CS50

What are algorithms?

Algorithms are step-by-step methods designed to solve specific problems and are suitable for implementation as computer programs.

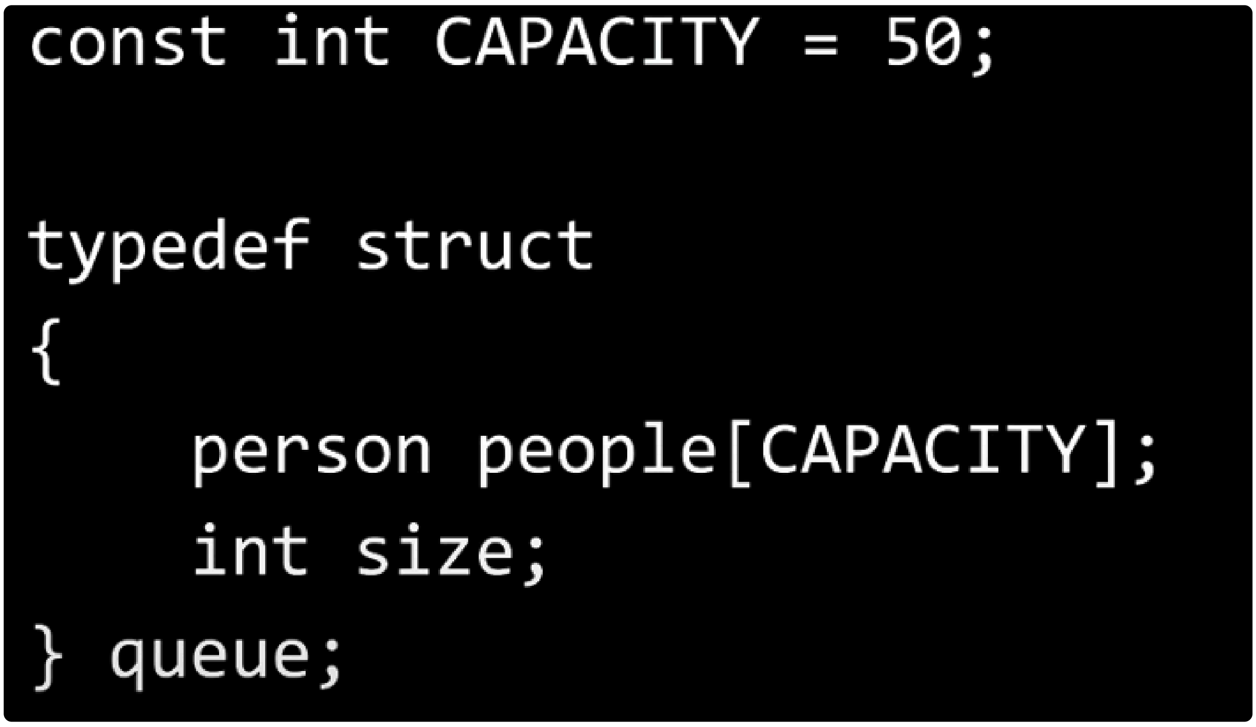
And Data Structures?

Data Structures are algorithms (problem-solving methods) for organizing data in different ways.

When solving a problem, there are often multiple approaches. For small problems, the choice of method might not have a significant impact. However, for larger problems, it's important to choose the most efficient methods in terms of time and space. A well-designed algorithm can lead to substantial savings in both computational resources and investment—savings that cannot be achieved merely by using a more powerful hardware.

Abstract Data Structures

- Queues: FIFO (First In, First Out).
  Queues operate on a FIFO (First In, First Out) principle, where the first element added is the first one to be removed. The core operations include **enqueueing (adding elements to the queue)** and **dequeuing (removing elements from the queue).** This process can be likened to being enlisted for an activity and then dismissed after its completion, ensuring that tasks or elements are handled in the exact order they arrive.



```c
const int CAPACITY = 50;

typedef struct
{
    person people[CAPACITY];
    int size;
} queue;
```

Example in C language.

- Stacks: LIFO (Last In, First Out).
  Stacks operate in a LIFO (Last In, First Out) principle, where the last element added is the first one to be removed. The core operations include **pushing (adding elements to the stack)** and **popping (removing elements from the stack)**. A common example is Gmail's inbox, where new emails appear on top of older ones, resembling the behavior of a stack.

Empty Stack:

```
|       |
|       |
|       |
---------
```

After Pushing `A`, you get:

```
|       |
|       |
|   A   |
---------
```

After Pushing `B`, you get:

```
|       |
|   B   |
|   A   |
---------
```

After Popping, you get:

```
|       |
|       |
|   A   |
---------
```

After Pushing `C`, you get:

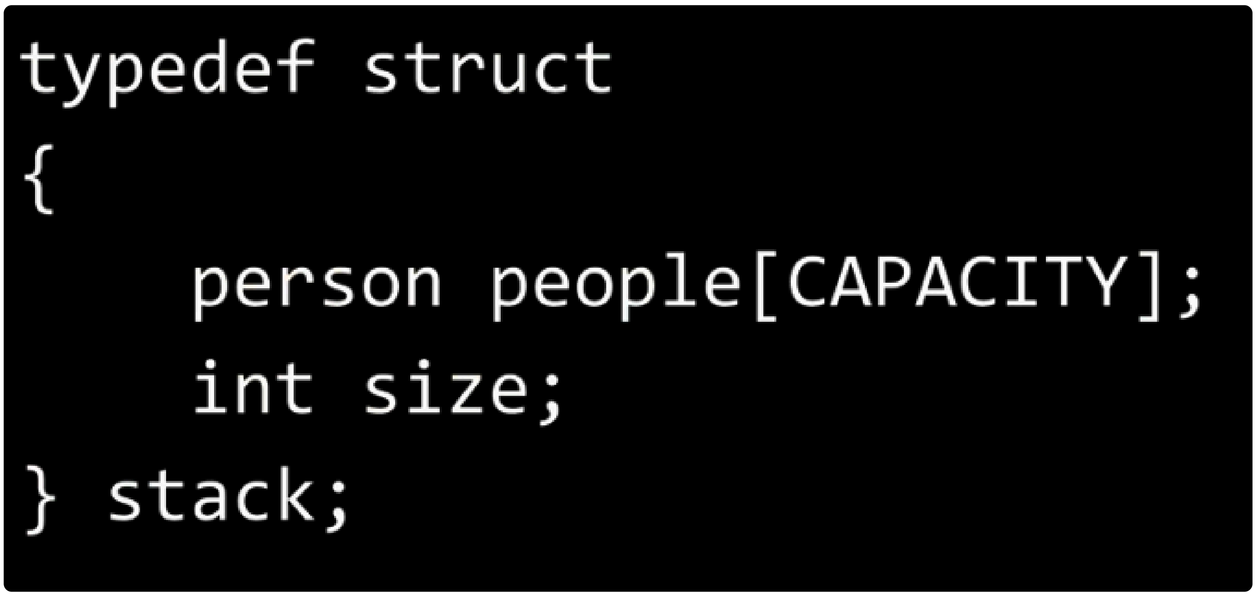
```
|       |
|   C   |
|   A   |
---------
```

After Popping, you get:

```
|       |
|       |
|   A   |
---------
```

After Popping, you get:

```
|       |
|       |
|       |
---------
```



```
typedef struct
{
    person people[CAPACITY];
    int size;
} stack;
```

Example in C language, it's just like what a queue implementation is, but it does have significant differences when it comes to the deletion of elements (the newest ones are deleted first).