CS2100 COMPUTER ORGANISATION

http://www.comp.nus.edu.sg/~cs2100/

Lecture #4d

Pointers and Functions

Questions?

IMPORTANT: DO NOT SCAN THE QR CODE IN THE VIDEO RECORDINGS. THEY NO LONGER WORK

Ask at
https://sets.netlify.app/module/676ca3a07d7f5ffc1741dc65

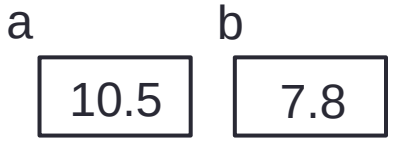
OR

Scan and ask your questions here!
(May be obscured in some slides)

4. Pass-by-Value and Scope Rule (1/4)

- In C, the actual parameters are passed to the formal parameters by a mechanism known as pass-by-value.

```
int main(void) {

  double a = 10.5, b = 7.8;
  printf("%.2f\n", sqrt_sum_square(3.2, 12/5);
  printf("%.2f\n", sqrt_sum_square(a, a+b);
  return 0;
}
```



Actual parameters:

10.5 and 18.3

```
double sqrt_sum_square(double x, double y) {

  double sum_square;
  sum_square = pow(x,2) + pow(y,2);
  return sqrt(sum_square);
}
```

Formal parameters:

4. Pass-by-Value and Scope Rule (2/4)

- Formal parameters are local to the function they are declared in.
- Variables declared within the function are also local to the function.
- Local parameters and variables are only accessible in the function they are declared – scope rule.
- When a function is called, an activation record is created in the call stack, and memory is allocated for the local parameters and variables of the function.
- Once the function is done, the activation record is removed, and memory allocated for the local parameters and variables is released.
- Hence, local parameters and variables of a function exist in memory only during the execution of the function. They are called automatic variables.
- In contrast, static variables exist in the memory even after the function is executed.

4. Pass-by-Value and Scope Rule (3/4)

- What's wrong with this code?

```
int f(int);

int main(void) {
  int a;
  ...
}

int f(int x) {
  return a + x;
}
```

Answer:
Variable a is local to main(), not f(). Hence, variable a cannot be used in f().

4. Pass-by-Value and Scope Rule (4/4)

- Trace this code by hand and write out its output.

A void function is a function that does not return any value.

```
In main, before: a=2, b=3
In g, before: a=2, b=3
In g, after : a=102, b=203
In main, after : a=2, b=3
```

```c
#include <stdio.h>
void g(int, int);

int main(void) {
  int a = 2, b = 3;

  printf("In main, before: a=%d, b=%d\n", a, b);
  g(a, b);
  printf("In main, after : a=%d, b=%d\n", a, b);
  return 0;
}

void g(int a, int b) {
  printf("In g, before: a=%d, b=%d\n", a, b);
  a = 100 + a;
  b = 200 + b;
  printf("In g, after : a=%d, b=%d\n", a, b);
}
```

PassByValue.c

main

addr	name	val
_	a	2
_	b	3

g

addr	name	val
_	a	102
_	b	203

4.1 Consequence of Pass-by-Value

- Can this code be used to swap the values in **a** and **b**?

```
In main, before: a=2, b=3
In main, after : a=2, b=3
```

```c
#include <stdio.h>
void swap(int, int);

int main(void) {
  int a = 2, b = 3;

  printf("In main, before: a=%d, b=%d\n", a, b);
  swap(a, b);
  printf("In main, after : a=%d, b=%d\n", a, b);
  return 0;
}

void swap(int a, int b) {
  int temp = a;
  a = b;
  b = temp;
}
```

SwapIncorrect.c

No

5. Function with Pointer Parameters (1/3)

- A function may not return any value (called a void function), or it may return a value.
- All parameters and variables in a function are local to the function (scope rule).
- Arguments from a caller are passed by value to a function's parameters.
- How do we then allow a function to return more than one value, or modify values of variables defined outside it?
- An example is swapping two variables. How can we write a function to do that? The previous slide shows a negative example.

5. Function with Pointer Parameters (2/3)

- What happens in SwapIncorrect.c?
- It's all about pass-by-value and scope rule!

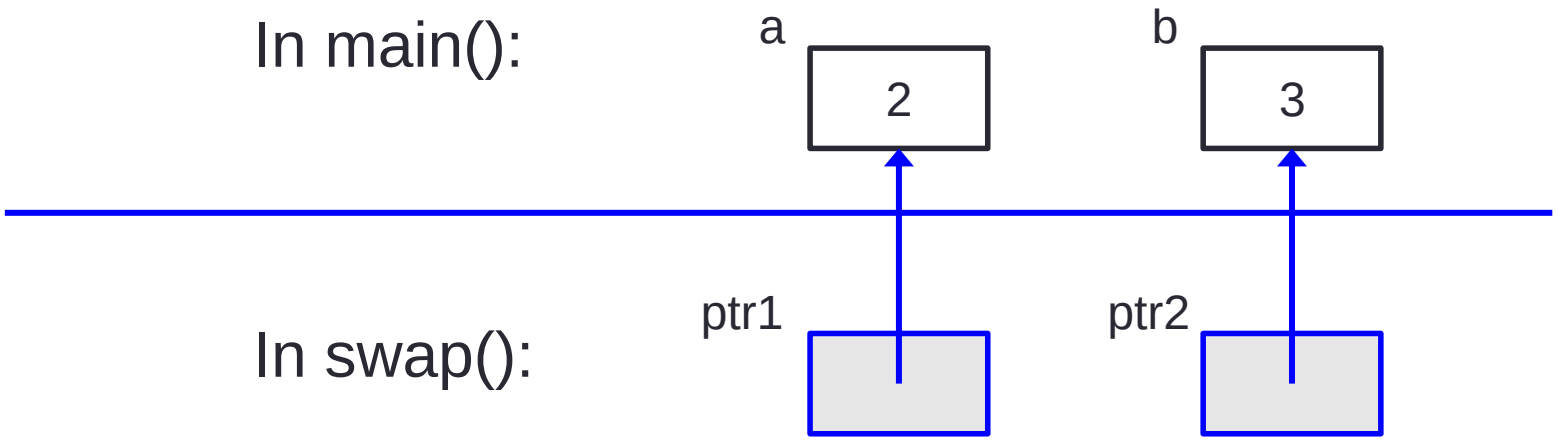
In main():

In swap():

- No way for swap() to modify the values of variables that are outside its scope (i.e. a and b), unless...

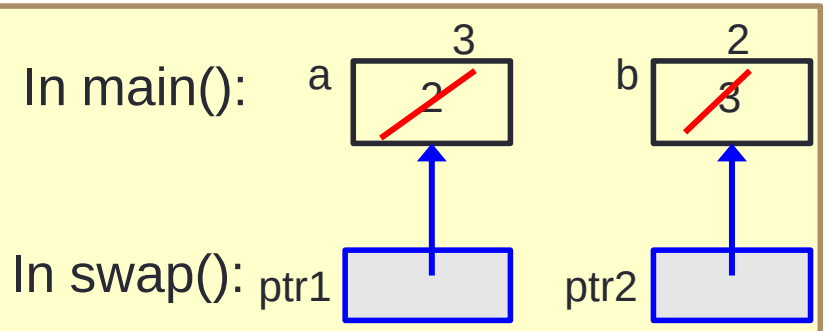
5. Function with Pointer Parameters (3/3)

- The only way for a function to modify the value of a variable outside its scope, is to find a way for the function to access that variable
- Solution: Use pointers!



5.1 Function To Swap Two Variables

```
#include <stdio.h>

void swap(int *, int *);

int main(void) {
   int a, b;

   printf("Enter two integers: ");
   scanf("%d %d", &var1, &var2);

   swap( &a, &b );

   printf("var1 = %d; var2 = %d\n", var1, var2);
   return 0;
}

void swap(int *ptr1, int *ptr2) {
   int temp;
   temp = *ptr1; *ptr1 = *ptr2; *ptr2 = temp;
}
```

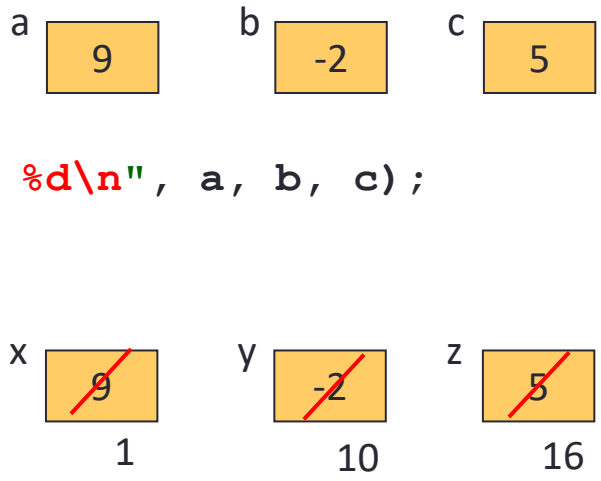


SwapCorrect.c

5.2 Examples (1/4)

Example1.c

```
#include <stdio.h>
void f(int, int, int);

int main(void) {
    int a = 9, b = -2, c = 5;
    f(a, b, c);
    printf("a = %d, b = %d, c = %d\n", a, b, c);
    return 0;
}

void f(int x, int y, int z) {
    x = 3 + y;
    y = 10 * x;
    z = x + y + z;
    printf("x = %d, y = %d, z = %d\n", x, y, z);
}
```

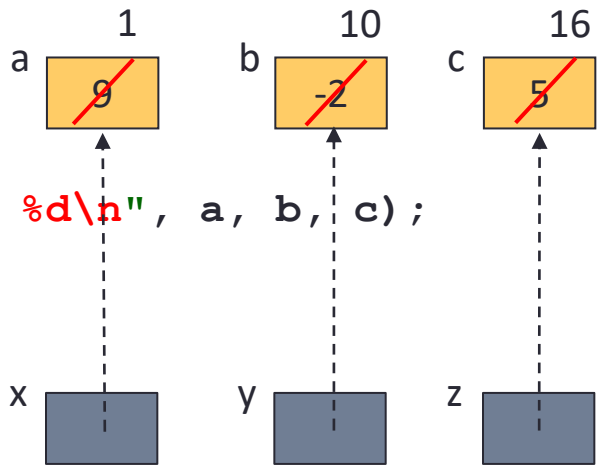



```
x = 1, y = 10, z = 16
a = 9, b = -2, c = 5
```

5.2 Examples (2/4)

Example2.c

```
#include <stdio.h>
void f(int *, int *, int *);

int main(void) {
    int a = 9, b = -2, c = 5;
    f(&a, &b, &c);
    printf("a = %d, b = %d, c = %d\n", a, b, c);
    return 0;
}

void f(int *x, int *y, int *z)
{
    *x = 3 + *y;
    *y = 10 * *x;
    *z = *x + *y + *z;
    printf("*x = %d, *y = %d, *z = %d\n", *x, *y, *z);
}
```



*x is a, *y is b, and *z is c!

```
*x = 1, *y = 10, *z = 16
a = 1, b = 10, c = 16
```

5.2 Examples (3/4)

Example3.c

```
#include <stdio.h>
void f(int *, int *, int *);

int main(void) {
    int a = 9, b = -2, c = 5;
    f(&a, &b, &c);
    printf("a = %d, b = %d, c = %d\n", a, b, c);
    return 0;
}

void f(int *x, int *y, int *z)
{
    *x = 3 + *y;
    *y = 10 * *x;
    *z = *x + *y + *z;
    printf("x = %d, y = %d, z = %d\n", x, y, z);
}
```

Compiler warnings, because x, y, z are NOT integer variables!
They are addresses (or pointers).

5.2 Examples (4/4)

Example4.c

```
#include <stdio.h>
void f(int *, int *, int *);

int main(void) {
    int a = 9, b = -2, c = 5;
    f(&a, &b, &c);
    printf("a = %d, b = %d, c = %d\n", a, b, c);
    return 0;
}

void f(int *x, int *y, int *z)
{
    *x = 3 + *y;
    *y = 10 * *x;
    *z = *x + *y + *z;
    printf("x = %p, y = %p, z = %p\n", x, y, z);
}
```

Use %p for pointers.

Addresses of variables a, b and c.
(Values change from run to run.)

```
x = ffbff78c, y = ffbff788, z = ffbff784
a = 1, b = 10, c = 16
```

Quiz

- Please complete Pointers and Functions Quiz 2 before 3 pm on 23 August 2022.

End of File