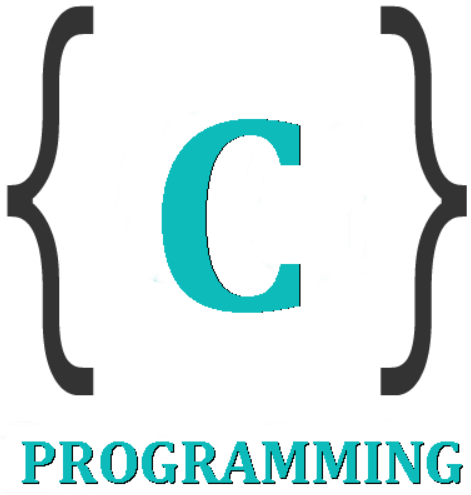
Introduction to Programming

Structures

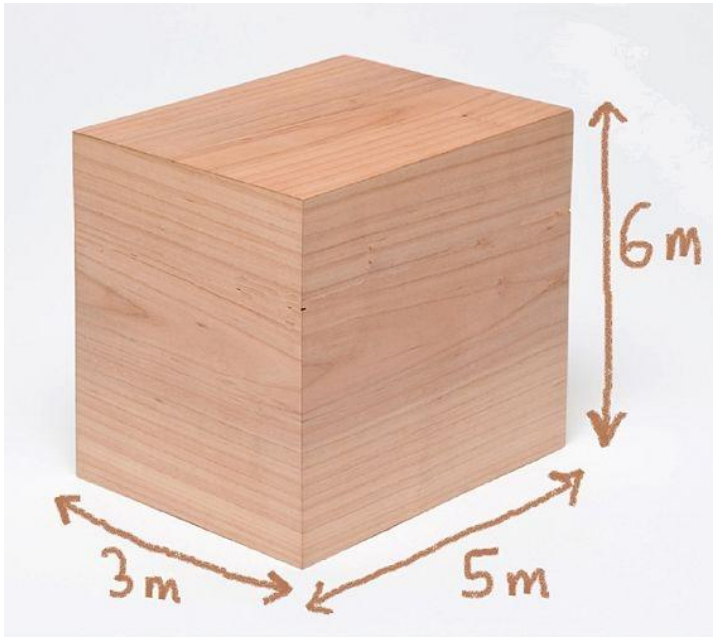
Mahmoud El-Gayyar

elgayyar@ci.suez.edu.eg



What is the Problem?

- *We need to compute the volume of a box*
  - = height * width * length
- *What if you need to compute the volume of two boxes?*
- *What about 100 boxes.*

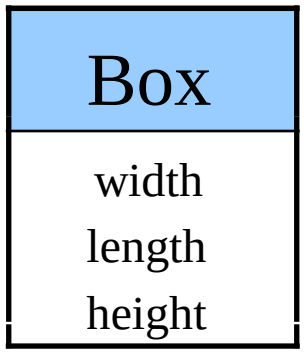


Solution: Structures

- *Aggregating associated data into a single variable*

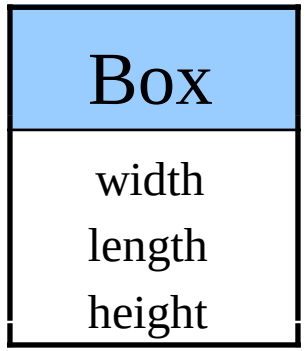
```
int main()
{
        Box mybox;
        Circle c;

        mybox.width = 10;
        mybox.length = 30;
        mybox.height = 10;
        c.radius = 10;
}
```



The Idea

- *I want to describe a box. I need variables for the width, length, and height.*
- *I can use three variables, but wouldn't it be better if I had a single variable to describe a box?*
- *That variable can have three parts, the width, length, and height.*



Structs

A struct (short for structure) in C is a grouping of variables together into a single type

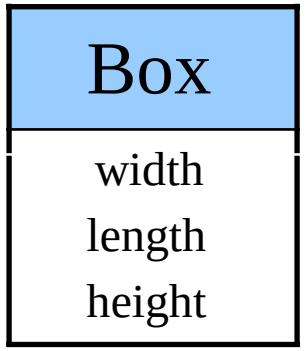
```
struct nameOfStruct
{
    type member;
    type member;
};
```

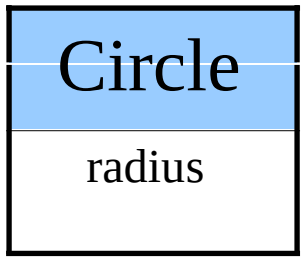
Note the semicolon at the end.

To declare a variable:

struct nameOfStruct variable_name;

Example 1 - I






```
#include <stdio.h>

struct Box
{
        int width;
        int length;
        int height;
};

struct Circle
{
        double radius;
};

int main()
{
        struct Box b;
        struct Circle c;
}
```

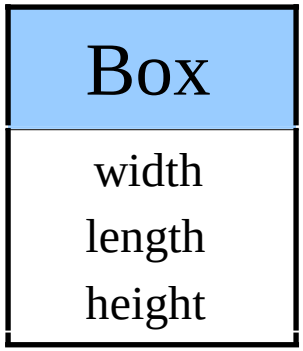
Data structure definition

You can declare variables

Example 1 - II

```
#include <stdio.h>

struct Box
{
        int width;
        int length;
        int height;
};
```




```
int main() {
        struct Box b;

        b.width = 10;
        b.length = 30;
        b.height = 10;

}
```

You can assign values to each member

We use a period "." to get to the elements of a struct.

If x is a struct, x.width is an element in a struct.

Example 2

```
struct bankRecord
{
  char name[50];
  float balance;
};
```

You can use mixed data types within the struct (int, float, char [])

```
struct bankRecord billsAcc;

cin>> billsAcc.balance;

cout<<“My balance is: ” << billsAcc.balance;

float bal = billsAcc.balance;
```

Copy Via =

You can set two struct type variables equal to each other and each element will be copied

```
struct Box {
    int width, length, height;
};

int main()
{
    struct Box b, c;
    b.width = 5; b.length=1; b.height = 2;
    c = b;          // copies all elements of b to c
    printf("%d %d %d\n", c.width, c.length, c.height);
}
```

Passing Structure to Functions

- You can pass a struct to a function. All the elements are copied.
- **Call by Value**

```
int myFunction(struct Person p){

        …

}
```

Using Structure in Functions

Write a program that

- Prompts the user to enter the dimensions of a 3D box
- Prints the volume of the box (use a function to compute the volume)

Sample run:

```
Enter the box dimensions (width,length,height): 1 2 3

Box volume = 6
```

Quiz Answer

```
struct Box { int width, height , length; };

int GetVolume(struct Box b)
{
     return b.width * b.height * b.length;
}

int main()
{
     struct Box b;

     cout<<"Enter the box dimensions (width length height): ";
     cin>> b.width >> b.length >> b.height;

     cout<< "Box volume =  " << GetVolume(b);
}
```

== Comparison doesn't Work

```
struct Box { int width, length, height; };

int main()
{
        struct Box b, c;
        b.width = 5; b.length=1; b.height = 2;
        c = b;
        if (c == b)          /* Error when you compile! */
                printf("c and b are identical\n");
        else
                printf("c and b are different\n");
} t
```

Error message: invalid operands to binary == (have 'Box' and 'Box')

Create Your Own Equality Test

```
struct Box { int width, height , length; };

int IsEqual(struct Box b, struct Box c)
{
   if (b.width==c.width &&
       b.length==c.length &&
       b.height==c.height)
       return 1;
   else
       return 0;
}
```

```
struct Box b, c;
b.width = 5; b.length=1; b.height = 2;
c = b;

if (IsEqual(b,c))
    printf("c and b are identical\n");
else
    printf("c and b are different\n");
```

typedef

typedef is a way in C to give a name to a custom type.

```
typedef type newname;

typedef int dollars;
typedef unsigned char Byte;
```

I can declare variables like:

dollars d;
Byte b, c;

It's as if the type already existed.

Using Structs with Typedef

```
typedef struct [nameOfStruct]
{
    type member;
    type member;
} TypeName;
```

optional

To declare a variable: **TypeName** variable_name;

Example

```
#include <stdio.h>

typedef struct
{
        int width;
        int length;
        int height;

} Box;


int main()
{
        Box b;   /* instead of struct Box */
        b.width = 10;
        b.length = 30;
        b.height = 10;
}
```

Arrays of Structs

You can declare an array of a structure and manipulate each one

```
typedef struct
{
  double radius;
  int x;
  int y;
  char name[10];
} Circle;

Circle circles[5];
```

Size of a Struct

```
typedef struct
{
  double radius;        /* 8 bytes */
  int x;                /* 4 bytes */
  int y;                /* 4 bytes */
  char name[10];        /* 10 bytes */
} Circle;

printf("Size of Circle struct is %d\n",
        sizeof(Circle));
```

Size of a Struct

8 + 4 + 4 + 10 = 26

- But sizeof() reports 28 bytes!!!

Most machines require alignment on 4-byte boundary (a word)

- last word is not filled by the char (2 bytes used, 2 left over)

DDDD	DDDD	IIII	IIII	CCCC	CCCC	CCXX
8 byte, 2 word double		4 byte, 1 word integer	4 byte, 1 word integer	10 byte char array, 2 bytes of the last word unused		

Summary

- *Structures*
  - Copy via =
  - Sending structures to functions
  - == is not working
- *Typedef*
- *Array of Structures*
- *Size of Structures*