

CPS 188

Computer Programming Fundamentals

Prof. Alex Ufkes

Topic 6.2: Pointers & examples

Notice!

Obligatory copyright notice in the age of digital delivery and online classrooms:

The copyright to this original work is held by Alex Ufkes. Students registered in course CPS 188 can use this material for the purposes of this course but no other use is permitted, and there can be no sale or transfer or use of the work for any other purpose without explicit permission of Alex Ufkes.

Today

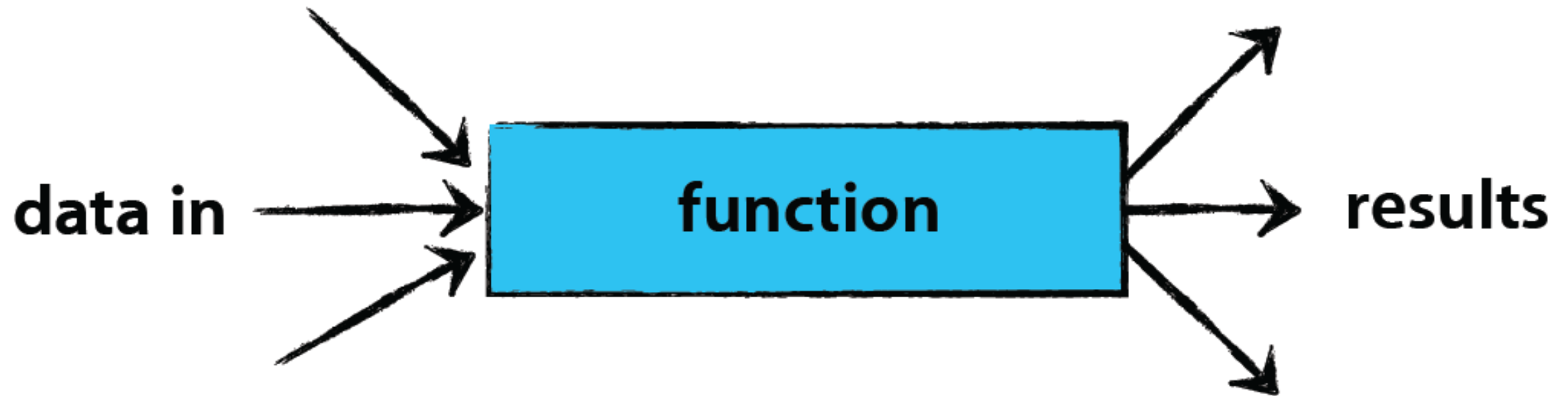
Assorted Examples



POINTERS

Multiple results?

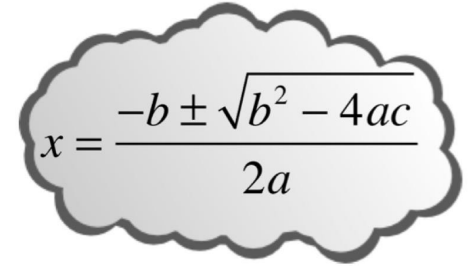
(WITHOUT using global variables!)





POINTERS
to the rescue!

quad()


$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

```
#include <stdio.h>
#include <math.h>

double quad (double a, double b, double c)
{
    double disc = sqrt(b*b - 4*a*c);
    double root1 = (-b + disc)/(2*a);
    double root2 = (-b - disc)/(2*a);
    return root1; // What about root2?
}

int main (void)
{
    double c1=1, c2=2, c3=3;
    double r1 = quad(c1, c2, c3);
    printf("root1 = %.2lf", r1);
    return 0;
}
```

"Can we have multiple results?"

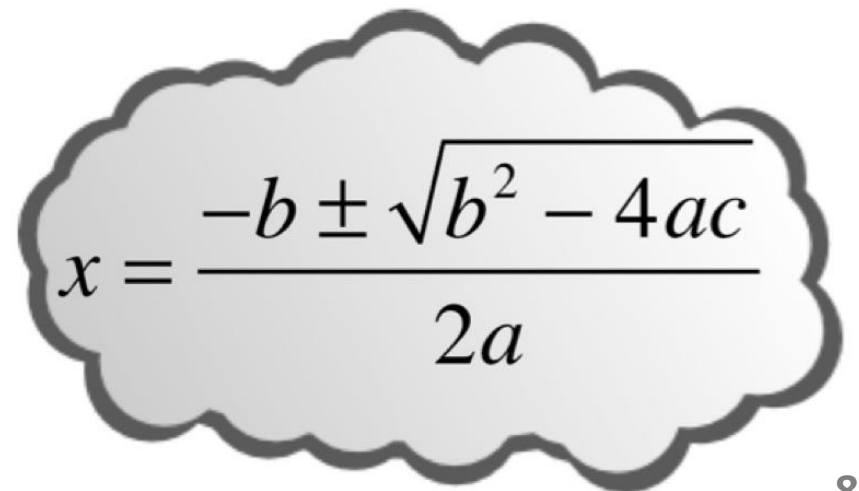
NO.

We're stuck!

- One option? Two separate functions, one for each root.
- Another option? Pointers, which we haven't learned yet.

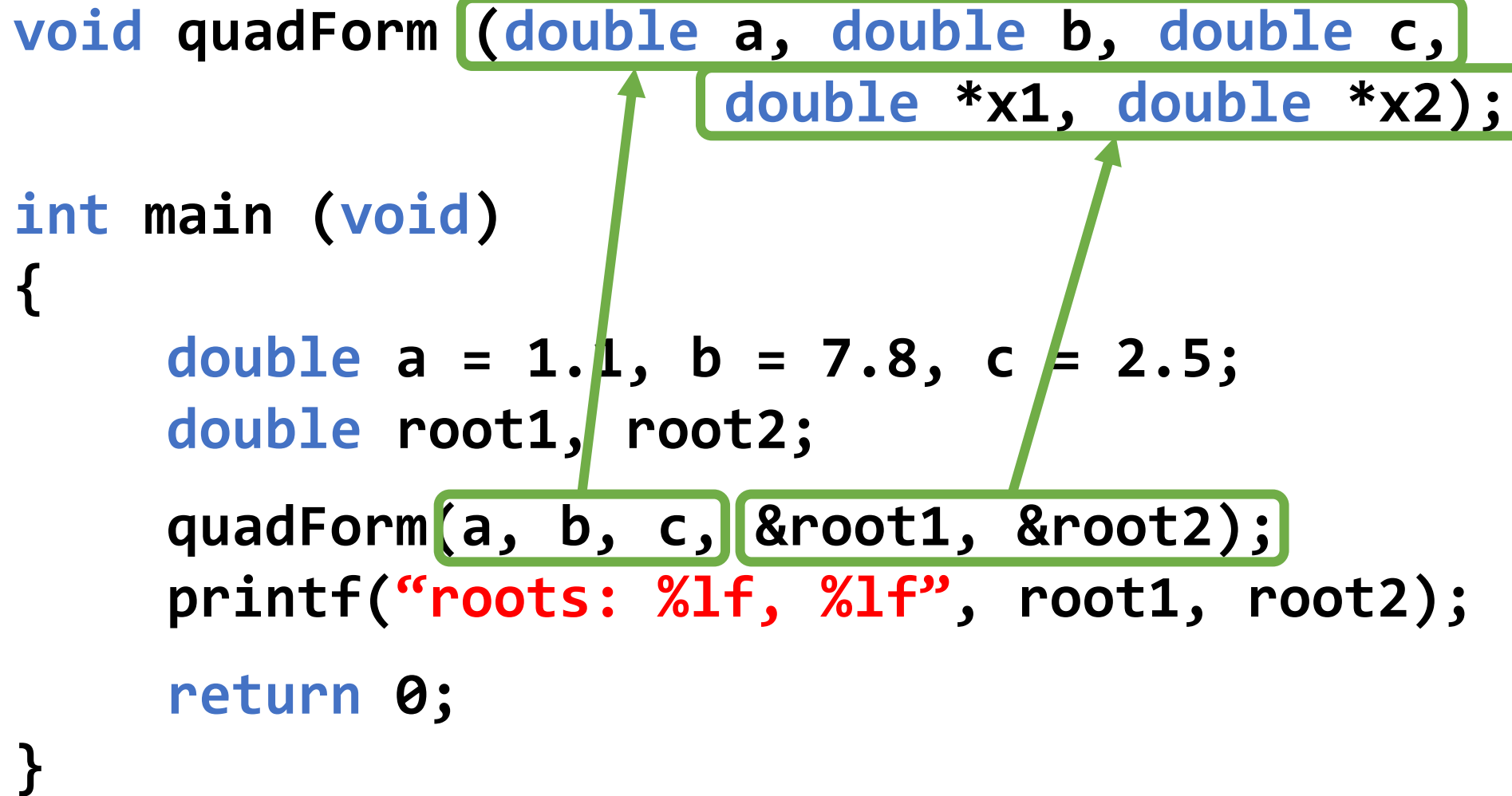
Quadratic Formula

```
void quadForm (double a, double b, double c,  
               double *x1, double *x2)  
{  
    double tmp = sqrt(b*b - 4*a*c);  
  
    *x1 = (-b + tmp)/(2*a);  
    *x2 = (-b - tmp)/(2*a);  
}
```


$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Quadratic Formula

```
void quadForm (double a, double b, double c,  
               double *x1, double *x2);  
  
int main (void)  
{  
    double a = 1.1, b = 7.8, c = 2.5;  
    double root1, root2;  
    quadForm(a, b, c, &root1, &root2);  
    printf("roots: %1f, %1f", root1, root2);  
    return 0;  
}
```



quad_form.c - C:\Users\aufke\Google Drive\Teaching\CPS 188\Code Samples - Geany

File Edit Search View Document Project Build Tools Help

Symbols quad_form.c

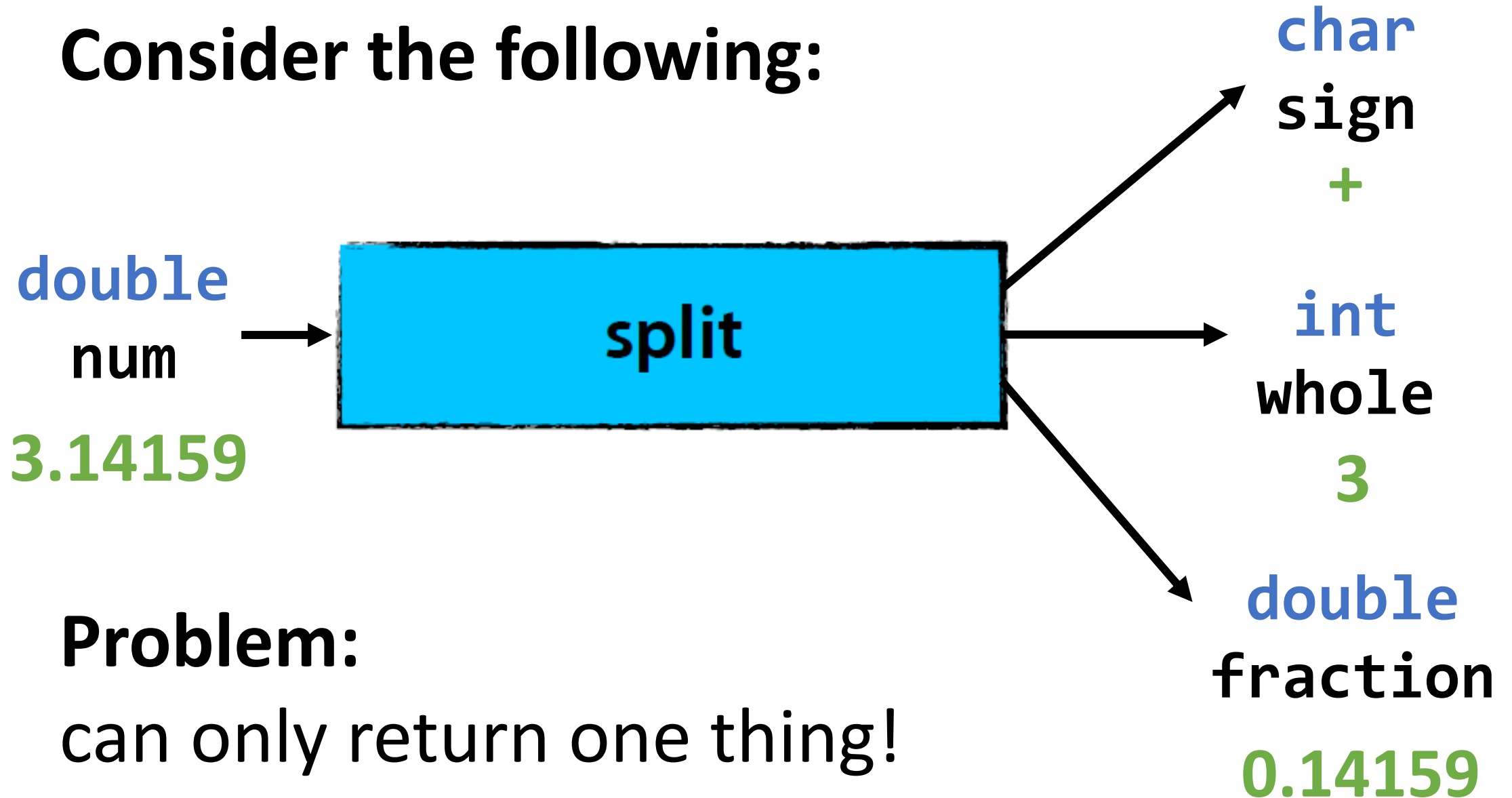
Functions
main [11]
quadForm [4]

```
1 #include <stdio.h>
2 #include <math.h>
3
4 void quadForm (double a, double b, double c, double *x1, double *x2)
5 {
6     double tmp = sqrt(b*b - 4*a*c);
7     *x1 = (-b + tmp)/(2*a);
8     *x2 = (-b - tmp)/(2*a);
9 }
10
11 int main (void)
12 {
13     double a = 1.1, b = 7.8, c = 2.0;
14     double root1, root2;
15
16     quadForm(a, b, c, &root1, &root2);
17     printf("roots: %lf, %lf", root1, root2);
18
19     return 0;
20 }
21
```

C:\WINDOWS\SYSTEM32\cmd.exe

```
roots: -0.336480, -6.754430
-----
(program exited with code: 0)
Press any key to continue . . .
```

Consider the following:



Pointers as
parameters!

`char`
`*sign`

`double`
`num`

`double`
`*fraction`

`split`

`int`
`whole`

Single return value:
`return whole;`

```
int split (double num, char *sign, double *fraction)
{
```

Pointers

```
    int whole = abs((int)num);
```

```
    *fraction = fabs(num) - whole;
```

```
    if (num >= 0)
```

```
        *sign = '+';
```

```
    else
```

```
        *sign = '-';
```

```
    return (whole);
```

De-reference pointers
to assign values

```
}
```

```
int split (double num, char *sign, double *fraction)
{ /*split code*/
}

int main (void)
{
    double f, n = 3.1415;
    char s;
    int w = split(n, &s, &f);
    printf("Sign:      %c\n", s);
    printf("Whole:      %d\n", w);
    printf("Fraction: %lf\n", f);

    return (0);
}
```

The diagram illustrates the relationship between variables in the `split` and `main` functions. An orange arrow points from the `int` parameter in the `split` function signature to the `int w` variable in the `main` function. Three green arrows point from the `n`, `&s`, and `&f` arguments in the `split` call to the `double num`, `char *sign`, and `double *fraction` parameters in the `split` function signature, respectively. The `n`, `&s`, and `&f` arguments are each enclosed in a green circle.

When we de-reference `sign` and `fraction` in `split`, we are accessing `s` and `f` in `main`!

```

int split (double num, double *fr, char *sign)
{
    int whole = abs((int)num);
    *fr = fabs(num) - whole;
    if (num >= 0) *sign = '+';
    else *sign = '-';
    return (whole);
}

int main (void)
{
    → double f; int w; char s;
    → w = split(3.14159, &f, &s);
    printf("%c %d %lf", s, w, f);
    return (0);
}

```

Memory

f		800
w		808
s		812
num	3.14159	912
fr	800	920
sign	812	924

```

int split (double num, double *fr, char *sign)
{
    → int whole = abs((int)num); /* 3 */
    → *fr = fabs(num) - whole; /* 0.14159 */
    {
        if (num >= 0) *sign = '+';
        else *sign = '-';
    }
    → return (whole);
}

int main (void)
{
    → double f; int w; char s;
    → w = split(3.14159, &f, &s);
    printf("%c %d %lf", s, w, f);
    return (0);
}

```

Memory

f	0.14159	800
w	3	808
s	'+'	812
num	3.14159	912
fr	800	920
sign	812	924
whole	3	928

Swap Two Variables

Very common operation!

```
void swap (int x, int y)
{
    x = y;
    y = x;
}
```

Is there a problem here?

Swap Two Variables

Very common operation!

```
void swap (int x, int y)
{
    int tmp = x;
    x = y;
    y = tmp;
}
```

How about now?

Swap Two Variables

```
void swap (int x, int y)
{
    int tmp = x;
    x = y;
    y = tmp;
}

int main (void)
{
    int a = 1, b = 2;
    printf("%d, %d\n", a, b);
    swap(a, b);
    printf("%d, %d\n", a, b);
    return 0;
}
```

What prints?

swap.c - C:\Users\aufke\Google Drive\Teaching\CPS 188\Code Samples - Geany

File Edit Search View Document Project Build Tools Help

+

▼

📁

▼

📄

📄

🔄

✖

⬅

➡

🔍

🧩

🔴

🖱

🔍

🔍

💡

🔴

Symbols

swap.c

Functions

- main [10]
- swap [3]

1#include <stdio.h>

2

3void swap (int x, int y)

4{

5int tmp = x;

6x = y;

7y = tmp;

8}

9

10int main (void)

11{

12int a = 1, b = 2;

13printf("%d, %d\n", a, b);

14swap(a, b);

15printf("%d, %d\n", a, b);

16return 0;

17}

18

19

C:\WINDOWS\SYSTEM32\cmd.exe

1, 2

1, 2

(program exited with code: 0)

Press any key to continue . . .

Swap Two Variables

```
void swap (int *x, int *y)
{
    int tmp = *x;
    *x = *y;
    *y = tmp;
}

int main (void)
{
    int a = 1, b = 2;
    printf("%d, %d\n", a, b);
    swap(&a, &b);
    printf("%d, %d\n", a, b);
    return 0;
}
```

What prints?



Symbols

swap.c

Functions
main [10]
swap [3]

```
1  #include <stdio.h>
2
3  void swap (int *x, int *y)
4  {
5      int tmp = *x;
6      *x = *y;
7      *y = tmp;
8  }
9
10 int main (void)
11 {
12     int a = 1, b = 2;
13     printf("%d, %d\n", a, b);
14     swap(&a, &b);
15     printf("%d, %d\n", a, b);
16     return 0;
17 }
18
19
```

C:\WINDOWS\SYSTEM32\cmd.exe

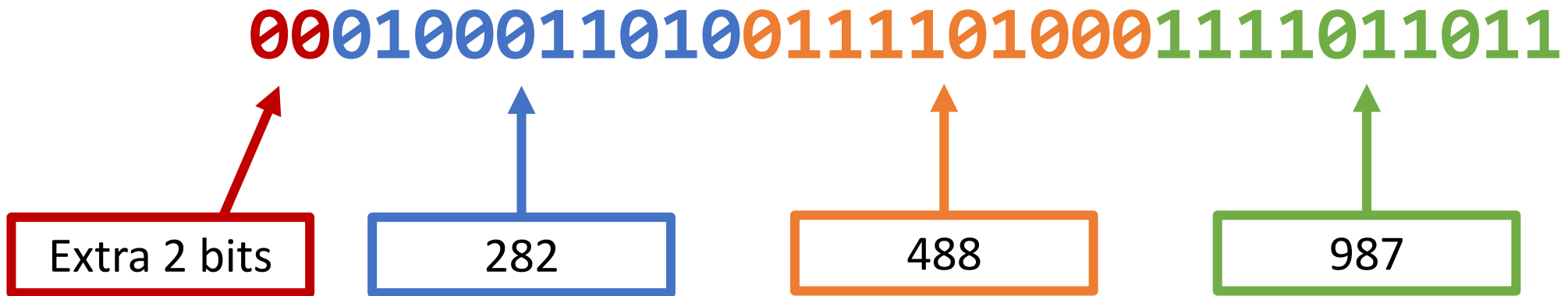
```
1, 2
2, 1
```

```
-----
(program exited with code: 0)
Press any key to continue . . .
```

Recall: Integer Packing

Shift 10 bits at a time:

- Consider three numbers: 282, 488, 987
- We want to pack them into a single `int` like so:






```
1  #include <stdio.h>
2
3  int pack3 (int a, int b, int c)
4  {
5      int packed = a << 10;
6      packed = (packed | b) << 10;
7      return packed | c;
8  }
9
10 void unpack3 (int packed)
11 {
12     printf("%d\n", packed & 0x3FF);
13     printf("%d\n", (packed >> 10) & 0x3FF);
14     printf("%d\n", (packed >> 20) & 0x3FF);
15 }
16
17 int main (void)
18 {
19     int packed = pack3(282, 488, 987);
20     printf("packed = %d\n", packed);
21     unpack3(packed);
22     return 0;
23 }
24
```

- Instead of printing in the function (usually bad)...
- Let's use pointers to send back all three values.


```
void unpack3 (int packed)
{
    printf("%d\n", packed & 0x3FF);
    printf("%d\n", (packed >> 10) & 0x3FF);
    printf("%d\n", (packed >> 20) & 0x3FF);
}
```



```
void unpack3 (int packed, int *a, int *b, int* c)
{
    *a = packed & 0x3FF;
    *b = (packed >> 10) & 0x3FF;
    *c = (packed >> 20) & 0x3FF;
}
```

```

swap.c x intpack.c x
1  #include <stdio.h>
2
3  int pack3 (int a, int b, int c)
4  {
5      int packed = a << 10;
6      packed = (packed | b) << 10;
7      return packed | c;
8  }
9
10 void unpack3 (int packed, int *a, int *b, int* c)
11 {
12     *a = packed & 0x3FF;
13     *b = (packed >> 10) & 0x3FF;
14     *c = (packed >> 20) & 0x3FF;
15 }
16
17 int main (void)
18 {
19     int n1, n2, n3;
20     int packed = pack3(282, 488, 987);
21     printf("packed    = %d\n", packed);
22     unpack3(packed, &n1, &n2, &n3);
23     printf("unpacked = %d, %d, %d\n", n1, n2, n3);
24     return 0;
25 }
26

```

```

C:\WINDOWS\SYSTEM32\cmd.exe
packed    = 296199131
unpacked = 987, 488, 282

-----
(program exited with code: 0)

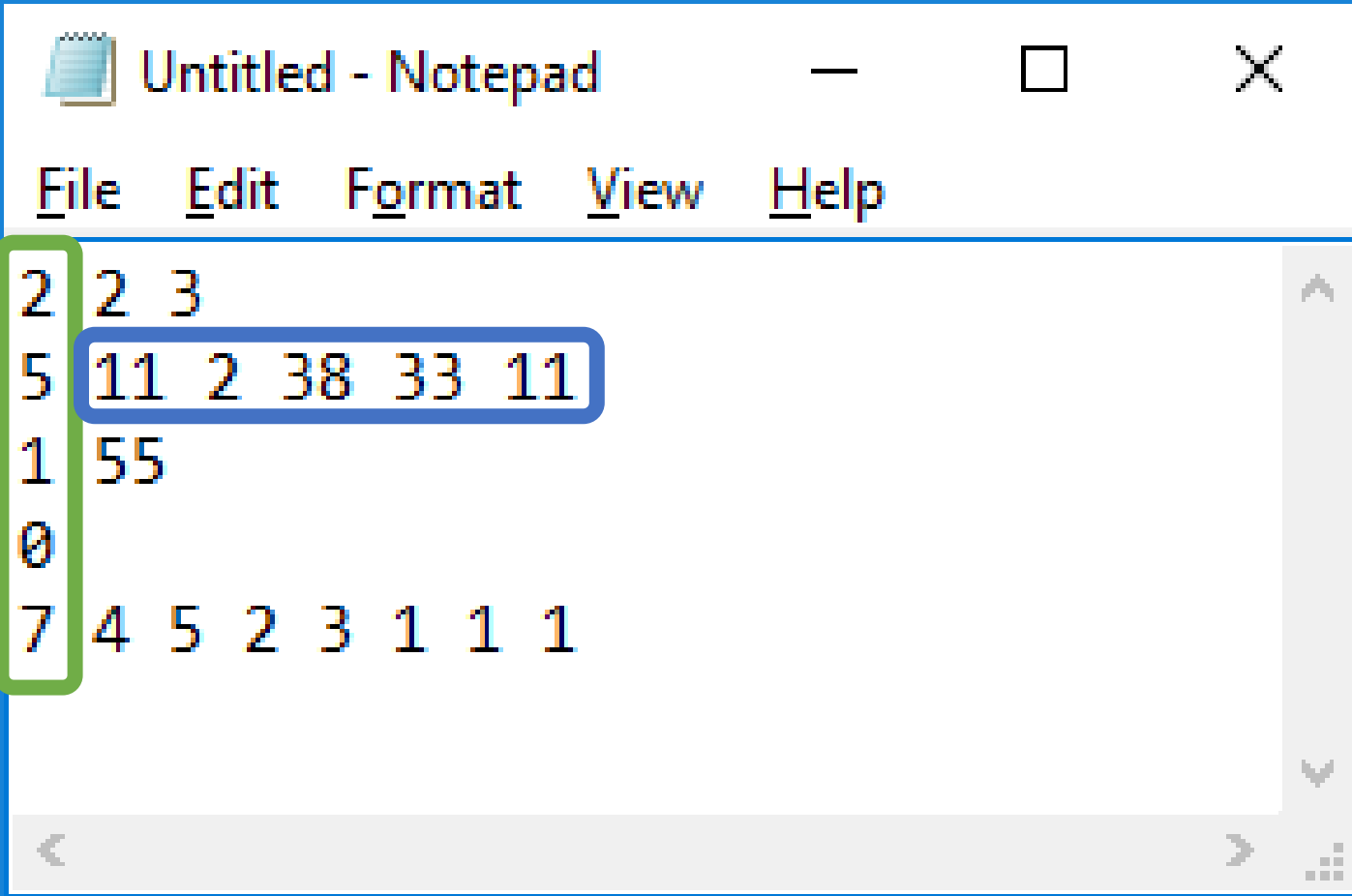
Press any key to continue . . .

```

Nested Loops + File I/O

Write a code snippet to do the following:

Compute the averages of the integers in each row in a file. The first number in each row indicates the number of integers on that row.



```
2 2 3
5 11 2 38 33 11
1 55
0
7 4 5 2 3 1 1 1
```

Write a code snippet to do the following:

Compute the averages of the integers in each row in a file. The first number in each row indicates the number of integers on that row.

```
int i, value, sum;
```

```
int num_in_row;
```

```
double avg;
```

```
FILE *in = fopen("data.txt", "r");
```

Read number of integers per row

```
while (fscanf(in, "%d", &num_in_row) != EOF)
```

```
{
```

```
    sum = 0;
```

Initialize sum for current row

```
    for (i = 0; i < num_in_row; i++) {
```

```
        fscanf(in, "%d", &value);
```

```
        sum += value;
```

```
    }
```

Read all integers on current row

```
    avg = (double)sum/num_in_row;
```

```
    printf("Row average = %6.2lf \n", avg);
```

Compute and print average

```
}
```

Repeat until EOF

**Did your file open
correctly?**

`fopen` isn't guaranteed to work

If your program involving file I/O *compiles* but doesn't work as expected, your file may not have opened successfully:

```
FILE *in = fopen("data.txt", "r");  
if (in == NULL) /* File didn't open correctly */  
    printf("File failed to open!\n");
```

Double check file name, location, etc.

* Not available at the Broome, Cairns Central, Castletown, Earlville, Mackay, Maroochydoore, Palmerston, Rockhampton or The Willows stores.

- A. Boys'. Piping Hot fleece hoodie, 7-16. Fleece trackpants, 7-16. **Save \$6 Now \$12**
- B. Women's. Fleece top, 10-18. Straight leg fleece pants, 10-18. **Save \$9 Now \$27**
- C. Men's. * Fleece jumper. * Trackpants, X. **Save \$6 Now \$12**
- D. Girls'. Top, 7-16. Trackpants, 7-16. **Save \$5 Now \$14**

Spot the **ERROR?**

Top
\$27
Save \$9

Hoodie
\$20
Save \$9

Jumper
\$27
Save \$12

Top
\$14
Save \$6



Fabulous fleece!
Fleece is a great fabric to wear; it's soft and cuddly, lightweight and easy to take care of.




```
#include <stdio.h>
```

```
int main (void)
```

```
{
```

```
    int x, y = 2;
```

```
    int* w, *z;
```

```
    *w = 1;
```

```
    *x = 125;
```

Pointer **not** initialized

```
    p
```

Primitives cannot be dereferenced

```
    return 3-2;
```

```
}
```



```
int main (void)
{
    int x, y, z;
    int *p1, *p2, p3*; char * p4;

    p1 = &x;
    x = 125;
    p2 = p1;
    p4 = p2;

    return 125;
}
```

Incompatible pointer types



```
#include <stdio.h>
```

```
int main(void)
```

```
{
```

```
    int n, x = -3;
```

```
    for (n = x, n <= 0, n += 1)
```

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

```
    return 0;
```

```
}
```

Should be semicolons, not commas



```
#include <stdio.h>
```

```
int main(void)
```

```
{
```

```
    int n;
```

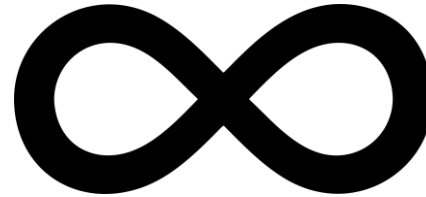
```
    for (n = 1; n != 50; n *= 2) {
```

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

```
    }
```

```
    return 0;
```

```
}
```



```
#include <stdio.h>
```

```
int main (void)
```

```
{
```

```
    int x = 5;
```

```
    while( x > 0 );
```

```
        x--;
```

```
    return 55;
```

```
}
```



```
#include <stdio.h>
```

```
int main(void)  
{
```

Variable not declared!

```
  for (i = 1; i <= 12; i++)  
  {  
    printf("%d\n", i);  
  }
```

```
  return (0);
```

```
}
```



```
#include <stdio.h>
```

```
int main(void)
```

```
{
```

```
    int n;
```

```
    do
```

```
    {
```

```
        printf("Enter a number ");
```

```
        printf("between 1 and 9: ");
```

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

```
    }while (n < 1 || n > 9)
```

```
    return (0);
```

```
}
```



Questions?

