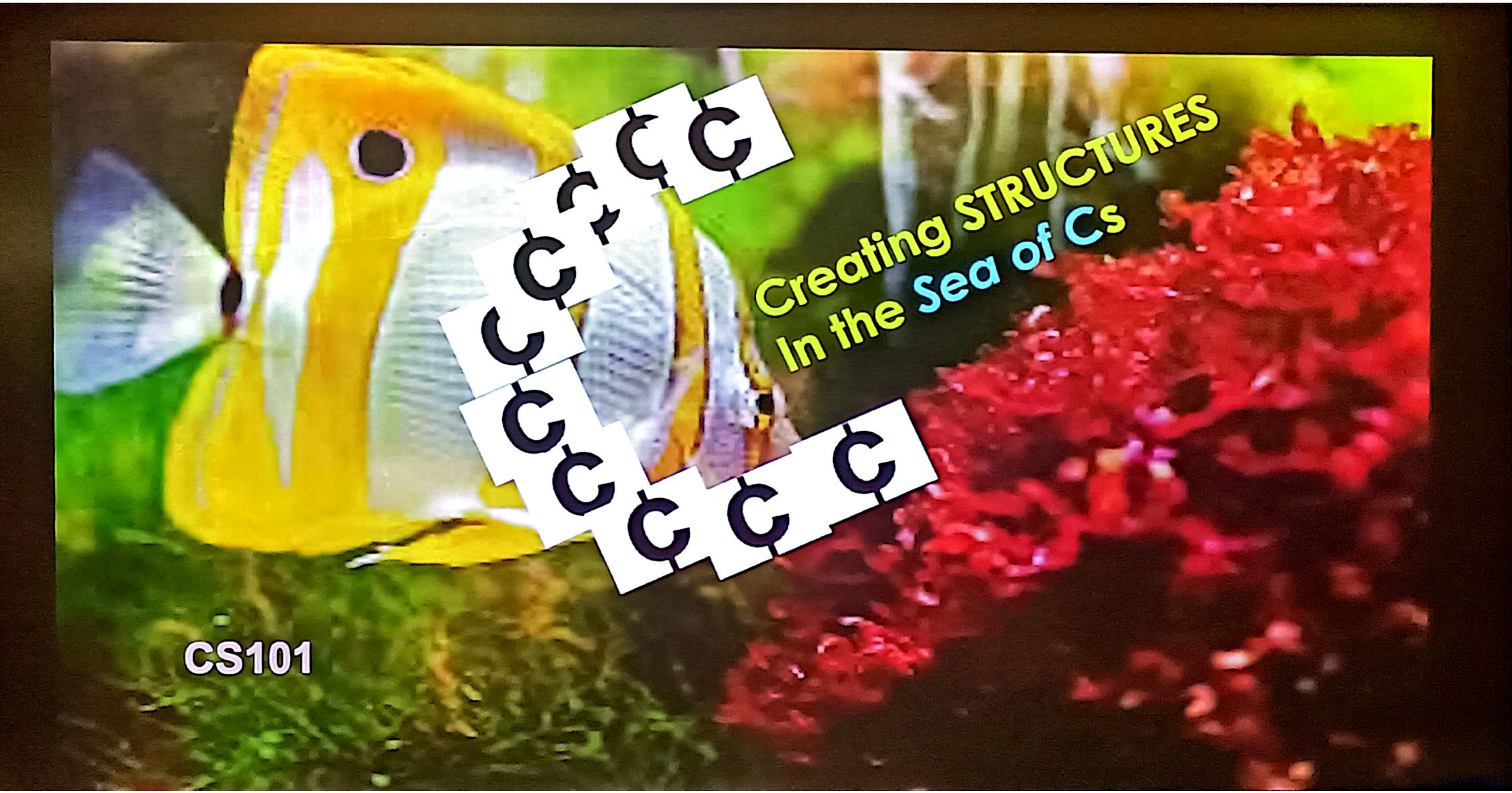

Creating STRUCTURES
In the Sea of Cs
CS101

Sharks

STORING DETAILS

Name: Zebra

Length: 3.5 m

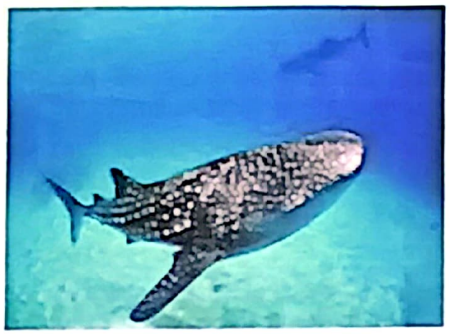
Lifespan: 30 years

char name[6];

Name

Z	E	B	R	A	\0

float length;
int lifespan;



Name: Whale

Length: 18.9 m

Lifespan: 130 years

char name[6];

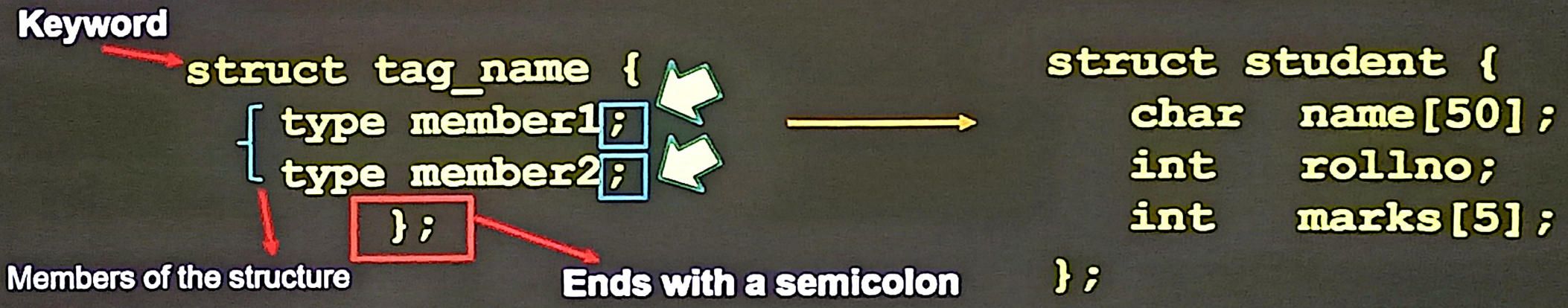
Name

W	H	A	L	E	\0

float length;
int lifespan;

Structures (Definition & Declaration)

- User-defined data type
- **Collection** of one or more variables, possibly of different types
- Helps to organize complicated data in large programs



Variable Declaration

New data type is defined

```
struct student {
    char   name[50];
    int    rollno;
    int    marks[5];
};
```

Structure is a user-defined data type

Let us create a variable of our new data type "*student*".....

```
/* Variable declaration */
struct structureName structureVariable;
```

```
struct student st1;
struct student st2,st3;
```

Variable Declaration

```
struct student {
    char   name[50];
    int    rollno;
    int    marks[5];
};
```

The members of the structure do not occupy memory until a structure variable is created.

Let us create a variable of our new data type "*student*".....

```
/* Variable declaration */
struct structureName structureVariable;
```

```
struct student st1;
struct student st2,st3;
```

Variable Definition Cum Declaration

The structure definition and variable declaration can be combined:

```
struct student {
    char   name[50];
    int    rollno;
    int    marks[5];
}st1,st2;
```

More on Structure Declaration…

```
struct tag_name {
    type member1;
    type member2;
    };
```

Completely optional

VALID!!!

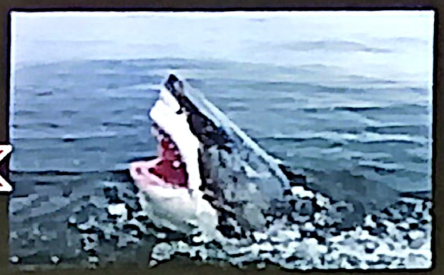
```
struct {
    char   name[50];
    int    rollno;
    int    marks[5];
}st;
```

More on Structure Declaration...

C doesn't allow variable initialization inside a structure declaration.

INVALID!!!

```
struct {
    char   name[50];
    int    rollno = 2101122;
    int    marks[5];
};
```



Initializing members of a Structure

Like any other variable, a structure variable can be initialized where they are declared. There is a one-to-one relationship between the members and their initialized values.

Definition

```
struct student {
    char   name[50];
    int    rollno;
    int    marks[5];
};
```

MEMBERS

Declaration

```
struct student st1;
struct student st2,st3;
```

VARIABLES

Initializing members of a Structure

Like any other variable, a structure variable can also be initialized where they are declared. There is a one-to-one relationship between the members and their initializing values.

```
/* Variable Initialization */

  struct structureName = { value1, value2,...};
```

Definition

```
struct student {
    char   name[50];
    int    rollno;
    int    marks[5];
};
```

Declaration and Initialization

```
struct student st1 = { "Alex", 43, {76, 78, 56, 98, 92}};
Struct student st2 = { "Max", 33, {87, 84, 82, 96, 78}};
```

Accessing members of a Structure

To access the members, we have to use . (the dot) operator.

```
/* Accessing Members of a Structure */
structureVariable.memberVariable
```

```
/* Example */
printf("Name: %s\n", st1.name);
printf("Roll No.: %d\n", st1.rollno);
for( int i = 0; i < 5; i++)
  printf("Marks in %dth subject: %d\n", i, st2.marks[i]);
```

Dot operator

```
/* Output */
Name: Alex
Roll No.: 43
Marks in 0th subject: 87
Marks in 1th subject: 84
Marks in 2th subject: 82
Marks in 3th subject: 96
Marks in 4th subject: 78
```

```
struct student st1 = { "Alex", 43, {76, 78, 56, 98, 92}};
struct student st2 = { "Max", 33, {87, 84, 82, 96, 78}};
```

st1.name = Alex
st1.rollno = 43
st1.marks[4] = ?

Let us Code it!!

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

struct student
  {
     char name[50];
     int rollno;
     int marks[5];
  };


void main()
{
   struct student st1= { "Alex", 43, {76, 78, 56, 98, 92}};

   printf("Name: %s",st1.name);
   printf("\nRoll No.:%d",st1.rollno);
   for(int i=0;i<5;i++)
   {
      printf("\nMarks %d:%d",i+1,st1.marks[i]);
   }
}
```

Variable declaration

Accessing *name "Alex"*

Loop to access marks of all subjects

Let us Code it!!

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

struct student
    {
        char name[50];
        int rollno;
    };

int main()
{
    struct student st;
    printf("Enter the name of the student:\n");
    scanf("%s",st.name);
    printf("Enter the roll no. of the student:\n");
    scanf("%d",&st.rollno);
    printf("Name is: %s\n", st.name);
    printf("Roll no. is: %d\n", st.rollno);
    return 0;
}
```

Output:

```
Enter the name of the student:
Alex
Enter the roll no. of the student:
100
Name is: Alex
Roll no. is: 100
```

Structures using Array

```
struct student
    {
        char name[50];
        int rollno;
        int marks[5];
    };
```

```
struct student stu[100];
```

To access the elements of the array stu and the members of each element, we can use loops.

```
/* Reading values from the user */

for(int i = 0; i
    {
        printf("Enter name:\n");
        scanf("%s",stu[i].name);
        printf("Enter roll:\n");
        scanf("%d",&stu[i].rollno);

        for( int
        {
            printf("Enter marks of %dth
subject:\n",j);
            scanf("%d",&stu[i].marks[j]);
        }

        printf("\n--------------------\n\n");
    }
```

```
scanf("%s",stu[i].name);
```

```
scanf("%s",&stu[i].rollno);
```

Nested Structure

Nesting a structure means having one or more structure variables inside another structure.

Like we declare an `int` member or `char` member, we can also declare a structure variable as a member.

```
struct mydate
    {
        int dt;
        int mt;
        int yr;
    };

struct student
    {
         char name[20];
         int rollno;
         int marks[5];
         struct mydate dob;
    };
```

An Example Program

```
#include <stdio.h>

struct mydate
    {
        int dt;
        int mt;
        int yr;
    };

struct student
    {
        char name[20];
        int rollno;
        int marks[5];
        struct mydate dob;
    };

int main()
{
    struct student stu = {"Alex", 43, {76,78,56,98,
92}, 1, 1, 1994};
    printf(" Year of birth is: %d \n", stu.dob.yr);
    return 0;
}
```

Output:

Year of birth is: 1994