My favorite programming languages

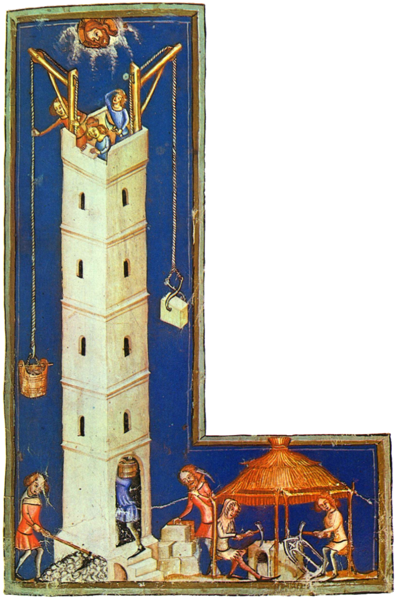
and three others

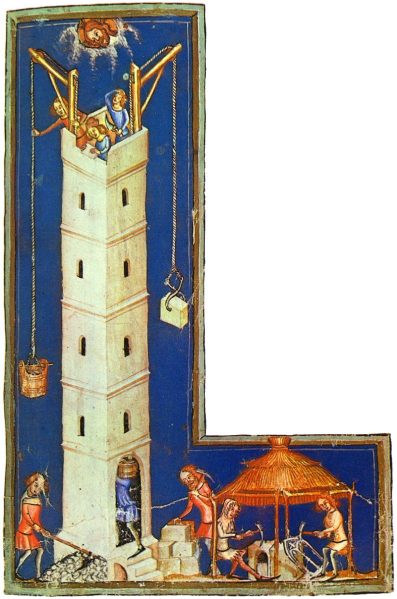
Douglas Creager

@dcreager

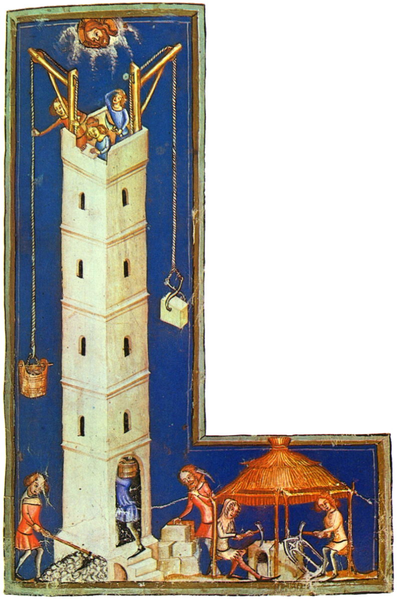
Craft Conf

June 2, 2022 – Budapest





COBOL



APL C# PERL TS

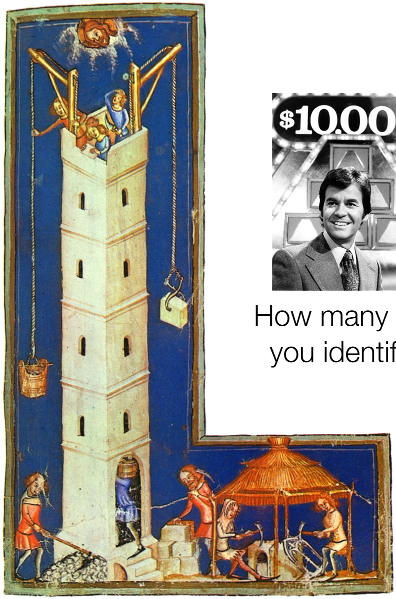
JS THE C PROGRAMMING LANGUAGE

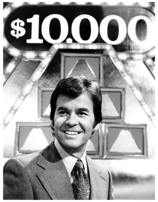
CLU

COBOL

julia

GO ERLANG php





How many can you identify?

Repetition

Fibonacci numbers

$$\begin{aligned} F_0 &= 0 \\ F_1 &= 1 \\ F_x &= F_{x-1} + F_{x-2} \end{aligned}$$

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \ldots$$

Loops

```go
func fib(x int) int {
    a := 0
    b := 1
    for i := 0; i < x; i++ {
        next := a + b
        a = b
        b = next
    }
    return a
}
```

$$\begin{aligned} F_0 &= 0 \\ F_1 &= 1 \\ F_x &= F_{x-1} + F_{x-2} \end{aligned}$$

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \ldots$$

Loops

```
sub fib {
    my $x = shift(@_);
    my $a = 0;
    my $b = 1;
    foreach (0..$x - 1) {
        my $next = $a + $b;
        $a = $b;
        $b = $next;
    }
    return $a;
}
```

$$\begin{aligned} F_0 &= 0 \\ F_1 &= 1 \\ F_x &= F_{x-1} + F_{x-2} \end{aligned}$$

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \ldots$$

Loops

```python
def fib(x):
    a = 0
    b = 1
    for i in range(0, x):
        a, b = b, a + b
    return a
```

$$\begin{aligned} F_0 &= 0 \\ F_1 &= 1 \\ F_x &= F_{x-1} + F_{x-2} \end{aligned}$$

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \ldots$$

Loops

```
long
fib(long x) {
    long a = 0;
    long b = 1;
    for (long i = 0; i < x; i++) {
        long next = a + b;
        a = b;
        b = next;
    }
    return a;
}
```

$$\begin{aligned} F_0 &= 0 \\ F_1 &= 1 \\ F_x &= F_{x-1} + F_{x-2} \end{aligned}$$

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \ldots$$

Loops

```
long
fib(long x) {
    long a = 0;
    long b = 1;
    for (long i = 0; i < x; i++) {
        long next = a + b;
        a = b;
        b = next;
    }
    return a;
}
```

$$\begin{aligned} F_0 &= 0 \\ F_1 &= 1 \\ F_x &= F_{x-1} + F_{x-2} \end{aligned}$$

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \ldots$$

Loops

```rust
fn fib(x: u64) -> u64 {
    let mut a = 0;
    let mut b = 1;
    for _ in 0..x {
        let next = a + b;
        a = b;
        b = next;
    }
    a
}
```

$$\begin{aligned} F_0 &= 0 \\ F_1 &= 1 \\ F_x &= F_{x-1} + F_{x-2} \end{aligned}$$

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \ldots$$

Recursion

```haskell
fib 0 = 0
fib 1 = 1
fib x = fib (x - 1) + fib (x - 2)
```

$$\begin{aligned} F_0 &= 0 \\ F_1 &= 1 \\ F_x &= F_{x-1} + F_{x-2} \end{aligned}$$

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \ldots$$

Recursion

```
long
fib(long x) {
    if (x == 0) return 0;
    if (x == 1) return 1;
    return fib(x - 1) + fib(x - 2);
}
```

$$\begin{aligned} F_0 &= 0 \\ F_1 &= 1 \\ F_x &= F_{x-1} + F_{x-2} \end{aligned}$$

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \ldots$$

Recursion

```haskell
fib x = fib' x 0 1
  where fib' 0 a b = a
        fib' x a b = fib' (x - 1) b (a + b)
```

$$\begin{aligned} F_0 &= 0 \\ F_1 &= 1 \\ F_x &= F_{x-1} + F_{x-2} \end{aligned}$$

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \ldots$$

Recursion

```
static long
fib_(long x, long a, long b) {
    if (x == 0) return a;
    return fib_(x - 1, b, a + b);
}

long
fib(long x) {
    return fib_(x, 0, 1);
}
```

$$\begin{aligned} F_0 &= 0 \\ F_1 &= 1 \\ F_x &= F_{x-1} + F_{x-2} \end{aligned}$$

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \ldots$$

Recursion schemes

Catamorphism
Paramorphism
Histomorphism

Recursion schemes

Catamorphism

Paramorphism

Histomorphism

Recursion schemes

Catamorphism
Paramorphism
Histomorphism

```
fib x = histo step x
  where step []      = 0
        step (_:[])  = 1
        step (a:b:_) = a + b
```

Handling failure

Digits

$$\begin{array}{rcl} \texttt{'0'} \ldots \texttt{'9'} & \Rightarrow & 0 \ldots 9 \\ \textit{anything else} & \Rightarrow & \textbf{error!} \end{array}$$

Exceptions

```java
public class ParseDigit {
    public static int parseDigit(char ch)
            throws NumberFormatException {
        if (ch ≥ '0' && ch ≤ '9') {
            return ch - '0';
        }
        throw new NumberFormatException();
    }
}
```

Exceptions

```python
def parse_digit(ch):
    if ord(ch) ≥ ord('0') and ord(ch) ≤ ord('9'):
        return ord(ch) - ord('0')
    raise ValueError("not a digit")
```

Exceptions

```
int
parse_digit(char ch) {
    if (ch ≥ '0' && ch ≤ '9') {
        return ch - '0';
    }
    throw invalid_argument("not a digit");
}
```

Exceptions

```
int
parse_digit(char ch) {
    if (ch >= '0' && ch <= '9') {
        return ch - '0';
    }
    throw invalid_argument("not a digit");
}

void
parse_file(const string& contents) {
    int digit = parse_digit(contents[0]);
}

void
use_file(const string& contents) {
    try {
        parse_file(contents);
    } catch (const invalid_argument& ex) {
        cout << ex.what() << endl;
    }
}
```

Error values

```c
int
parse_digit(char ch) {
    if (ch ≥ '0' && ch ≤ '9') {
        return ch - '0';
    }
    return -1;
}
```

Error values

```c
int
parse_digit(char ch) {
    if (ch ≥ '0' && ch ≤ '9') {
        return ch - '0';
    }
    return -1;
}
```

```c
int
parse_file(const char* contents) {
    int digit = parse_digit(contents[0]);
    if (digit == -1) {
        return -1;
    }
    return 0;
}

void
use_file(const char* contents) {
    int rc = parse_file(contents);
    if (rc == -1) {
        printf("not a digit!\n");
    }
}
```

Error values

```go
var InvalidDigit = errors.New("not a digit")

func ParseDigit(ch byte) (int, error) {
    if ch >= '0' && ch <= '9' {
        return int(ch - '0'), nil
    }
    return 0, InvalidDigit
}
```

```go
func ParseFile(contents string) error {
    _, err := ParseDigit(contents[0])
    if err != nil {
        return err
    }
    return nil
}

func UseFile(contents string) {
    err := ParseFile(contents)
    if err != nil {
        print(err)
    }
}
```

Error values

```rust
struct InvalidDigit;

fn parse_digit(ch: u8)
  -> Result<u8, InvalidDigit> {
    if ch >= b'0' && ch <= b'9' {
        return Ok(ch - b'0');
    }
    Err(InvalidDigit)
}
```

```rust
fn parse_file(contents: &[u8])
  -> Result<(), InvalidDigit> {
    parse_digit(contents[0])?;
    Ok(())
}

fn use_file(contents: &[u8]) {
    match parse_file(contents) {
        Ok(_) => {}
        Err(_) => println!("not a digit!"),
    }
}
```

Error values

```haskell
data InvalidDigit = InvalidDigit

parseDigit ch =
  if ch ≥ '0' && ch ≤ '9' then
    Right (ord ch - ord '0')
  else
    Left InvalidDigit
```

```haskell
parseFile contents = do
  parseDigit (head contents)

useFile contents =
  case parseFile contents of
    Right _ → pure ()
    Left _  → print "not a digit"
```

Cleaning up

Manual memory management

```c
struct person {
    char* name;
    int age;
};

struct person*
person_new(const char *name, int age) {
    struct person* person =
        malloc(sizeof(struct person));
    person->name = strdup(name);
    person->age = age;
    return person;
}

void
person_free(struct person* loc) {
    free(loc->name);
    free(loc);
}
```

Manual memory management

```c
struct person {
    char* name;
    int age;
};

struct person*
person_new(const char *name, int age) {
    struct person* person =
        malloc(sizeof(struct person));
    person->name = strdup(name);
    person->age = age;
    return person;
}

void
person_free(struct person* loc) {
    free(loc->name);
    free(loc);
}
```

```c
void
process_family(void) {
    struct person* me = person_new("Doug", 42);
    printf("%s is %d years old\n", me->name, me->age);
    person_free(me);
}
```

Manual memory management

```c
struct person {
    char* name;
    int age;
};

struct person*
person_new(const char *name, int age) {
    struct person* person =
        malloc(sizeof(struct person));
    person->name = strdup(name);
    person->age = age;
    return person;
}

void
person_free(struct person* loc) {
    free(loc->name);
    free(loc);
}
```

```c
void
process_family(void) {
    struct person* me = person_new("Doug", 42);
    printf("%s is %d years old\n", me->name, me->age);
    /* person_free(me); */
}
```

Automatic memory management

```go
type Person struct {
    Name string
    Age  int
}
```

```go
func ProcessFamily() {
    me := Person{Name: "Doug", Age: 42}
    fmt.Printf("%s is %d years old\n", me.Name, me.Age)
}
```

Automatic memory management

```python
@dataclass
class Person:
    name: str
    age: int
```

```python
def process_family():
    me = Person("Doug", 42)
    print(f"{me.name} is {me.age} years old")
```

Automatic memory management

```cpp
struct person {
    string* name;
    int age;

    person(string name_, int age_) {
        name = new string(name_);
        age = age_;
    }

    ~person() {
        delete name;
    }
};
```

```cpp
void
process_family(void) {
    person* me = new person("Doug", 42);
    cout << *me->name << " is "
         << me->age << " years old" << endl;
    delete me;
}
```

Automatic memory management

```cpp
struct person {
    unique_ptr<string> name;
    int age;

    person(const string& name, int age) :
        name(make_unique<string>(name)),
        age(age) {}
    ~person() = default;
};
```

```cpp
void
process_family(void) {
    shared_ptr<person> me =
        make_shared<person>("Doug", 42);
    cout << *me->name << " is "
         << me->age << " years old" << endl;
}
```

Automatic memory management

```
struct Person {
    name: Box<String>,
    age: u8,
}

impl Person {
    fn new(name: &str, age: u8) -> Person {
        let name = name.to_string();
        let name = Box::new(name);
        Person { name, age }
    }
}
```

```
fn process_family() {
    let me = Arc::new(Person::new("Doug", 42));
    println!("{} is {} years old", me.name, me.age);
}
```

Managing other resources

```c
int
save_file(const char* filename)
{
    FILE* fp = fopen(filename, "w");
    if (fp == NULL) goto error0;

    int rc = fputs("lots of interesting data", fp);
    if (rc < 0) goto error1;

    fclose(fp);
    return 0;

error1:
    fclose(fp);
error0:
    return -1;
}
```

Managing other resources

```
void
save_file(const char* filename)
{
    ofstream fp(filename);
    fp << "lots of interesting data";
}
```

Managing other resources

```rust
fn save_file(filename: &str) -> Result<(), std::io::Error> {
    let mut fp = File::create(filename)?;
    write!(fp, "lots of interesting data")?;
    Ok(())
}
```

Managing other resources

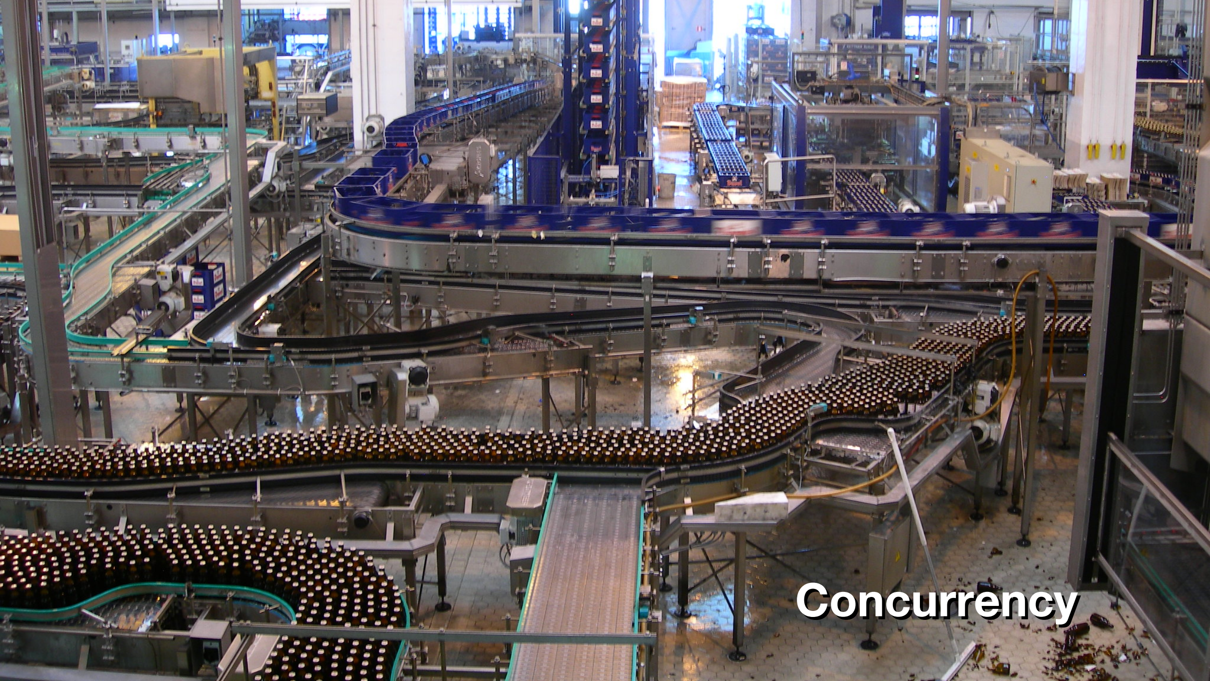
```go
func SaveFile(filename string) error {
    fp, err := os.Open(filename)
    if err != nil {
        return err
    }
    defer fp.Close()

    _, err = fp.WriteString("lots of interesting data")
    if err != nil {
        return err
    }

    return nil
}
```

Managing other resources

```python
def save_file(filename):
    with open(filename, "w") as fp:
        fp.write("lots of interesting data")
```

Managing other resources

```zig
fn save_file(filename: []const u8) !void {
    const cwd = std.fs.cwd();
    var fp = try cwd.createFile(filename, .{});
    defer fp.close();
    _ = try fp.write("lots of interesting data");
}
```



Concurrency

Goroutines

```go
func DownloadFiles() error {
    err := Download("https://a.example.com/a.csv")
    if err != nil {
        return err
    }

    err = Download("https://b.example.com/b.csv")
    if err != nil {
        return err
    }

    err = Download("https://c.example.com/c.csv")
    if err != nil {
        return err
    }

    return nil
}
```

Goroutines

```go
func DownloadFiles() error {
    go Download("https://a.example.com/a.csv")
    go Download("https://b.example.com/b.csv")
    go Download("https://c.example.com/c.csv")
    return nil
}
```

Goroutines

```go
func DownloadFiles() error {
    var wg sync.WaitGroup
    var errA error
    var errB error
    var errC error
    go downloadOne("https://a.example.com/a.csv", &wg, &errA)
    go downloadOne("https://b.example.com/b.csv", &wg, &errB)
    go downloadOne("https://c.example.com/c.csv", &wg, &errC)
    wg.Wait()

    if errA != nil { return errA }
    if errB != nil { return errB }
    if errC != nil { return errC }
    return nil
}

func downloadOne(url string, wg *sync.WaitGroup, err *error) {
    wg.Add(1)
    defer wg.Done()
    *err = Download(url)
}
```

OS threads

```python
def download_files():
    a = download_one("https://a.example.com/a.csv")
    b = download_one("https://b.example.com/b.csv")
    c = download_one("https://c.example.com/c.csv")
    a.join()
    b.join()
    c.join()

def download_one(url):
    thread = threading.Thread(target=download, args=(url,))
    thread.start()
    return thread
```

OS threads

```python
def download_files():
    files = 100000 * ["https://a.example.com/a.csv"]
    threads = [download_one(url) for url in files]
    for thread in threads:
        thread.join()

def download_one(url):
    thread = threading.Thread(target=download, args=(url,))
    thread.start()
    return thread
```

Async / futures / promises / tasks

```js
function downloadFiles() {
    return download("https://a.example.com/a.csv").then(
        () => download("https://b.example.com/b.csv").then(
            () => download("https://c.example.com/c.csv")
        )
    );
}
```

Async / futures / promises / tasks

```
function downloadFiles() {
    return Promise.all([
        download("https://a.example.com/a.csv"),
        download("https://b.example.com/b.csv"),
        download("https://c.example.com/c.csv"),
    ]);
}
```

Async / futures / promises / tasks

```javascript
function downloadFiles() {
    return download("https://a.example.com/a.csv").then(
        () => download("https://b.example.com/b.csv").then(
            () => download("https://c.example.com/c.csv")
        )
    );
}
```

Async / futures / promises / tasks

```javascript
async function downloadFiles() {
    await download("https://a.example.com/a.csv");
    await download("https://b.example.com/b.csv");
    await download("https://c.example.com/c.csv");
}

async function download(url) { console.log(url); }
```

Async / futures / promises / tasks

```js
async function downloadFiles() {
    await Promise.all([
        download("https://a.example.com/a.csv"),
        download("https://b.example.com/b.csv"),
        download("https://c.example.com/c.csv"),
    ]);
}
```

Async / futures / promises / tasks

```python
async def download_files():
    await asyncio.gather(
        download("https://a.example.com/a.csv"),
        download("https://b.example.com/b.csv"),
        download("https://c.example.com/c.csv"),
    )
```

Async / futures / promises / tasks

```rust
async fn download_files() -> Result<(), Error> {
    futures::try_join!(
        download("https://a.example.com/a.csv"),
        download("https://b.example.com/b.csv"),
        download("https://c.example.com/c.csv"),
    )?;
    Ok(())
}
```

Picture credits

Slide 2	Meister der Weltenchronik, “Weltchronik in Versen, Szene: Der Turmbau zu Babel” Public domain, `https://commons.wikimedia.org/wiki/File:Meister_der_Weltenchronik_001.jpg`
Slide 2	ABC Television, “The $10,000 Pyramid” Public domain, `https://commons.wikimedia.org/wiki/File:Dick_Clark_$10000_Pyramid.JPG`
Slide 3	Matjaž Mirt, “Tern/čigra” CC-BY-2.0, `https://flic.kr/p/2kXydKp`
Slide 4	Mark Gunn, “This just tern’ed into a swarm!” CC-BY-2.0, `https://flic.kr/p/P11JH1`
Slide 5	Dave Shafer, “Luzern mirror maze, Switzerland” CC-BY-2.0, `https://flic.kr/p/5RQPx9`
Slide 10	Blondinrikard Fröberg, “Fail” CC-BY-2.0, `https://flic.kr/p/B9WA8y`
Slide 14	Alan Levine, “King of the Trash Hill” CC-BY-2.0, `https://flic.kr/p/eRjo3W`
Slide 18	Jim, “Duvel Assembly Line” CC-BY-SA-2.0, `https://flic.kr/p/NpqGS`
Slide 22	Mark Gunn, “Time to tern in” CC-BY-2.0, `https://flic.kr/p/NHDXRb`