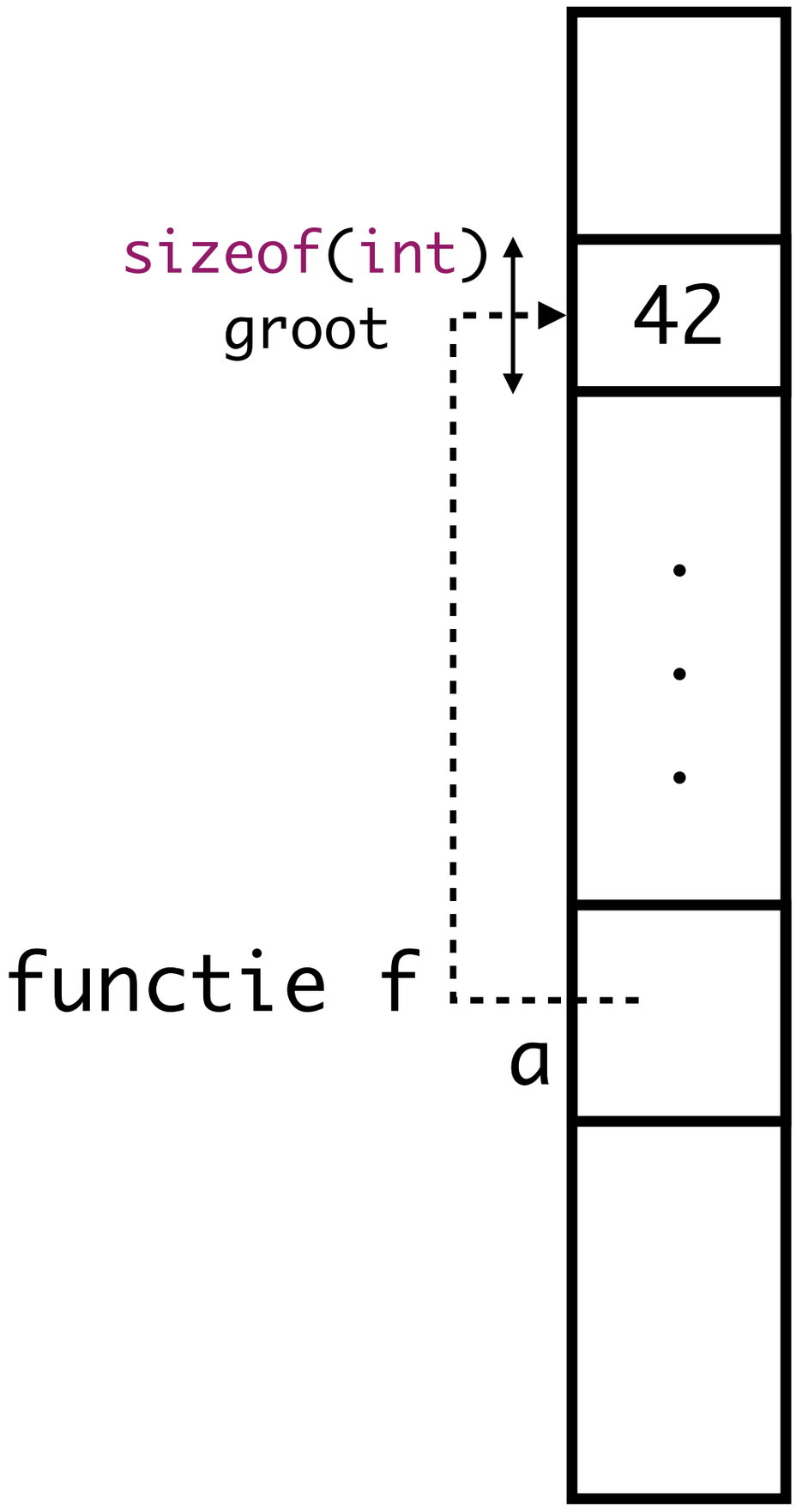
malloc

```
int *f() {
   int *a = malloc(sizeof(int));
   if (a != NULL) {
      *a = 42;
   }
   return a;
}
```



malloc

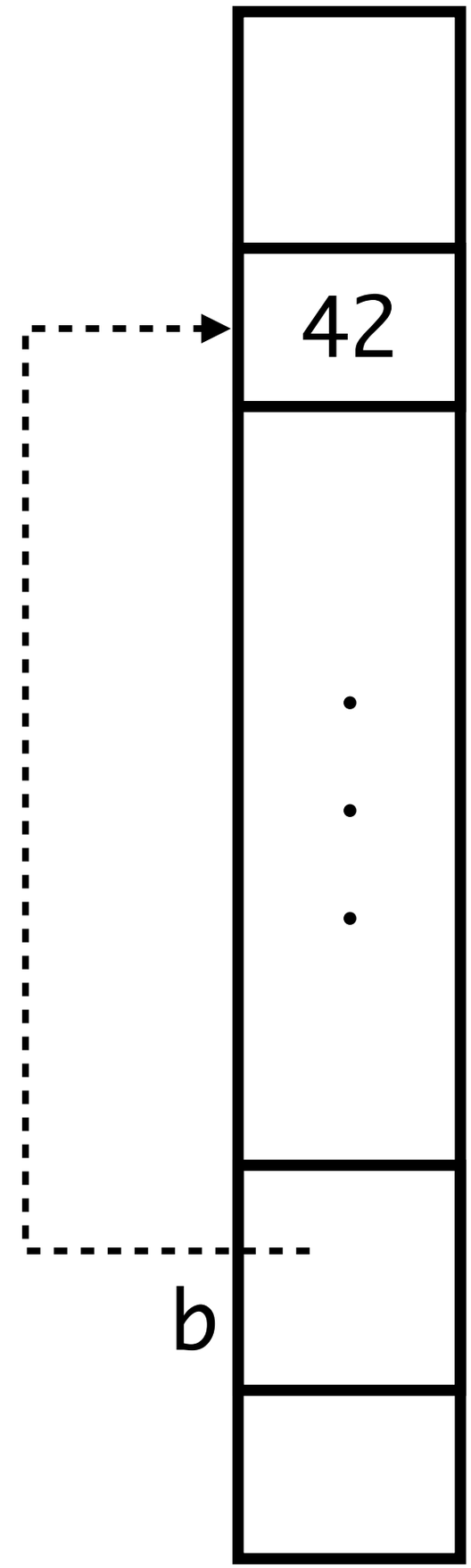
```
int *f() {
   int *a = malloc(sizeof(int));
   if (a != NULL) {
      *a = 42;
   }
   return a;
}
```



malloc

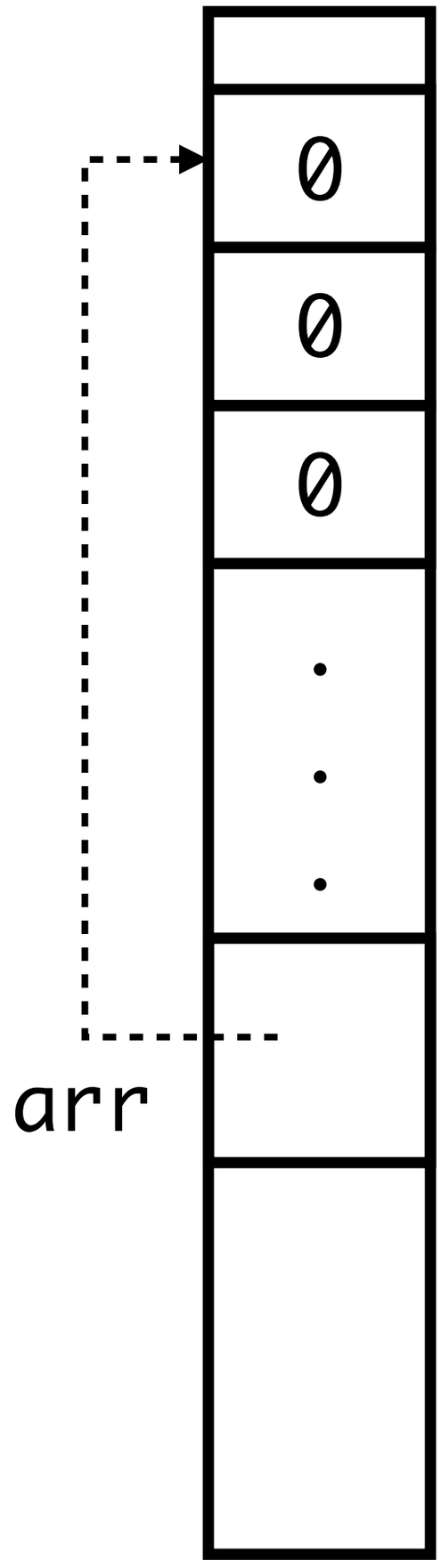
```
int *f() {
   int *a = malloc(sizeof(int));
   if (a != NULL) {
      *a = 42;
   }
   return a;
}

int main() {
   int *b = f();
}
```



calloc

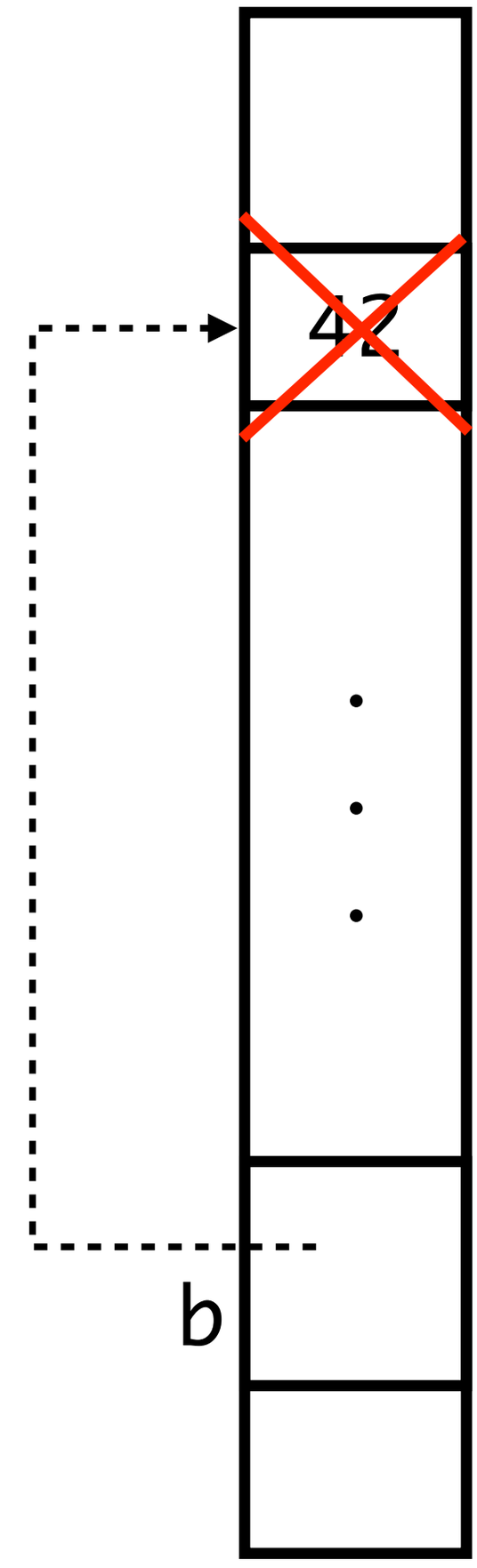
```c
int *g() {
   int *arr = calloc(3, sizeof(int));
   // array overal 0
   if (arr != NULL) {
      return arr;
   }
}
```



free

```
int *f() {
   int *a = malloc(sizeof(int));
   if (a != NULL) {
      *a = 42;
   }
   return a;
}

int main() {
   int *b = f();
   . . .
   free(b);
   *b; // geheugen niet meer toegankelijk
}
```



free

```
int *f() {
  int *a = malloc(sizeof(int));
  if (a != NULL) {
    return a;
  }
}
```

```
int main() {
  int x = 10;
  int *a = &x;
  . . .
  free(a); // a niet dynamisch gealloceerd
}
```

Statisch vs. Dynamisch

Statisch

```
int *f() {
  int a = 1;
  int arr[10];
  return &a;
}

int main() {
  int *x = f();
}
```

Compiler weet op voorhand:

- Hoeveel geheugen er moet voorzien worden
- Wanneer dat geheugen moet vrijgemaakt worden

Dynamisch

```
int *f(int x) {
  int *a = malloc(sizeof(int));
  int *arr = calloc(x, sizeof(int));
  return a;
}

int main() {
  int *x = f(3);
  free(x);
}
```

Compiler weet geen van beide.