CSE102
Computer Programming

```
struct number {
        int img;
        float real;
};
```

Credits: www.programiz.com

Syntax

keyword

```
struct structure_name{
    data_type var1;
    data_type var2;
    data_type var3;
    ...
};
```

Collection of different types of variables under single name

Credits: www.programiz.com

Example

```
struct person{
    char name[30];
    char gender;
    int  age;
    char address[][60];
    int  aadhar_number;
};
```

Person is user-defined type!!

Variable Declaration

```
struct person{
    char name[30];
    char gender;
    int  age;
    char address[][60];
    int  aadhar_number;
}person1, person2[20];
```

Credits: How to Program C

Variable Declaration

```
int main(){
  struct person person1,
                person2[50];
  return 0;
}
```

array of structures

Credits: www.programiz.com

Accessing Members

```
person1.gender
persn1.age
person1.aadhar_number
```

Examples: studentInfo, addDistance

Custom Datatype

```
typedef struct person{
    char name[30];
    char gender;
    int  age;
    char address[][60];
    int  aadhar_number;
} personnel;
```

Variable Declaration

```
int main(){
 personnel person1,
                person2[50];
 return 0;
}
```

Your own datatype!!

Nested Structures

```
struct employee{
  struct person persnl;
  char designation[20];
  int experience;
} emp[1000];
```

Accessing age of person

emp[0].persnl.age

Structure Pointers

```
struct name {
        member1;
        member2;
        .
        .
};

int main()
{
        struct name *ptr;
}
```

Credits: www.programiz.com

Union

Like structure. As easy as replacing struct keyword with union

```
union car {
        char name[50];
        float price;
}
```

Credits: www.programiz.com

Union Declaration

```
union car
{
   char name[50];
   int price;
} car1, car2, *car3;
```

```
car1.price
```

```
(*car3).price
or;
car3->price
```

```
union car
{
   char name[50];
   int price;
};

int main()
{
   union car car1, car2, *car3;
   return 0;
}
```

Credits: www.programiz.com

Union vs Structure

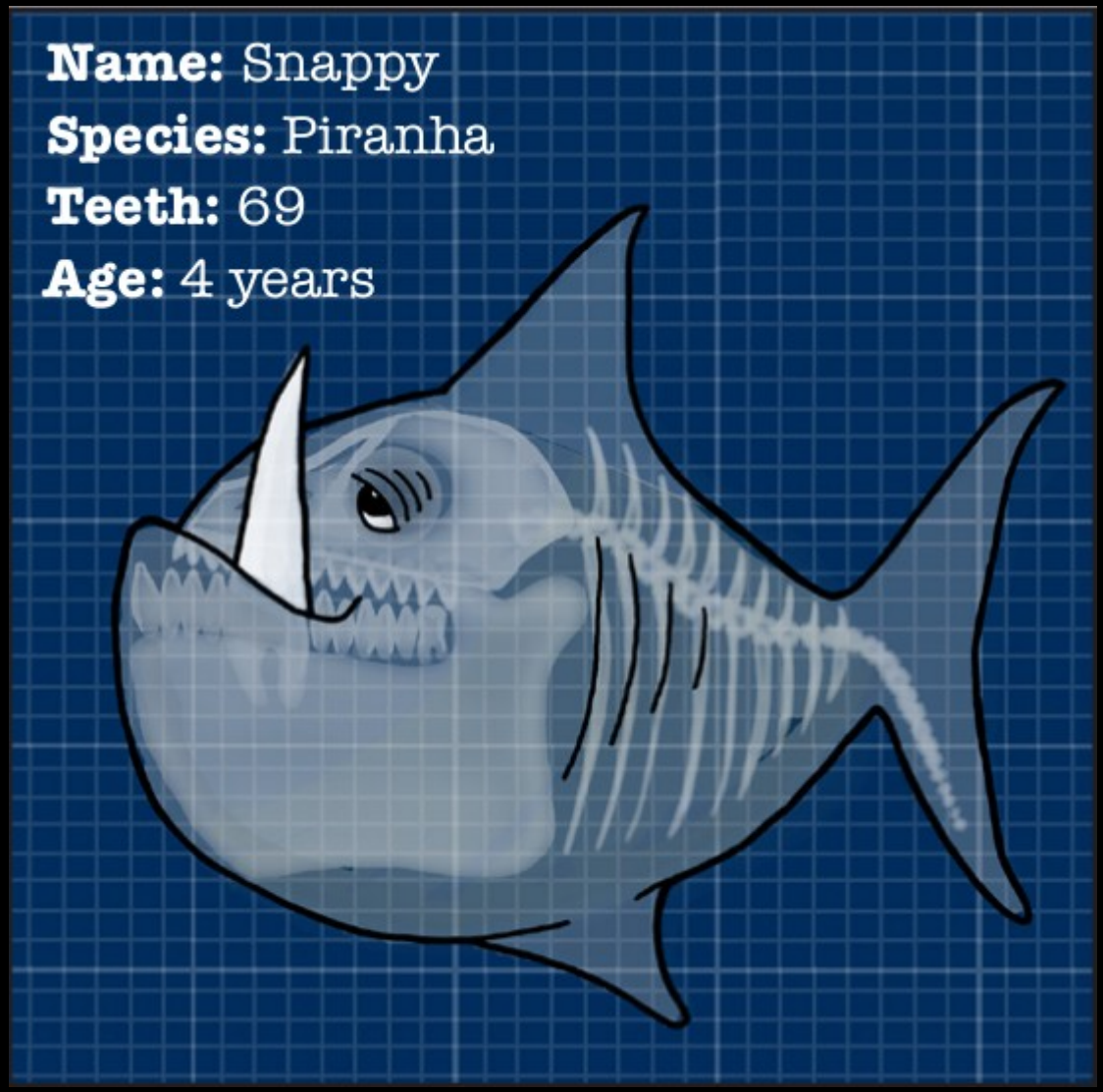
Fig: Memory allocation in case of structure

Fig: Memory allocation in case of union

Credits: www.programiz.com

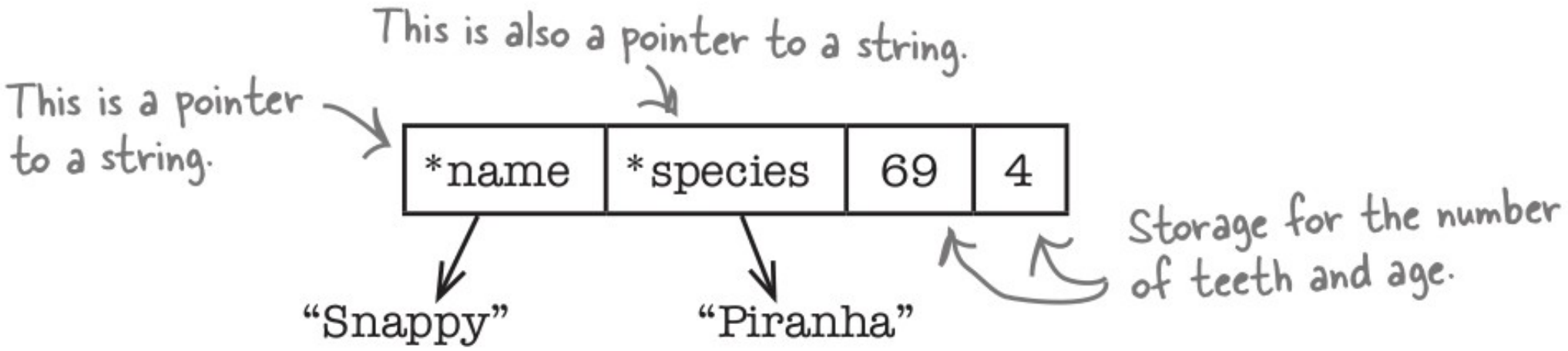
```
struct packetheader {
    int sourceaddress;
    int destaddress;
    int messagetype;
    union request {
        char fourcc[4];
        int requestnumber;
    };
};
```

```
struct fish {
  const char *name;
  const char *species;
  int teeth;
  int age;
};
```

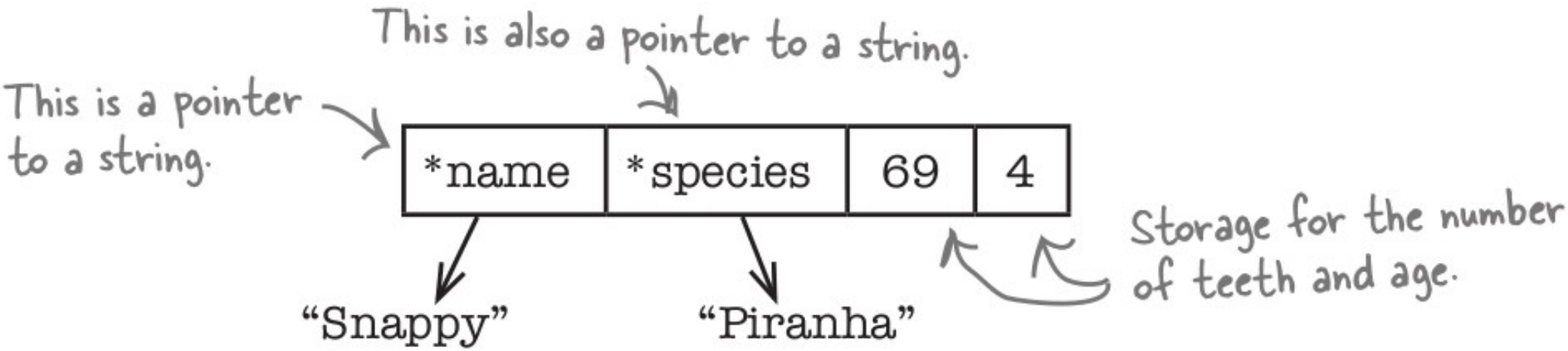


```
struct fish {
  const char *name;
  const char *species;
  int teeth;
  int age;
};
```

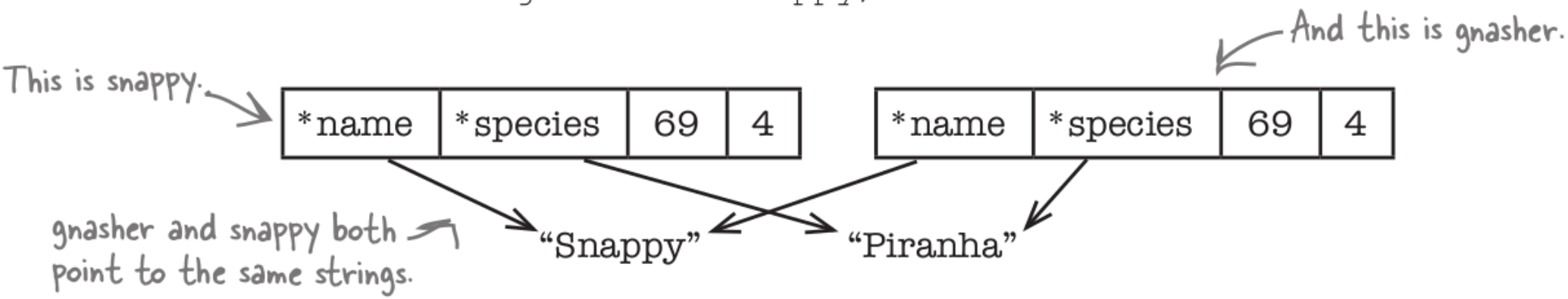
```
struct fish snappy = {"Snappy", "Piranha", 69, 4};
```



```
struct fish snappy = {"Snappy", "Piranha", 69, 4};
```




```
struct fish snappy = {"Snappy", "Piranha", 69, 4};
struct fish gnasher = snappy;
```

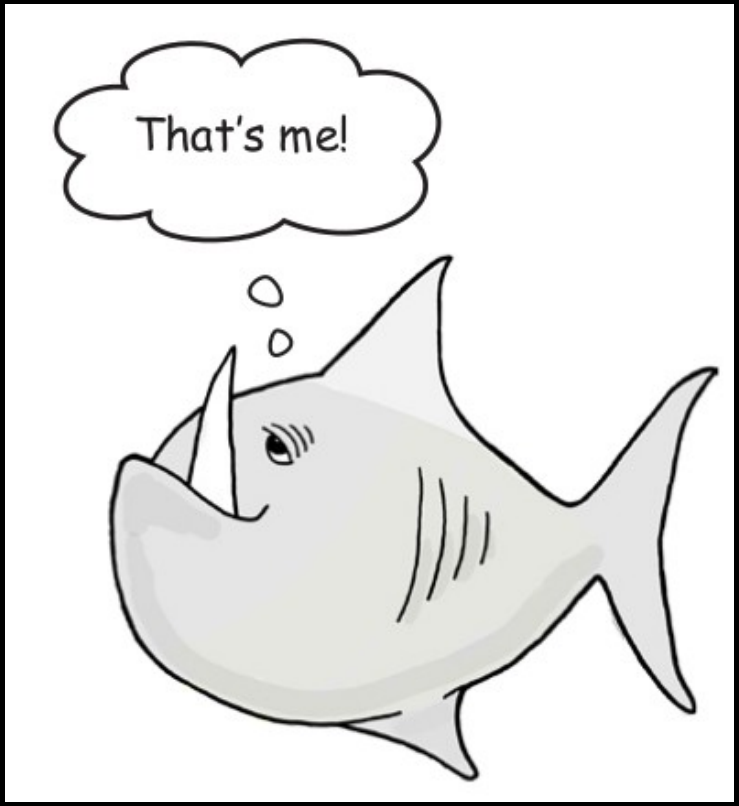


Remember: when you're assigning struct variables, you are telling the computer to *copy* data.

```
int main()
{
  catalog("Snappy", "Piranha", 69, 4);
  label("Snappy", "Piranha", 69, 4);
  return 0;
}
```

You are passing the same four pieces of data twice.

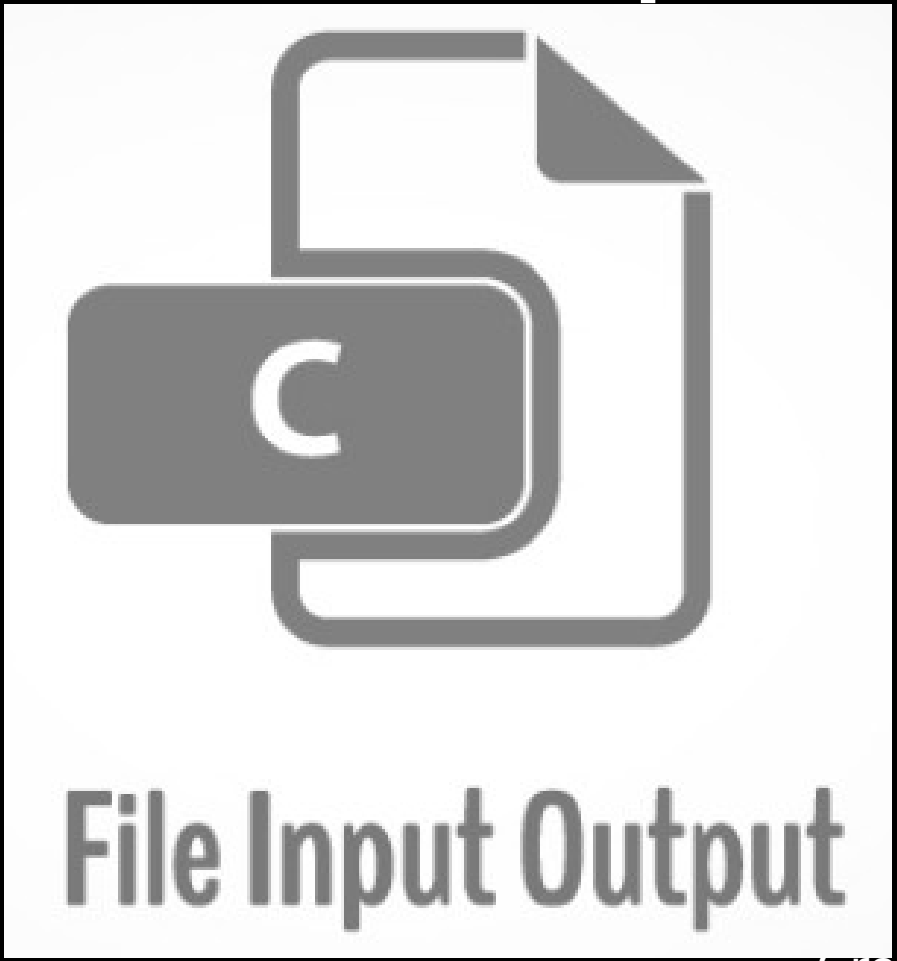
There's only one fish, but you're passing four pieces of data.



```
/* Print out the catalog entry */
void catalog(struct fish f)
{
   ...
}

/* Print the label for the tank */
void label(struct fish f)
{
   ...
}
```

CSE102
Computer Programming
(Next Topic)



Credits: www.programiz.com