

COMP20005 Workshop Week 6

Preparation:

- open grok in a tab
- open github.com/anhvir/c205/ws6.pdf in another tab

Plan:

1 Recursive Functions: When & How
Scopes

2 **15 min Group Work:** ex. 6.2 (*discussion: scopes*)
ex. 5.14 (*coding: log**)

3 Pointers, Pointers as Function Arguments

4 **Individual Work + Zoom + grok:**

- *try today's sample test if you haven't*
- *ex. 6.5, 6.6 (coding: sorting 2, 3 integers)*
- *try quizzes in ws6.pdf pp 18-21*
- *ex. 6.9 (coding: coin change revisited)*

5 Notes on the coming MST, walking through a sample test.

Recursive functions

- **What:** A function that calls itself.
- **When:**
 - a task of size n can be reduced to a task (or tasks) of size $< n$, and
 - solution for small, trivial n can be easily found.
- Example: $f(n) = \text{compute } n!$, ie. $1*2*3*...*(n-1)*n$
 - $f(n) = f(n-1)*n$ and $f(0) = 1$;
- **How** to write a recursive function?
 - solve all trivial cases (*base cases*) first, then
 - solve the general case by calling the function itself.

```
int fact(int n){
```

}

Recursive functions: Example

- How to write a recursive function?
 - solve all trivial cases (*base cases*) first, then
 - solve the general case by calling the function itself.
- Example: write a recursive function to return the n-th Fibonacci number.

The Fibonacci numbers are

1, 1, 2, 3, 5, 8, ...

where the 0-th and 1-st numbers are 1, and each subsequent number is the sum of its 2 immediate predecessors.

Recursive functions: Example

- How to write a recursive function?
 - solve all trivial cases (*base cases*) first, then
 - solve the general case by calling the function itself.
- Example: write a recursive function to return the n-th Fibonacci number.

The Fibonacci numbers are

1, 1, 2, 3, 5, 8, ...

where the 0-th and 1-st numbers are 1, and each subsequent number is the sum of its 2 immediate predecessors.

And so:

General case: $f(n) = f(n-1) + f(n-2)$ when $n > 1$

Base cases: $f(0) = f(1) = 1$ when $n = 0, 1$

Recursive functions: Example

Example: write a recursive function to return the n-th Fibonacci number. The Fibonacci numbers are **1, 1, 2, 3, 5, 8, ...** where the 0-th and 1-st numbers are 1, and each subsequent number is the sum of its 2 immediate predecessors.

```
int fib(int n) {  
    }  
}
```

Scopes, local & global variables

```
#include <stdio.h>
int fact(int n);
```

```
int main(int argc, char *argv[]){
    int n= 3, val;
    val= fact(n);
    printf("%d! = %d\n", n, val);
    return 0;
}
```

argc, argv,
n, and val
available
here

```
int fact(int n) {
    int i, f= 1;
    for (i=1; i<=n; i++) {
        f *= i;
    }
    return f;
}
```

n, i, and f
available
here

function
fact
available
here

Scopes, local & global variables

```
#include <stdio.h>
int ga, gb;
int fact(int n);
```

```
int main(int argc, char *argv[]){
    int n= 3, val;
    val= fact(n);
    printf("%d! = %d\n", n, val);
    return 0;
}
```

```
int fact(int n) {
    int i, f= 1;
    for (i=1; i<=n; i++) {
        f *= i;
    }
    return f;
}
```

scope of
local
variables
argc, argv,
n, and val

scope of
local
variables n,
i, and f

scope of function fact

scope of global variables ga and gb

A Rule: Never use global variables

```
#include <stdio.h>
int ga, gb;
int fact(int n);
```

```
int main(int argc, char *argv[ ]) {
    int n= 3, val;
    val= fact(n);
    printf("%d! = %d\n", n, val);
    return 0;
}
```

scope of
local
variables
argc, argv,
n, and val

```
int fact(int n) {
    int i, f= 1;
    for (i=1; i<=n; i++) {
        f *= i;
    }
    return f;
}
```

scope of
local
variables n,
i, and f

scope of function fact

Group work: 5.14 (W5X) 6.2 (W6), time limit: 15 min

6.2: For each of the 3 marked points, write down a list of all of the program-declared variables and functions that are in scope at that point, and for each identifier, its type. Don't forget main, argc, argv. Where there are more than one choice of a given name, be sure to indicate which one you are referring to.

```
1 int bill(int jack, int jane);
2 double jane(double dick, int fred, double dave);
3
4 int trev;
5
6 int main(int argc, char *argv[ ]) {
7     double beth;
8     int pete, bill;      /* -- point #1 -- */
9     return 0;
10 }
11 int bill (int jack, int jane) {
12     int mary;
13     double zack;        /* -- point #2 -- */
14     return 0;
15 }
16 double jane(double dick, int fred, double dave) {
17     double trev;        /* -- point #3 -- */
18     return 0.0;
19 }
```

6.3

What's wrong with these functions?

Should we employ variable/function names in this way?

```
1 int dumb(int dumb) {  
2     return dumb+1;  
3 }  
  
11 int dumber (int dumber) {  
12     int dumber= 6;  
13     return dumb(dumber+1);  
14 }
```

Never use a same identifier for more than one object/purpose!

Variable and address, operators & and *

What happens when:

`int n=10;` is executed?

Variable and address, operators & and *

What happens when:

`int n=10;` is executed?

unary operators & and * : referencing and dereferencing

```
int n=10;
```

```
int *pn;
```

```
pn= &n;
```

Check your understanding:

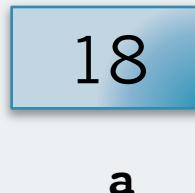
- a) The datatype of `pn` is _____
- b) If `n` is at the address 4444, then `pn` has the value of _____
- c) The value of `*pn` is _____
- d) After

```
*pn= 100;
```

the value of `pn` is _____, of `n` is _____

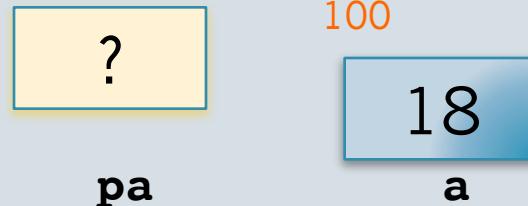
Pointers: check your understanding

int a= 18;



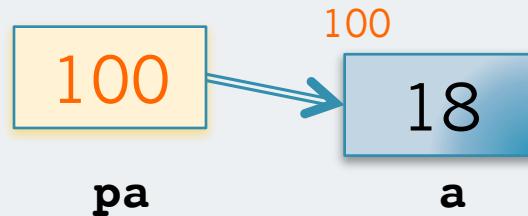
a is a location in the memory, interpreted as an **int**, with value of 18

int *pa;



pa is an **int pointer**, it can hold the address of an **int**

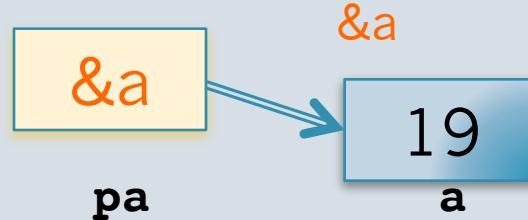
pa= &a;



pa now holds the address of **a**, or, it "points" to **a**.

***pa** is another method to access **a**

*pa= *pa + 1;



These statements are equivalent:

***pa** = ***pa** + 1;

a = **a** + 1;

***pa** = **a** + 1;

a = ***pa** + 1;

Pointers – application in function parameters

```
1 int n=10;
```

What sent to **printf** ?

```
2 printf("%d", n);
```

What sent to **scanf**?

```
3 scanf("%d", &n);
```

What **scanf** do to **&n**, to **n**?

```
4 void int_swap(int *a, int *b);  
5 swap(&n, &m);
```

What passed to **swap**?
Can this call make change to
&n or **&m**?

Can this call make change to
m or **n**?

example: pointers as function parameters

Pointers can be used to change the value of variables *indirectly*.
Example:
Function call in line 4 leads to the change of value of sum and product.

```
1 int main(...) {  
2     int a=2, b=4, sum, product;  
3     ...  
4     sAndP(a, b, &sum, &product);  
5     printf("sum=%d", sum);  
6     printf("prod=%d", product);  
7     ...  
8 }  
9  
10 void sAndP(int m, int n, int *ps, int *pp) {  
11     *ps = m + n;  
12     *pp = m * n;  
13 }  
14  
15 int_swap(&a, &b) ← int_swap(int *pa, int  
16 *pb)
```

&sum and &product are kept unchanged, the changes happen to **sum** and **product**

Lab Time: Individual Work, use Zoom/grok

	<p>Preparation:</p> <ul style="list-style-type: none">• open grok in a tab• open github.com/anhvir/c205/ws6.pdf in another tab
1	<p>Plan:</p> <p>Recursive Functions: When & How Scopes</p>
2	<p>15 min Group Work: ex. 6.2 (<i>discussion: scopes</i>) ex. 5.14 (<i>coding: log*</i>)</p>
3	<p>Pointers, Pointers as Function Arguments</p>
4	<p>Individual Work + Zoom + grok:</p> <ul style="list-style-type: none">• <i>try today's sample test if you haven't</i>• <i>ex. 6.5, 6.6 (coding: sorting 2, 3 integers)</i>• <i>try quizzes in ws6.pdf pp 18-21</i>• <i>ex. 6.9 (coding: coin change revisited)</i>
5	<p>Notes on the coming MST, walking through a sample test.</p>

Quiz 1

With the fragment:

```
int x= 10;  
f(&x);
```

which function below will set **x** to zero?

A:

```
int f(int n) {  
    return 0;  
}
```

C:

```
void f( int *n) {  
    n= 0;  
}
```

B:

```
void f( int *n) {  
    &n= 0;  
}
```

D:

```
void f( int *n) {  
    *n= 0;  
}
```

Quiz 2

Given function:

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

what are values of **m** and **n** after the following fragment:

```
m= 5;  
n= 10;  
f(m, &n);
```

A) 5 and 10

B) 1 and 2

C) 5 and 2

D) 1 and 10

Quiz 3

In executing the program:

```
int a=100, b=200;
void f(int a) {
    a++;
    print("1: a= %d b= %d\n", a, b) ;
}
int main(int argc, char *argv[ ]) {
    int a=5, b= 10;
    f(a);
    print("2: a= %d b= %d\n", a, b) ;
    return 0;
}
```

what will be printed out?:

A 1: a= 6 b= 200 2: a= 5 b= 10	B 1: a= 6 b= 200 2: a= 6 b= 10
C 1: a= 6 b= 10 2: a= 5 b= 10	D 1: a= 6 b= 10 2: a= 6 b= 10

Quiz 4

In executing the program:

```
#define N 3
int f(int);

int main(int argc, char *argv[ ]) {
    printf("%d \n", f(N));
    return 0;
}

int f(int n) {
    if (n <= 1) return 1;
    return n*n + f(n-1);
}
```

what will be printed out?:

A

14

B

1

C

6

D

9

Questions on the coming online test?

- Walk thru a sample test?
- Other questions/concerns?