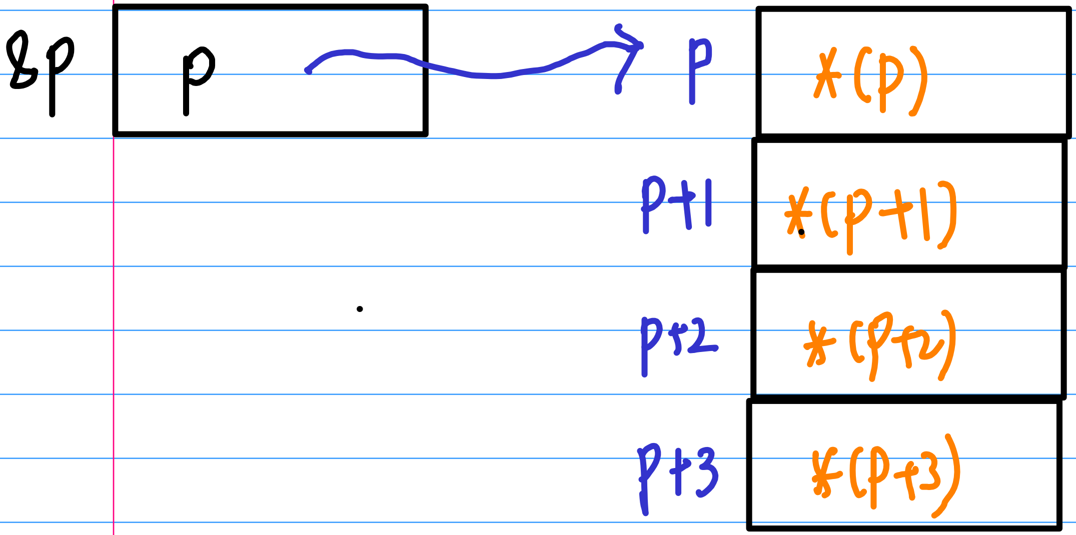
C Programming
Day08.B

2017.10.16

for loop
functions

storage class,
scope,
linkage



p : pointer var

⊕ ⊖

$*(p) \Leftrightarrow p[0]$

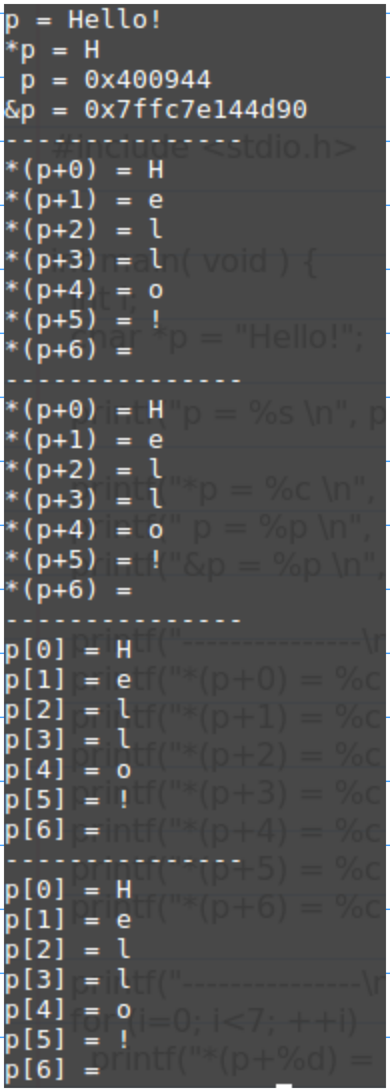
$*(p+1) \Leftrightarrow p[1]$

$*(p+2) \Leftrightarrow p[2]$

$*(p+3) \Leftrightarrow p[3]$

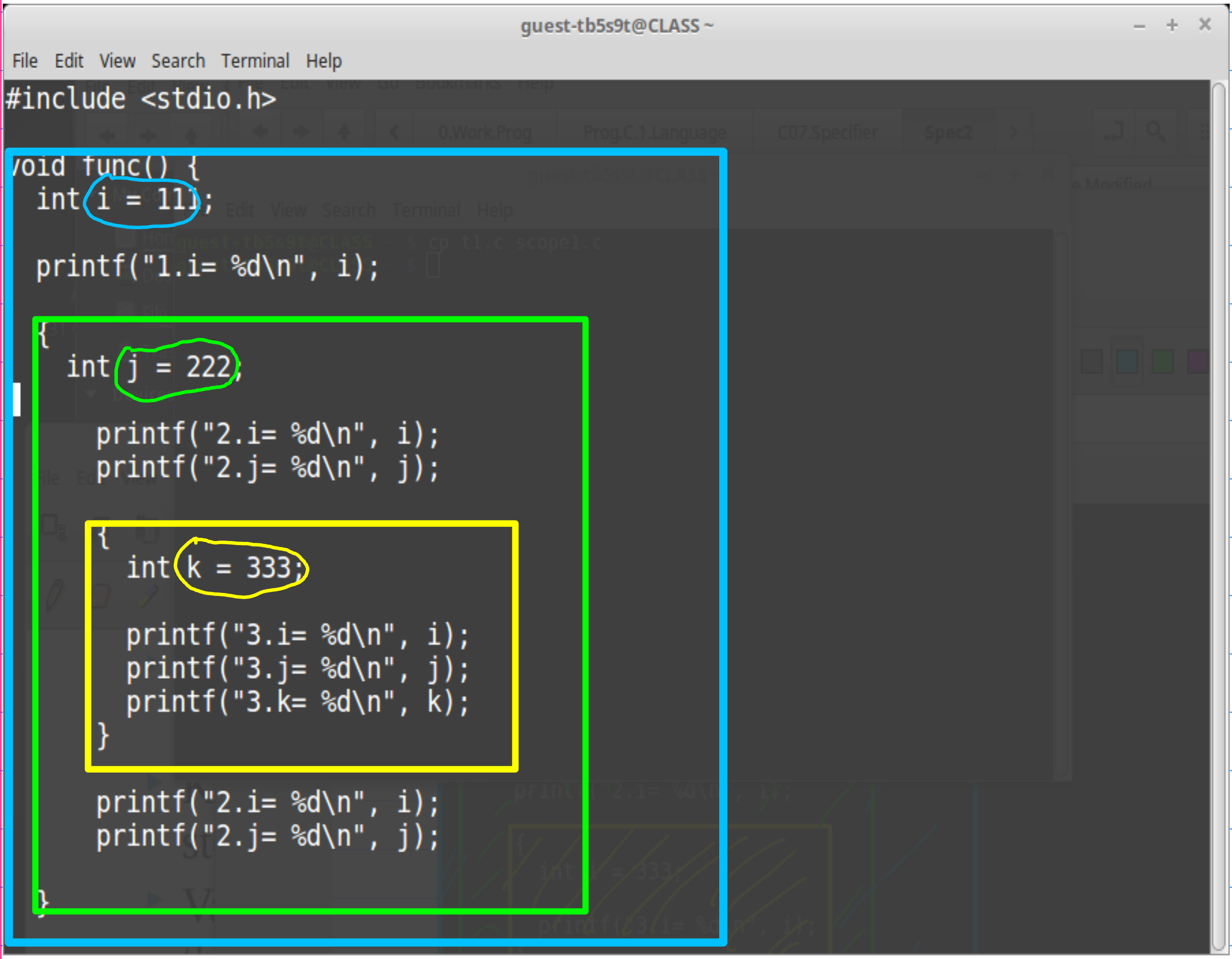
array

```
#include <stdio.h>


int main( void ) {
  int i;
  char *p = "Hello!";

  printf("p = %s \n", p);

  printf("*p = %c \n", *p);
  printf(" p = %p \n",  p);
  printf("&p = %p \n", &p);

  printf("---------------\n");
  printf("*(p+0) = %c \n", *(p+0) );
  printf("*(p+1) = %c \n", *(p+1) );
  printf("*(p+2) = %c \n", *(p+2) );
  printf("*(p+3) = %c \n", *(p+3) );
  printf("*(p+4) = %c \n", *(p+4) );
  printf("*(p+5) = %c \n", *(p+5) );
  printf("*(p+6) = %c \n", *(p+6) );

  printf("---------------\n");
  for (i=0; i<7; ++i)
    printf("*(p+%d) = %c \n", i, *(p+i) );


  printf("---------------\n");
  printf("p[0] = %c \n", p[0] );
  printf("p[1] = %c \n", p[1] );
  printf("p[2] = %c \n", p[2] );
  printf("p[3] = %c \n", p[3] );
  printf("p[4] = %c \n", p[4] );
  printf("p[5] = %c \n", p[5] );
  printf("p[6] = %c \n", p[6] );

  printf("---------------\n");
  for (i=0; i<7; ++i)
    printf("p[%d] = %c \n", i, p[i] );
}
```



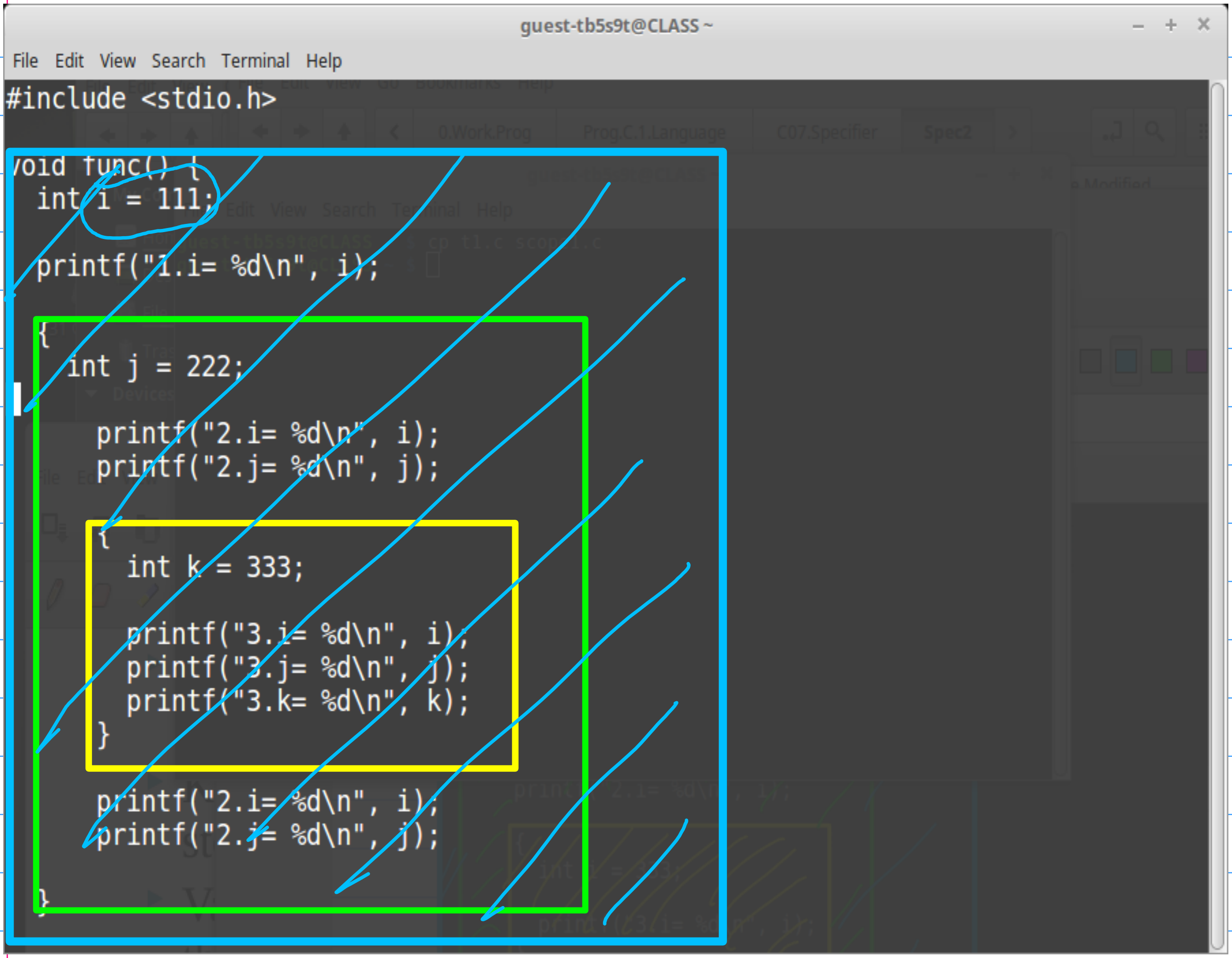
```
p = Hello!
*p = H
 p = 0x400944
&p = 0x7ffc7e144d90
---------------
*(p+0) = H
*(p+1) = e
*(p+2) = l
*(p+3) = l
*(p+4) = o
*(p+5) = !
*(p+6) =
---------------
*(p+0) = H
*(p+1) = e
*(p+2) = l
*(p+3) = l
*(p+4) = o
*(p+5) = !
*(p+6) =
---------------
p[0] = H
p[1] = e
p[2] = l
p[3] = l
p[4] = o
p[5] = !
p[6] =
---------------
p[0] = H
p[1] = e
p[2] = l
p[3] = l
p[4] = o
p[5] = !
p[6] =
```

local variable i,j,k



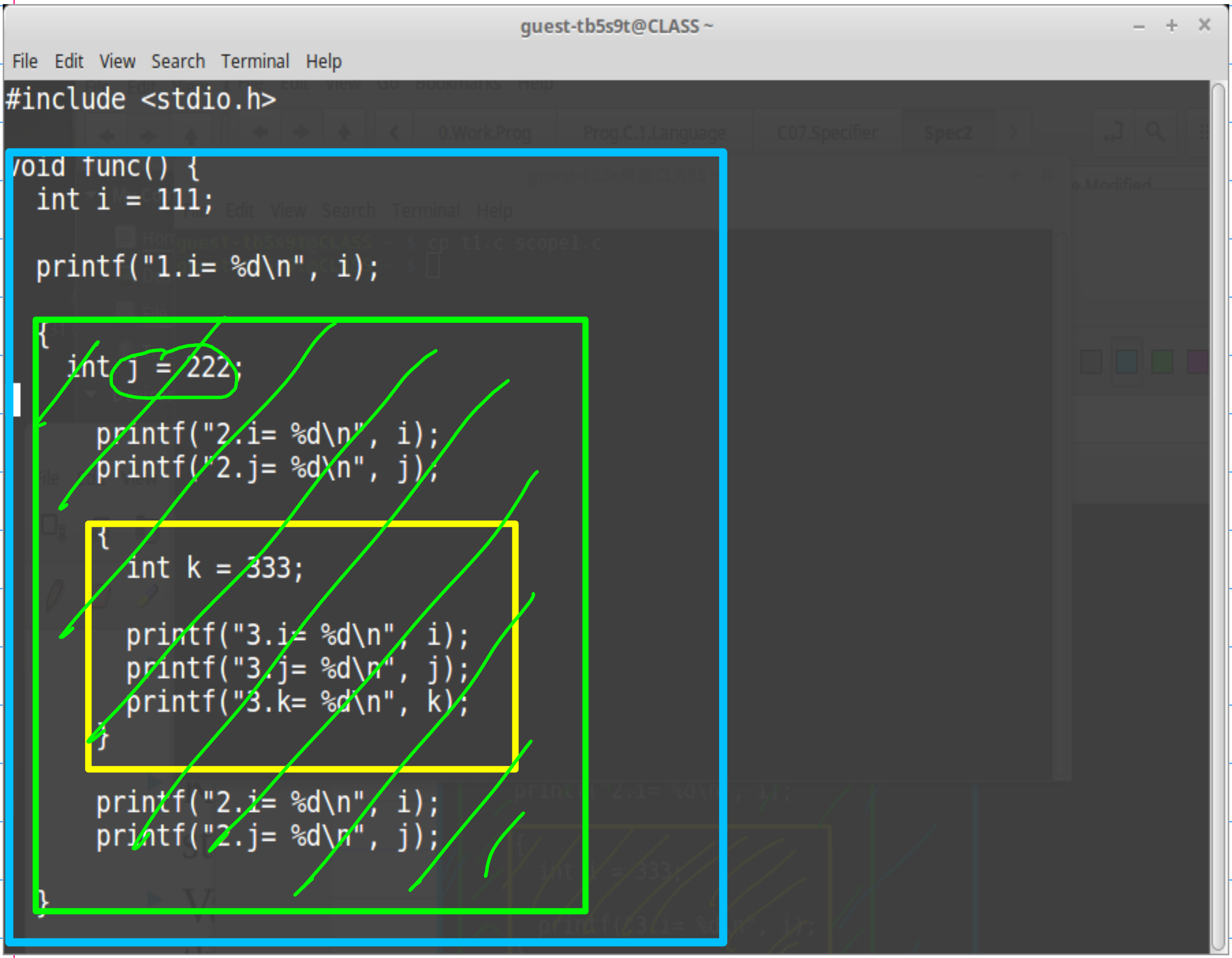
```
#include <stdio.h>

void func() {
  int i = 111;

  printf("1.i= %d\n", i);

  {
    int j = 222;

      printf("2.i= %d\n", i);
      printf("2.j= %d\n", j);

      {
        int k = 333;

        printf("3.i= %d\n", i);
        printf("3.j= %d\n", j);
        printf("3.k= %d\n", k);
      }

      printf("2.i= %d\n", i);
      printf("2.j= %d\n", j);

  }
```

local variable i's scope



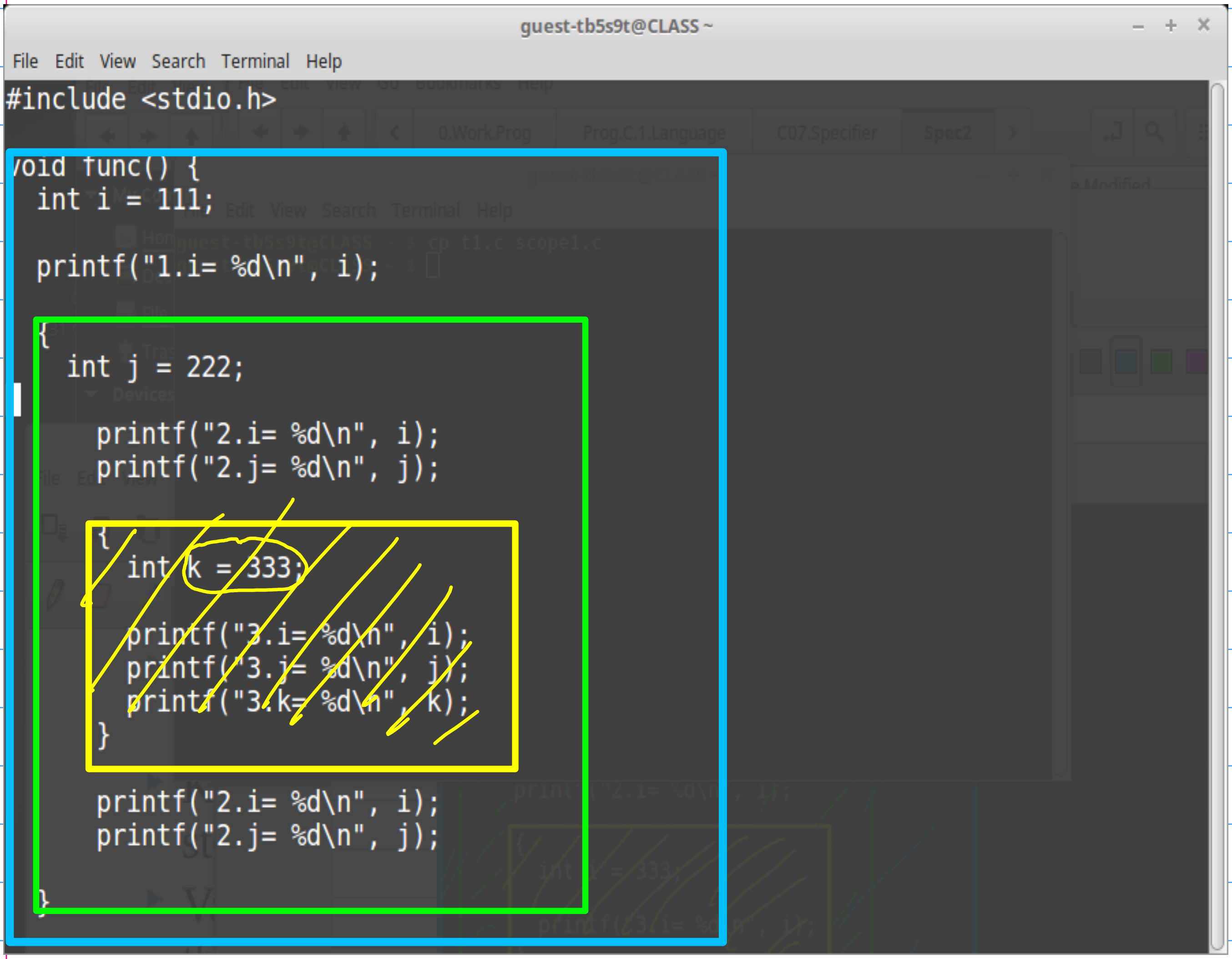
```
#include <stdio.h>

void func() {
  int i = 111;

  printf("1.i= %d\n", i);

  {
    int j = 222;

      printf("2.i= %d\n", i);
      printf("2.j= %d\n", j);

      {
        int k = 333;

        printf("3.i= %d\n", i);
        printf("3.j= %d\n", j);
        printf("3.k= %d\n", k);
      }

      printf("2.i= %d\n", i);
      printf("2.j= %d\n", j);

  }
```

local variable 's scope



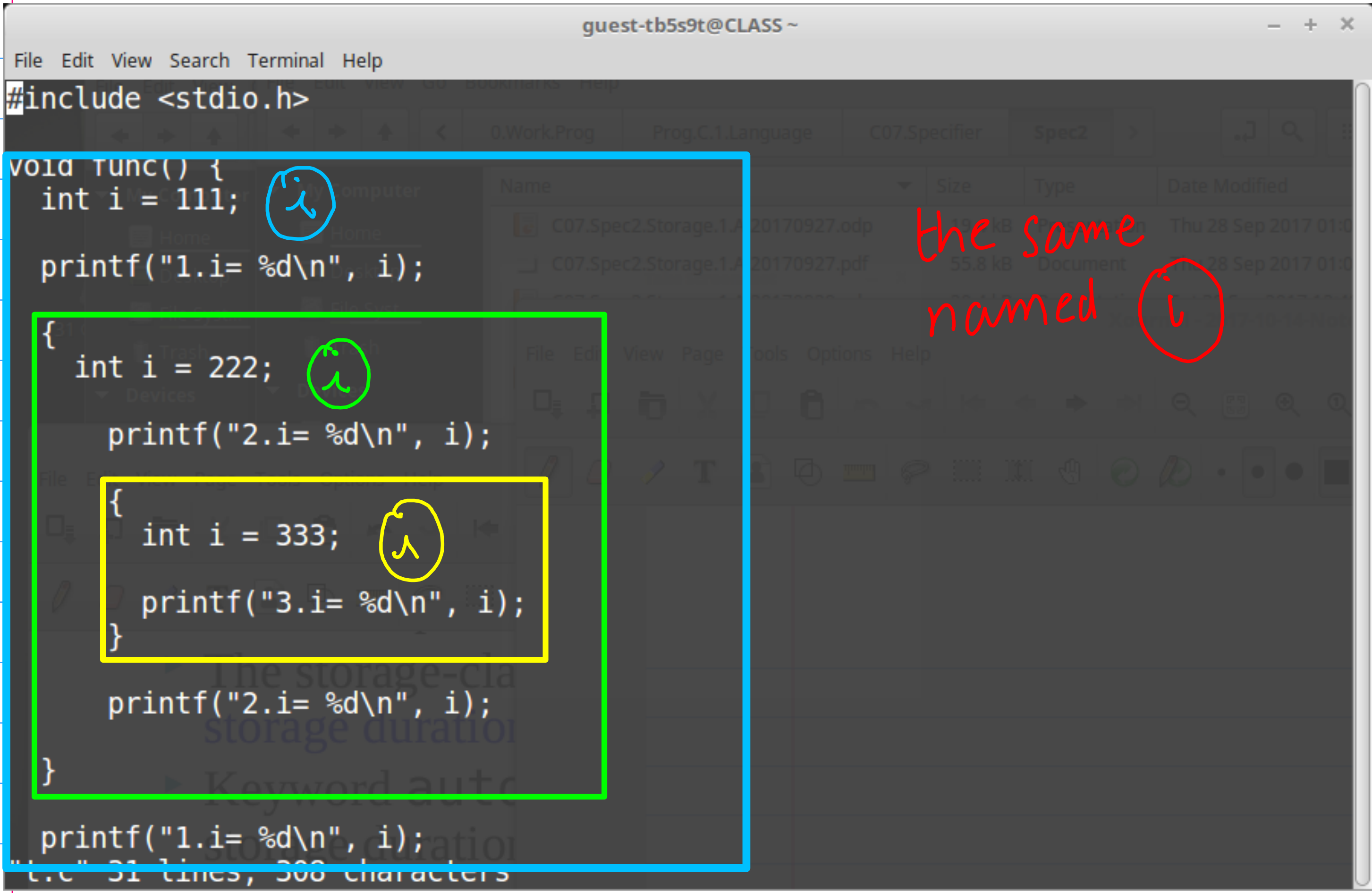
```
#include <stdio.h>

void func() {
 int i = 111;

 printf("1.i= %d\n", i);

 {
  int j = 222;

   printf("2.i= %d\n", i);
   printf("2.j= %d\n", j);

   {
    int k = 333;

    printf("3.i= %d\n", i);
    printf("3.j= %d\n", j);
    printf("3.k= %d\n", k);
   }

   printf("2.i= %d\n", i);
   printf("2.j= %d\n", j);

 }
```

local variable k's scope



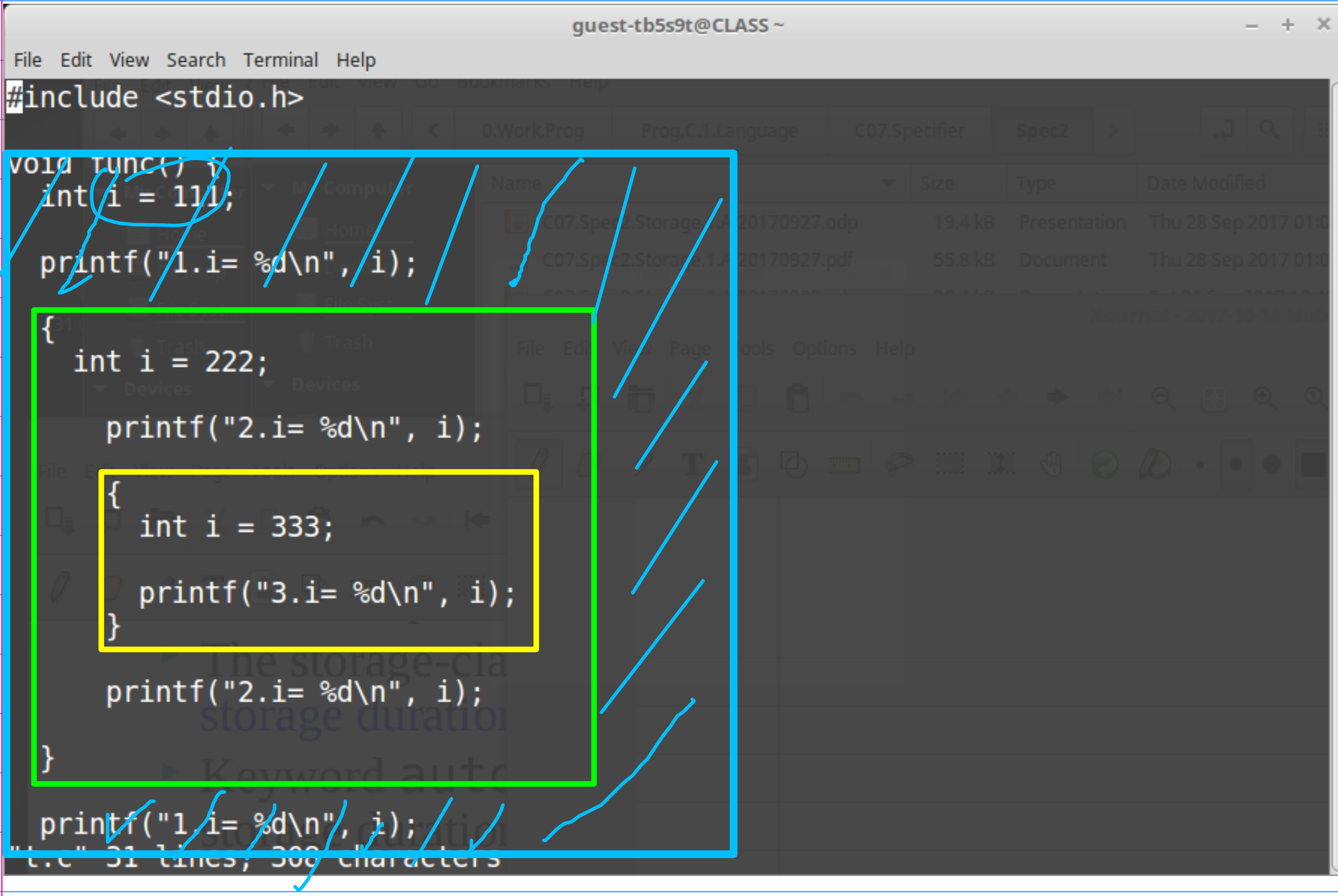
```
#include <stdio.h>

void func() {
 int i = 111;

 printf("1.i= %d\n", i);

 {
  int j = 222;

   printf("2.i= %d\n", i);
   printf("2.j= %d\n", j);

   {
    int k = 333;

    printf("3.i= %d\n", i);
    printf("3.j= %d\n", j);
    printf("3.k= %d\n", k);
   }

   printf("2.i= %d\n", i);
   printf("2.j= %d\n", j);

 }
```

Local Variable hides other variable with the same name



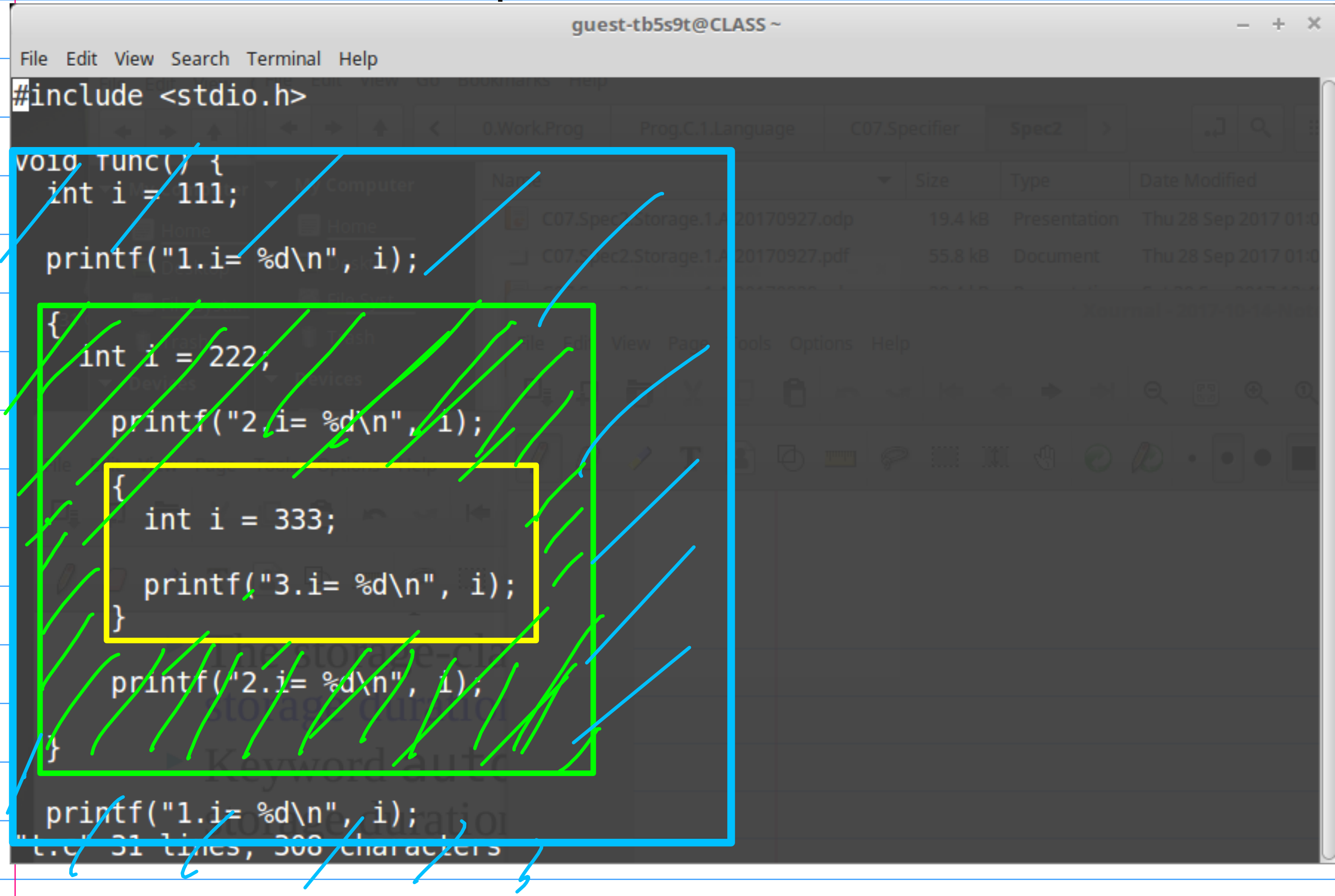
```
#include <stdio.h>

void func() {
 int i = 111;

 printf("1.i= %d\n", i);

 {
  int i = 222;

   printf("2.i= %d\n", i);

   {
    int i = 333;

    printf("3.i= %d\n", i);
   }

   printf("2.i= %d\n", i);

 }

 printf("1.i= %d\n", i);
"t.c" 31 lines, 308 characters
```

1st i's scope



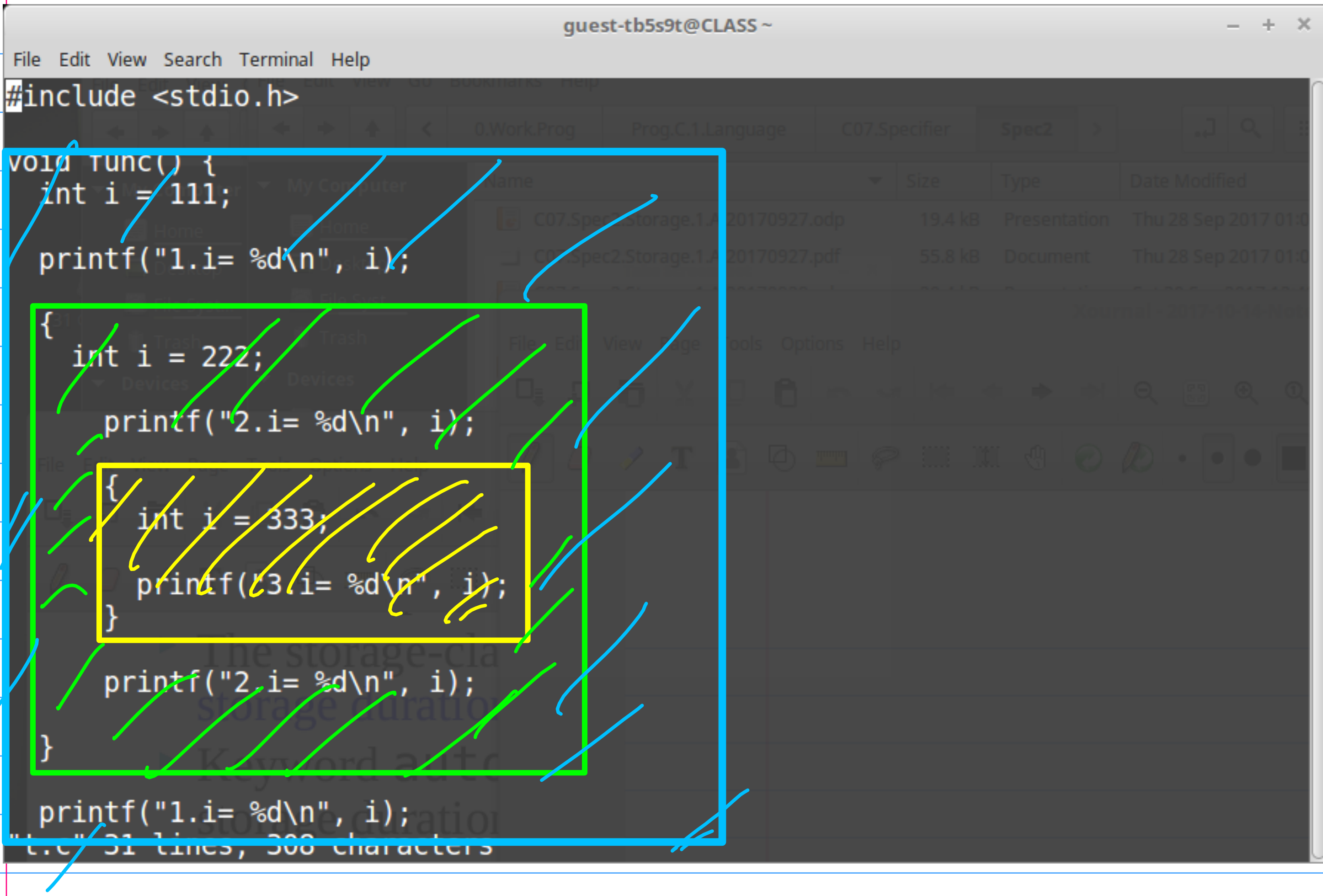
```
#include <stdio.h>

void func() {
  int i = 111;

  printf("1.i= %d\n", i);

  {
    int i = 222;

      printf("2.i= %d\n", i);

      {
        int i = 333;

        printf("3.i= %d\n", i);
      }

      printf("2.i= %d\n", i);

  }

  printf("1.i= %d\n", i);
"t.c" 31 lines, 308 characters
```

1st i's scope
2nd i's scope



```
#include <stdio.h>

void func() {
  int i = 111;

  printf("1.i= %d\n", i);

  {
    int i = 222;

      printf("2.i= %d\n", i);
      {
        int i = 333;

        printf("3.i= %d\n", i);
      }

      printf("2.i= %d\n", i);

  }

  printf("1.i= %d\n", i);
"t.c" 31 lines, 308 characters
```

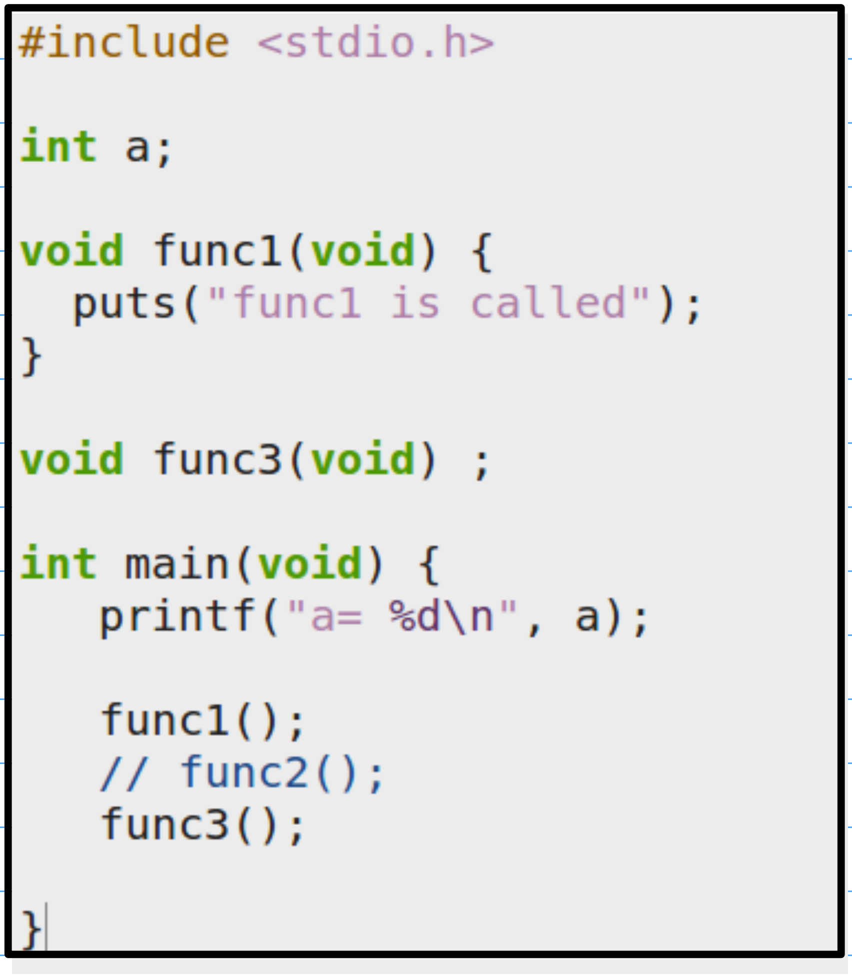
1st i's scope
2nd i's scope
3rd i's scope



```
#include <stdio.h>

void func() {
 int i = 111;

 printf("1.i= %d\n", i);

 {
  int i = 222;

   printf("2.i= %d\n", i);

   {
    int i = 333;

    printf("3.i= %d\n", i);
   }

   printf("2.i= %d\n", i);

 }

 printf("1.i= %d\n", i);
"t.c" 31 lines, 308 characters
```

separate compilation --- multiple-source program

t1.c

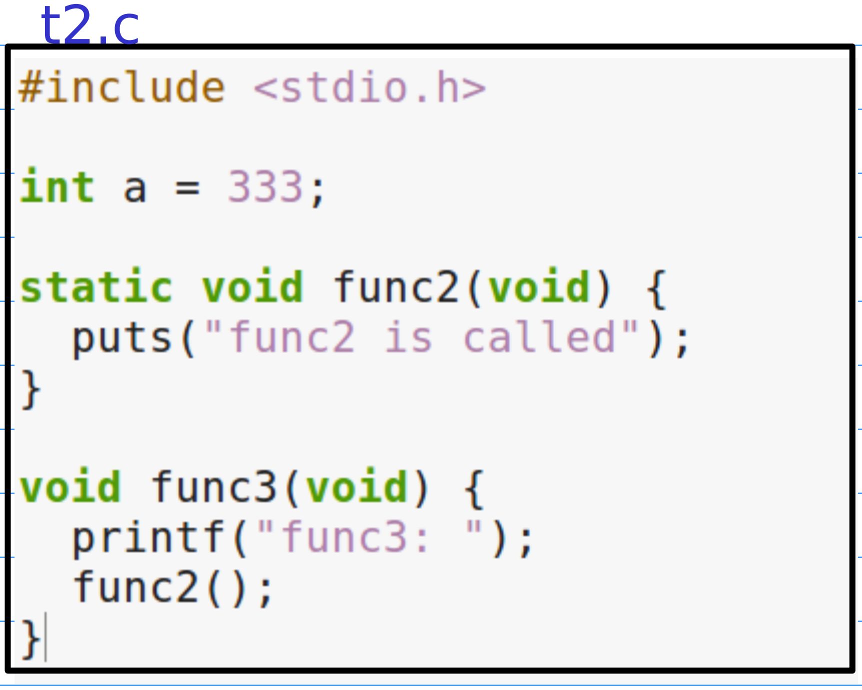


```c
#include <stdio.h>

int a;

void func1(void) {
  puts("func1 is called");
}

void func3(void) ;

int main(void) {
   printf("a= %d\n", a);

   func1();
   // func2();
   func3();

}
```

gcc -c t1.c

=> t1.o

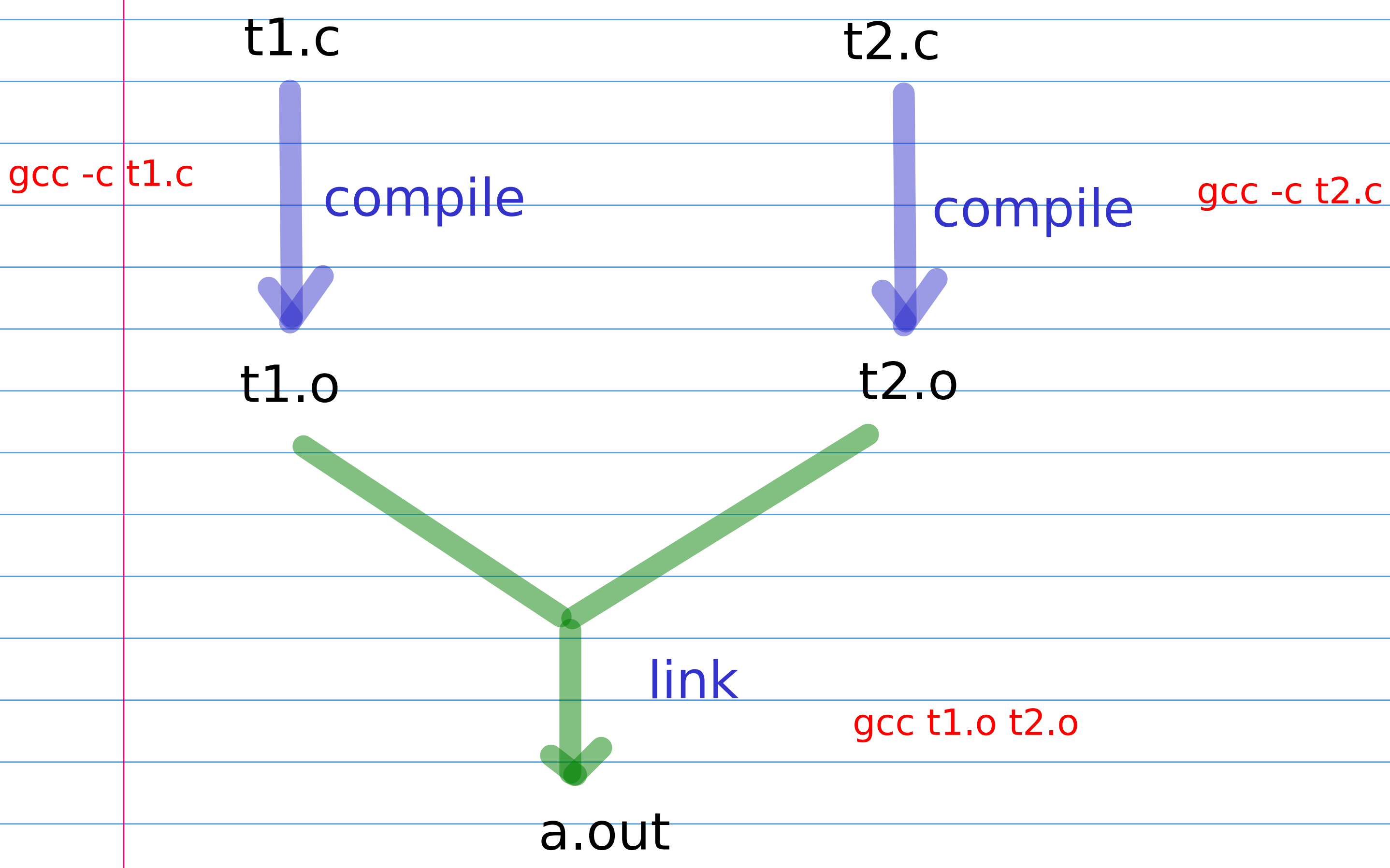
t2.c



```c
#include <stdio.h>

int a = 333;

static void func2(void) {
  puts("func2 is called");
}

void func3(void) {
  printf("func3: ");
  func2();
}
```

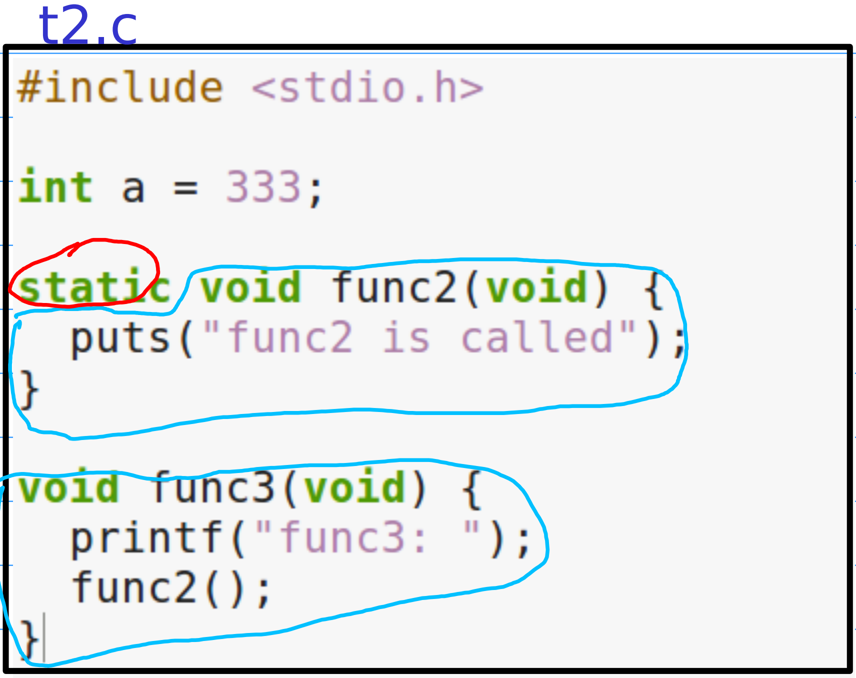
gcc -c t2.c

=> t2.o



```
gcc -c t1.c
gcc -c t2.c
gcc t1.o t2.o
```

t2.c



```c
#include <stdio.h>

int a = 333;

static void func2(void) {
  puts("func2 is called");
}

void func3(void) {
  printf("func3: ");
  func2();
}
```

func2 can be used only within t2.c

↓

Useful to hide implementation details.

func2 has an internal linkage

thus, it cannot called in main()
which is defined in t1.c

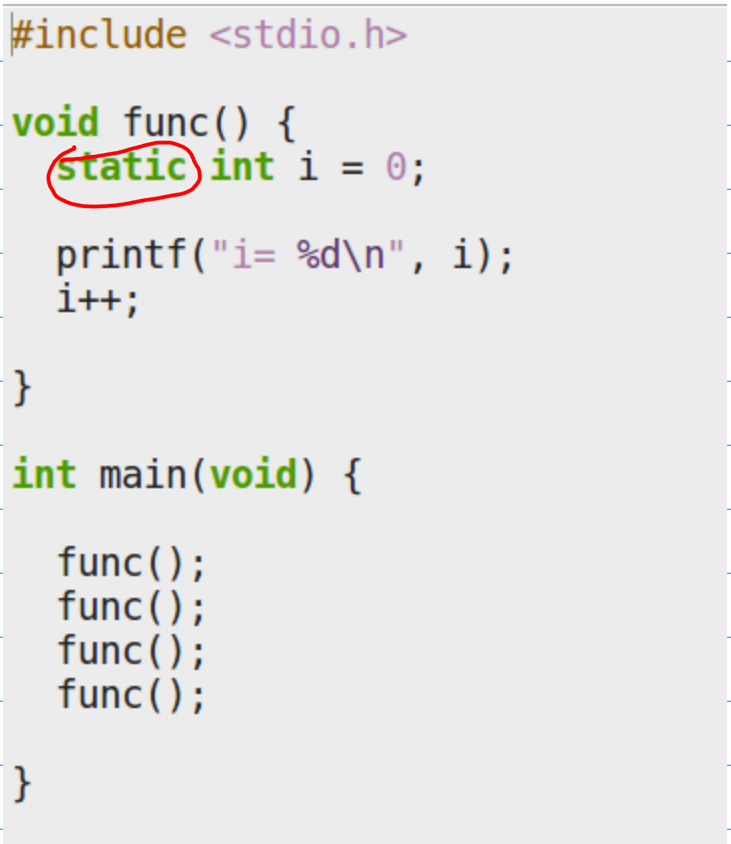
but func3 has an external linkage

and it is called in main()

func2 cannot be called by main() in t1.c
but can be called by func3() in t2.c

func2 is known only in t2.c
because of static keyword

static storage class example



```c
#include <stdio.h>

void func() {
  static int i = 0;

  printf("i= %d\n", i);
  i++;

}

int main(void) {

  func();
  func();
  func();
  func();

}
```

local variable --> automatic storage

static local variable --> static storage