CMPE-013/L

Introduction to "C" Programming

Max Lichtenstein

Piazza Poll: Which function(s) should use a stack?

A) **int IsPalindrome(char * string);**

```
if(IsPalindrome("racecar")) {
    printf("racecar is a palindrome!");}
```

FIFO + FILO

B) **int IsPalindrome(char * string, int string_length);**

```
if(IsPalindrome("racecar", 7)) {
    printf("racecar is a palindrome!");}
```

C) **int PrintTwice(char * string);**

```
PrintTwice("Hello"); //prints "HelloHello"
```

HHeelloo

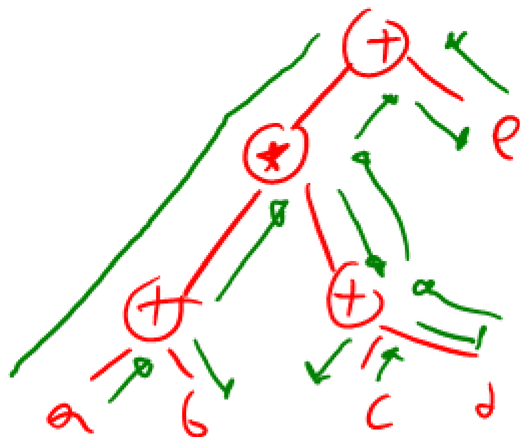
Helloolleh

D) **int FindMax(int * list_of_numbers);**

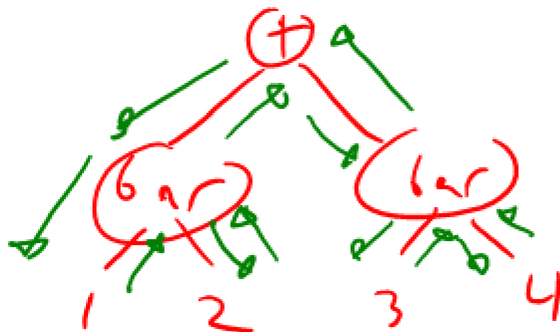
```
X = FindMax({8, 6, 7, 5, 3, 0, 9}); // X = 9
```

order doesn't matter

$(a + b) * (c + d) + e$



```
foo()
   return bar()+bar()

bar()
   return i++;

main(){
  foo();
```



9
0
3
5
7
6
8

9 0 3 5 7 6 8

int currentMax = first element

if next_element > current element

current max = next element

Palindrome(" ")

r

a

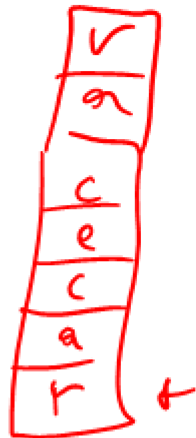
c

e

c

a

r



Palindrome(" ", len)

e d
l
a
r

Roadmap

- Lab 4 stuff
- Stack review
- Structs
  - What they are
  - definining, initializing, and using structs
  - 2 different struct notations and when you should ~~[illegible]~~ use which
- Break
- Typedefs
- The Heap and allocation (?)

Lab 4 Announcements

- Modified documentation slightly:
  - rpn.h - added newline clarification
  - stack.h - clarified spec, changed to doxygen notation
  - Lab manual - removed typo line about not using stacks in rpn.c (you definitely use stacks in rpn.c!)
- Lab04 autochecker runs
  - Still rudimentary, but hey

Lab 4 Tips

- Struct notation – be careful, do reading
- Start with stack.c/stack_test.c,
- Then do rpn.c/stack_test.c,
- Then do main()

Stack Review

Stacks

Stack Operations

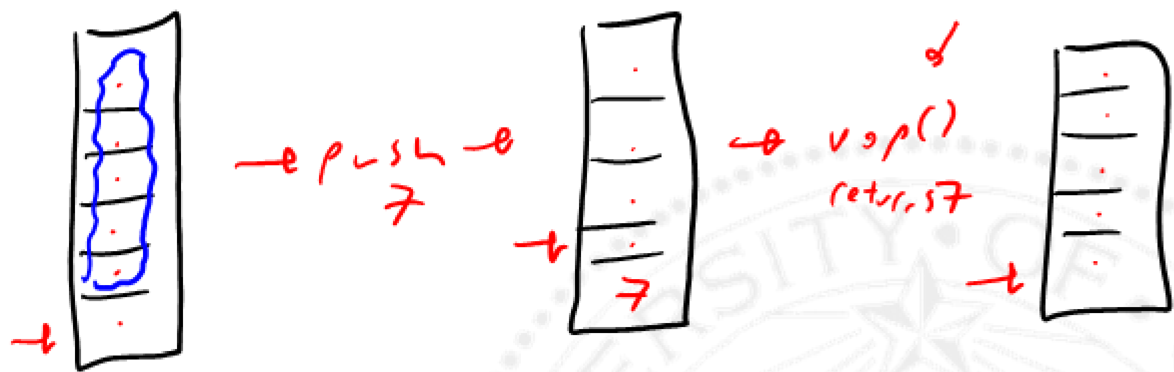
Minimum, 2 operations:

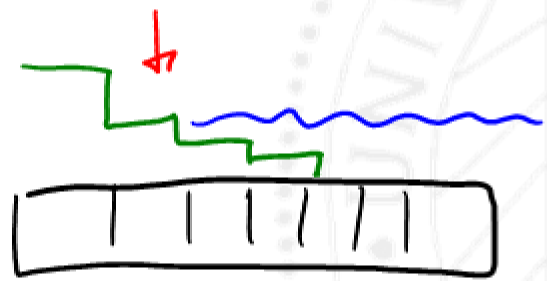
- Push()   push(x)
- Pop()   y = pop()

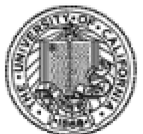
Also maybe:

- Initialize()
- Peek()

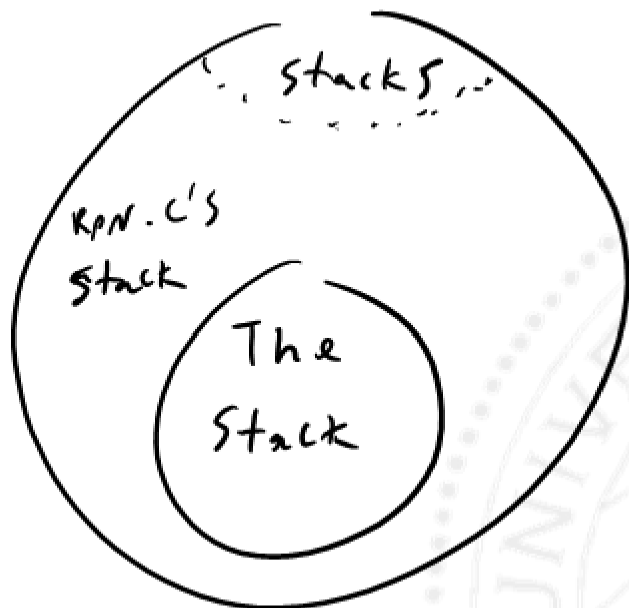
Stacks in memory







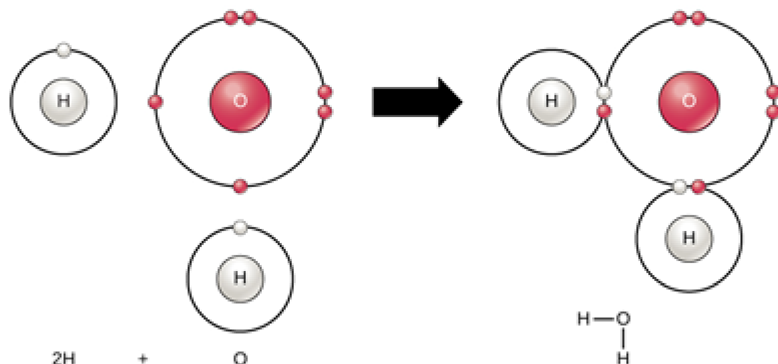
stacks and The Stack



Structs

A collection of fundamental data types

int
char
float
double



struct water {
hydrogen,
hydrogen,
oxygen
}

For example:

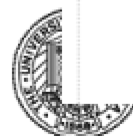
```
struct President {
    int number;
    char first_name[20];
    char last_name[20];
    int start_year;
    int end_year;
};

void main(void){

    struct President obama = {44, "Barack", "Obama", 2009, 2017};
    struct President bush = {43, "George", "Bush"};

    bush.number = 43;
    bush.start_year = 2001;
    bush.end_year = 2009;

    printf("President %s %s:\n", obama.first_name, obama.last_name);
    printf("   Served from %d to %d.\n", obama.start_year, obama.end_year);
```

member

declaration

instantiation

access



Why make a struct?

- Group concepts together for readability

- Pass lots of information into, out of functions

struct bar foo (struct bar);

- Iterate over sets of consistently-structured data

- *objectify*

Structs in Memory

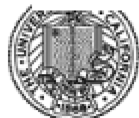
Structs: Values vs Pointers

```
void PrintPresidentByValue(struct President pres){
    printf("President %s %s:\n", pres.first_name, pres.last_name);
    printf("    Served from %d to %d.\n", pres.start_year, pres.end_year);
    pres.end_year = 2021;
}

void PrintPresidentByReference(struct President * pres){
    printf("President %s %s:\n", pres->first_name, pres->last_name);
    printf("    Served from %d to %d.\n", pres->start_year, pres->end_year);
    pres->end_year = 2021;
}
```

*(pres.start_year)

Question: Who serves until 2021?

```
struct President obama = {44, "Barack", "Obama", 2009, 2017};
struct President trump = {45, "Donald", "Trump", 2017, 2025};

PrintPresidentByValue(obama);
PrintPresidentByReference(&trump);

struct President president_list[2] = {obama, trump};

for(i = 0; i<2; i++){
    struct President this_pres = president_list[i];
    if(this_pres.end_year == 2021) {
        printf("%s serves until 2021\n", this_pres.last_name);
    }
}
```



Structs of Structs

5′ end
phosphate
deoxyribose
adenine
guanine
purine bases
cytosine
thymine (in DNA)
pyrimidine bases
3′ end

ribose (in RNA)
uracil (in RNA)

NH_2
CH_3

© 2015 Encyclopædia Britannica, Inc.

Structs of Structs

```c
struct Bicycle {
    char * model;
    unsigned int year;
};

struct Human {
    char * name;
    unsigned int age;
};

struct Bus {
    struct Human driver;
    struct Human passengers[30];
    struct Bicycle bike_rack[3];
};
```

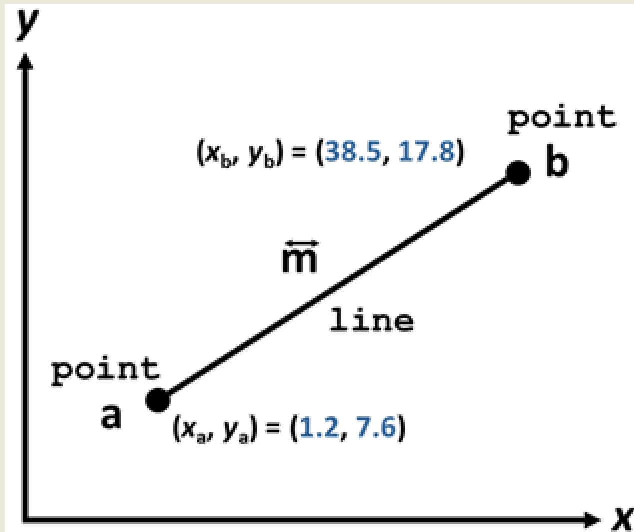
```c
void main(void){

    struct Bus myBus = {
        16,  //line number
        {"Joe The Bus Driver",31}, //bus driver (name, age)
        {{"Max", 31}, {"Pavlo",23}}, //a couple passengers
        {{"GT","1990"}} //Max's bike
    };
```

Structures

Nesting Structures

Example

```
struct point {
  float x;
  float y;
};

struct line {
  struct point a;
  struct point b;
};
```




```
int main(void)
{
  struct line m = {{1.2, 7.6}, {38.5, 17.8}};
  ...
}
```

Structures

Instantiation upon Declaration

Syntax

```
struct StructName {
    type1 memberName1;
    ...
    typen memberNamen;
} varName1,...,varNamen;
```

Example

```
// Structure to handle complex numbers
struct Complex {
  float re;
  float im;
} x, y;        // Declare x and y of type complex
```

Anonymous Structs

- Don't need type names!

```
struct{
    char IsTicking;
    char IsDone;
    int TimeLeft;
} global_timer;
```

Typedef

typedef in C

"Alias"

- Just a new name for an existing datatype

```
typedef old_thing new_name;
```

old_type

- So there are never *really* any new datatypes

typedef example

```
typedef unsigned int year;

void main(void){

    year AmericasBirthYear = 1789;
```

Why typedef?

- ☐ Conceptual clarity
- ☐ Readability
  - new way to communicate about data
- ☐ Type safing

```
year HowLongAgo (year past_year,
                 year this_year);

year 24; ✗
```

Typedef anonymous structs

```
typedef enum{
    CAT, DOG, PARROT, GERBIL
} species;

typedef struct {
    char * name;
    species species;
} pet1;

struct pet2{
    char * name;
    species species;
};
```

```
void main(void){


    struct pet2 myDog = {"River", DOG};
    pet1 myParentsCat = {"Fargo", CAT};
```

CAT==pet1.species

A couple more struct things

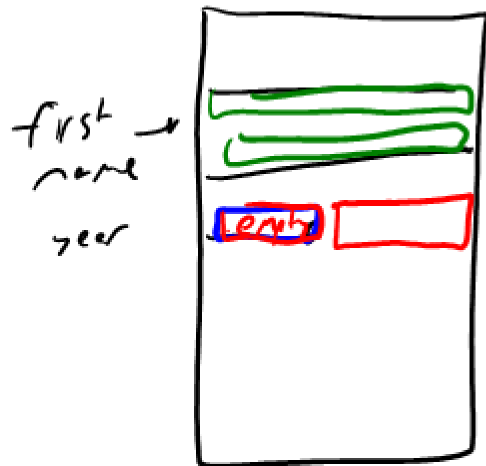
- Structs are *copied* on assignment
  - Also copied on function return and pass-by-value
  - If your struct is big this can be slow!

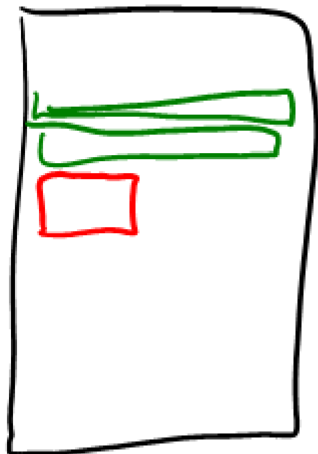
- Structs can contain pointers to structs of the same type
  - or to themselves!
  - Be careful with reference/dereference operator

- They have a close cousin, the union

Struct

Union





Structs recap

- Implicit new type
- A way to group data conceptually
- Useful way to pass lots of information around
- Can be instantiated on declaration
- Can be anonymous (typeless)

10. Which of the four snippets below both creates a new type, `struct Student`, AND declares a variable named `thisStudent`?

```
//A:
typedef struct Student{
    char firstname[100];
    char lastname[100];
} thisStudent;
```

```
//C:
struct Student{
    char firstname[100];
    char lastname[100];
} thisStudent;
```

```
//B:
struct Student{
    char firstname[100];
    char lastname[100];
};
thisStudent = Student;
```

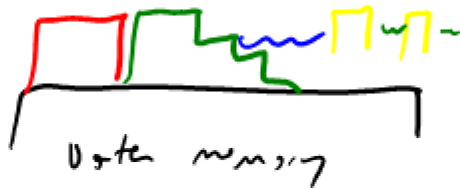
```
//D:
typedef struct{
    char firstname[100];
    char lastname[100];
} Student;
Student thisStudent;
```

The Heap and Allocation

static stack

Heap

Prog memory

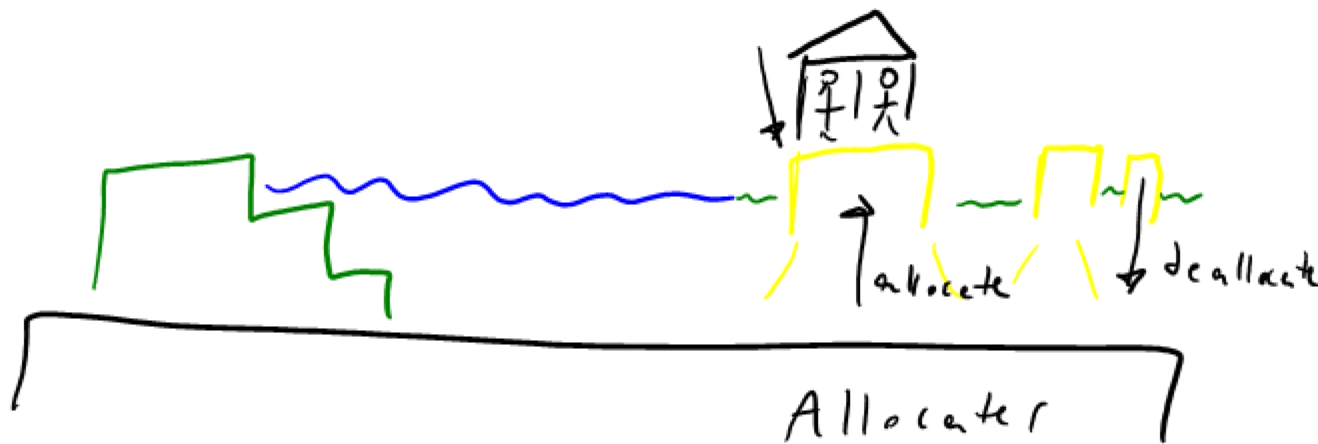


void* malloc(island size);

Heap

dealloc(void* memory);

ints
fuels



```
stack_pop(stack*){
    return stack->top.item;
}
```

&