UPR: Structured Data Types (`struct`)

Jan Gaura

https://github.com/geordi/upr-course

Outline

- **Administrative**
- `struct`
- Final notes

What did I do?

Administrative

On SUN's years:

Kicked butt, had fun, didn't cheat,
loved our customers, changed computing forever.

Scott McNealy, SUN Microsystems

Administrative

On SUN's years:

Kicked butt, had fun, didn't cheat,
loved our customers, changed computing forever.

Scott McNealy, SUN Microsystems

Q & A Session

- Last week, we made a Q & A session for you

Q & A Session

- Last week, we made a Q & A session for you
  - three (3) people came

Q & A Session #2

- So...

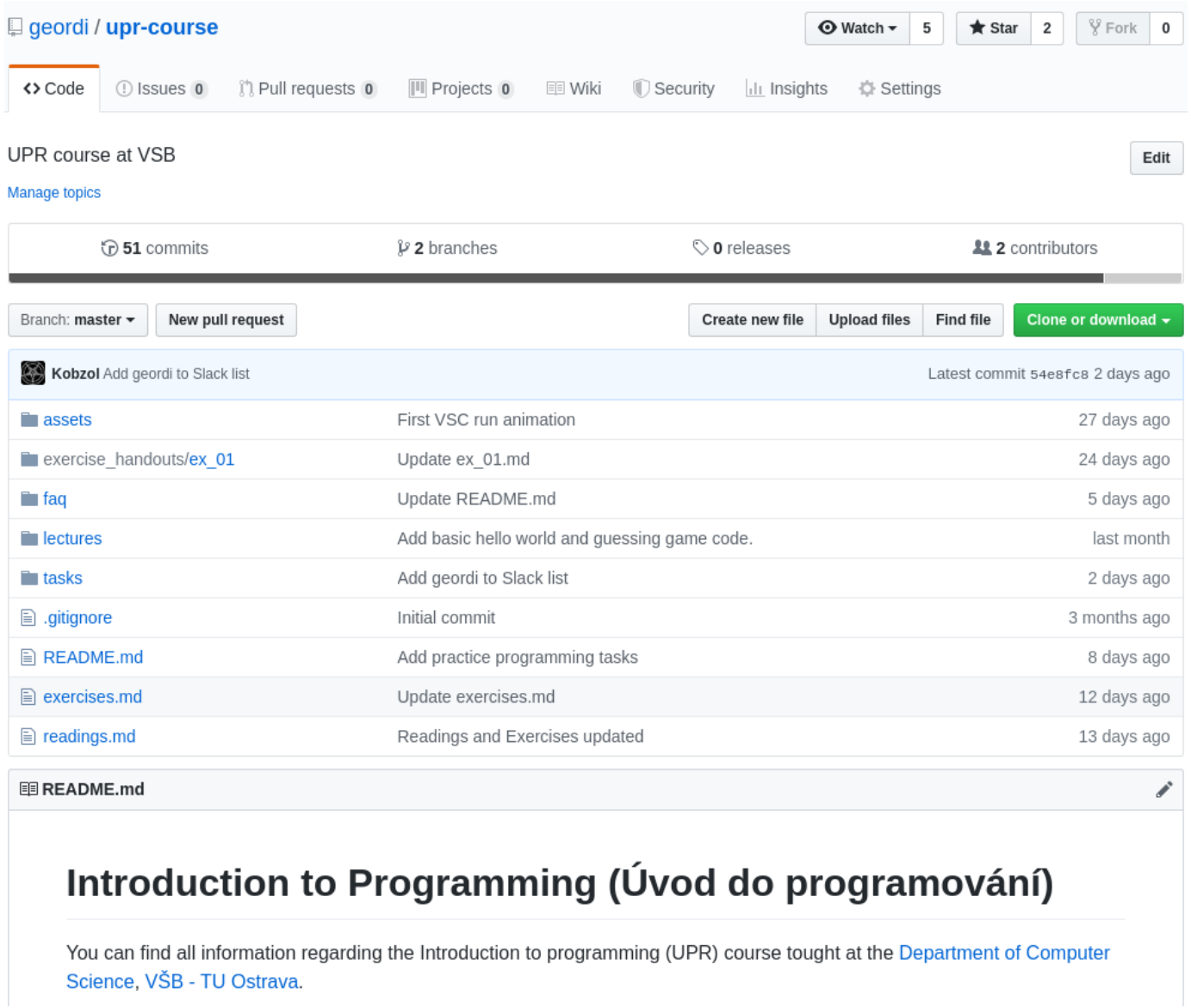
Q & A Session #2

- So...
- ... we'll hold another one NEXT week again

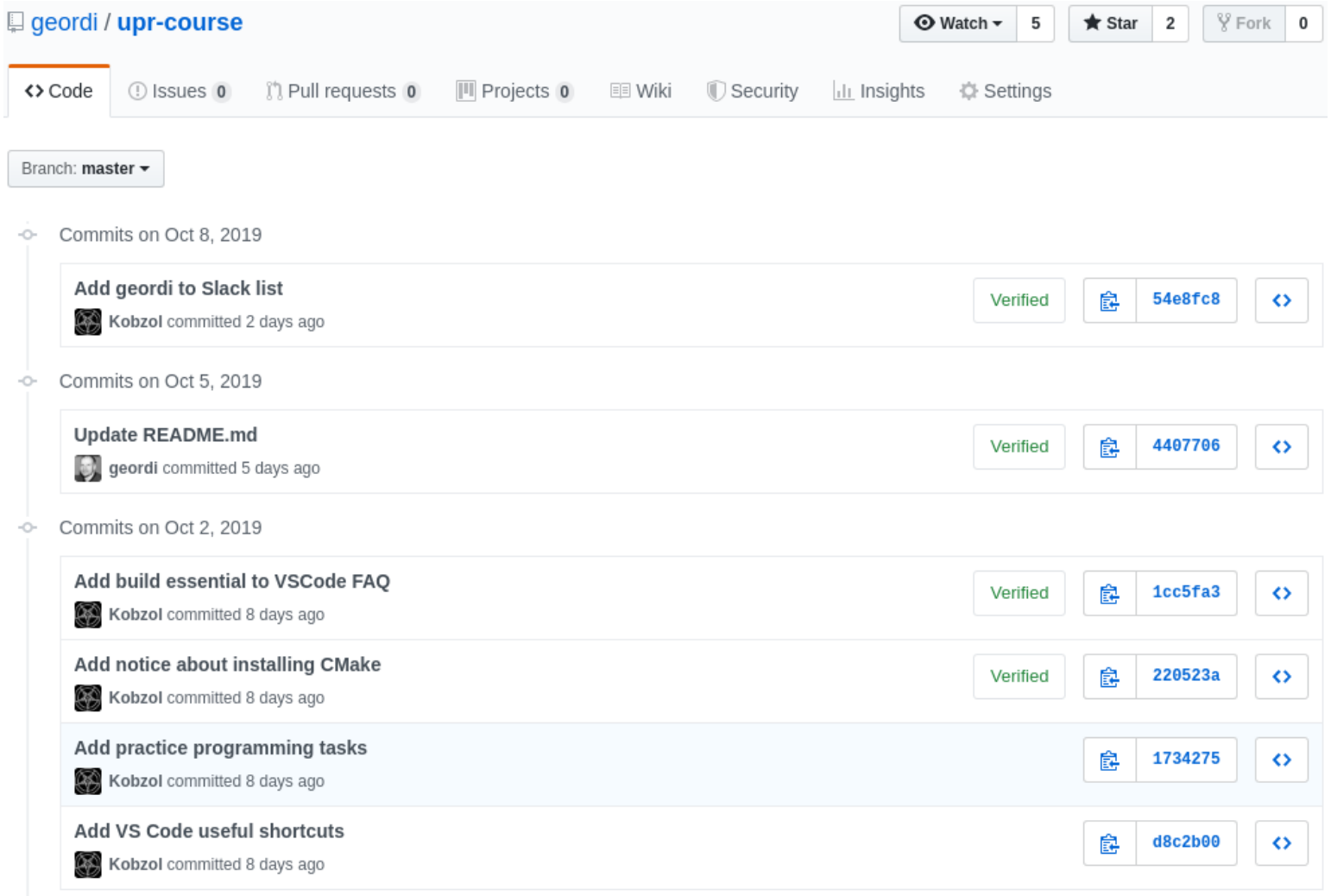
Q & A Session #2

- So...
- ... we'll hold another one NEXT week again
- Tuesday 22. 10. 2019 in EB213

GitHub - The Course's Page



GitHub - Easy to Check What's Going On



Outline

- Administrative
- **`struct`**
- Final notes

Structured Data Types - A Motivation

- Working with all arrays and lengths of arrays separately...

Structured Data Types - A Motivation

- Working with all arrays and lengths of arrays separately...
- ... can be a mess

Structured Data Types - A Motivation

- Working with all arrays and lengths of arrays separately...
- ... can be a mess

```
01 int stats( int * arr, int arr_len ) {
02     int max = arr[ 0 ];
03
04     for ( int i = 1; i < arr_len, i++ ) {
05         if ( arr[ i ] > max ) {
06             max = arr[ i ];
07         }
08     }
09
10     return max;
11 }
```

Structured Data Types - A Motivation

- Working with all arrays and lengths of arrays separately...
- ... can be a mess

```
01 int stats( int * arr, int arr_len ) {
02     int max = arr[ 0 ];                    ← May result in Segmentation Fault
03
04     for ( int i = 1; i < arr_len, i++ ) {
05         if ( arr[ i ] > max ) {
06             max = arr[ i ];
07         }
08     }
09
10     return max;
11 }
```

Structured Data Types - A Motivation

- Working with all arrays and lengths of arrays separately...
- ... can be a mess

```c
int stats( int * arr, int arr_len ) {
    int max = arr[ 0 ];

    for ( int i = 1; i < arr_len, i++ ) {
        if ( arr[ i ] > max ) {
            max = arr[ i ];
        }
    }

    return max;
}
```

← May result in Segmentation Fault

```c
void stats( int * arr1, int arr1_len, int * arr2, int arr2_len, int *max1, int *max2 ) {
    *max1 = arr1[ 0 ];
    *max2 = arr2[ 0 ];

    for ( int i = 0; i < arr1_len, i++ ) {
        ...

    for ( int i = 0; i < arr2_len, i++ ) {
        ...

}
```

Structured Data Types - A Motivation

- What if...

Structured Data Types - A Motivation

- What if an array would also hold it's length?

Structured Data Types - A Motivation

- What if an array would also hold it's length?

```
01 int stats( int * arr, int arr_len ) {
02     int max = arr[ 0 ];
03 
04     for ( int i = 1; i < arr_len, i++ ) {
05         if ( arr[ i ] > max ) {
06             max = arr[ i ];
07         }
08     }
09 
10     return max;
11 }
```

Structured Data Types - A Motivation

- What if an array would also hold it's length?

```
01 int stats( int * arr, int arr_len ) {
02     int max = arr[ 0 ];
03
04     for ( int i = 1; i < arr_len, i++ ) {
05         if ( arr[ i ] > max ) {
06             max = arr[ i ];
07         }
08     }
09
10     return max;
11 }
```

```
01 int stats( Array * arr ) {
02     int max = arr.items[ 0 ];
03
04     for ( int i = 1; i < arr.len, i++ ) {
05         if ( arr.items[ i ] > max ) {
06             max = arr.items[ i ];
07         }
08     }
09
10     return max;
11 }
```

Is there any solution?

Sure :-)

The aswer is: struct

Struct Defintion

- To concentrate things together...

Struct Defintion

- To concentrate things together...
- We use structures (`struct`) in C language

Struct Defintion

- To concentrate things together...
- We use structures (`struct`) in C language
- How can we define a `struct`?

Struct Defintion

- To concentrate things together...
- We use structures (`struct`) in C language
- How can we define a `struct`?

```
struct Array {
    int * items;
    int len;
};
```

Struct Defintion

Struct Defintion

```
struct Array {
    int * items;
    int len;
};
```

Struct Defintion

```c
struct Array {
    int * items;
    int len;
};
```

```c
int main() {
    struct Array * a = array_new( 10 );

    printf( "Array len= %d\n", a->len );

    return 0;
}
```

Struct Defintion

```
struct Array {
    int * items;
    int len;
};
```

```
int main() {
    struct Array * a = array_new( 10 );

    printf( "Array len= %d\n", a->len );

    return 0;
}
```

```
struct Array * array_new( const int len ) {
    struct Array * a = NULL;

    a = (struct Array *)malloc( sizeof( struct Array ) );
    // memory check

    a->items = (int *)malloc ( len * sizeof( int ) );
    // memory check
    a->len = len;

    return a;
}
```

Struct Usage Simplification

- Ohhh, if I could only get rid of the `struct` keyword here and there

Struct Usage Simplification

- Ohhh, if I could only get rid of the `struct` keyword here and there
- Well... you can

Struct Usage Simplification

- Ohhh, if I could only get rid of the `struct` keyword here and there
- Well... you can

```
1 struct Array {
2     int * items;
3     int len;
4 };
5
6 typedef struct Array Array;
```

Struct Usage Simplification

- Ohhh, if I could only get rid of the `struct` keyword here and there
- Well... you can

```
1 struct Array {
2     int * items;
3     int len;
4 };
5
6 typedef struct Array Array;
```

Struct Usage Simplification

- Ohhh, if I could only get rid of the `struct` keyword here and there
- Well... you can

```
1 struct Array {
2     int * items;
3     int len;
4 };
5
6 typedef struct Array Array;
```

```
int main() {
    Array * a = array_new( 10 );

    printf( "Array len= %d\n", a->len );

    return 0;
}
```

Struct Usage Simplification

- Ohhh, if I could only get rid of the `struct` keyword here and there
- Well... you can

```
1 struct Array {
2     int * items;
3     int len;
4 };
5
6 typedef struct Array Array;
```

```
int main() {
    Array * a = array_new( 10 );

    printf( "Array len= %d\n", a->len );

    return 0;
}
```

```
Array * array_new( const int len ) {
    Array * a = NULL;

    a = (Array *)malloc( sizeof( Array ) );
    // memory check

    a->items = (int *)malloc ( len * sizeof( int ) );
    // memory check
    a->len = len;

    return a;
}
```