Programming in C

Chapter - 8

Structures and Unions

by, Santa Basnet

Structures and Unions

Introduction

A structure is a collection of one or more variables, possibly of different types, grouped together under a single name for convenient handling.

Structures and Unions

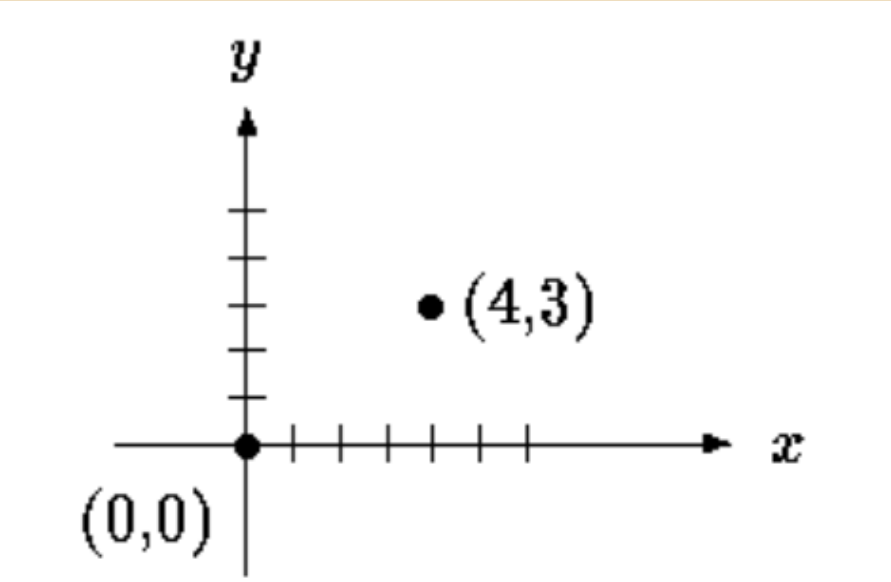
Introduction

Structures help to organize complicated data, particularly in large programs, because they permit a group of related variables to be treated as a unit instead of as separate entities.

Structures and Unions

Illustration

An example of 2 dimensional cartesian coordinates:




```
struct Point {
        int x;
        int y;
};

struct Point p1, p2;
```

(We call Point is an user defined type)

Structures and Unions

Initialization

An example of 2 dimensional cartesian coordinates:

```
struct Point {
        int x;
        int y;
};

struct Point p1, p2;
p1.x = 0; p1.y = 0;
p2.x = 4; p2.y = 3;
```

```
struct Point p1 = {0, 0};
struct Point p2 = {4, 3};
```

Structures and Unions

Initialization

An example of Student type:

```
struct Student {
        char name[64];
        int roll_no;
        int age;
        float weight;
};
```

```
struct Student shyam = {
        "Shyam Karki",
        345,
        22,
        60.8f
};
```

```
struct Student shyam;
strcpy(shyam.name, “Shyam Karki”);
shyam.roll_no = 345;
shyam.age = 22;
shyam.weight = 60.8f;
```

Structures and Unions

Initialization

An example array of structures:

```
struct Point {
        int x;
        int y;
};

Point points[3] = {
        {0, 0},
        {1, 3},
        {6, 4},
};
```

Can these three points build a triangle ?
If so, can you calculate the centroid ?
How can you extend these array to represent a rectangle ?

Structures and Unions

Initialization

An example array of structures:

```
struct Point {
        int x;
        int y;
};
```

How can we define rectangle with the points ?

Structures and Unions

Initialization

An example of multiple Students:

```
struct Student {
        char name[64];
        int roll_no;
        int age;
        float weight;
};
```

```
Student students[2] = {
        {
                "Shyam Karki",
                345,
                22,
                60.8f
        },{
                "Phil Taylor",
                346,
                61,
                73.0f
        }
};
```

Structures and Unions

An example of multiple Students:

```
struct Student {
        char name[64];
        int roll_no;
        int age;
        float weight;
};

/** Find Oldest Student **/
Student students[MAX];
…
Student oldest = students[0];
for(int i=1;i<MAX;i++) {
        if(students[i].age > oldest.age)
                oldest = students[i];
}
```

Structures and Unions

Nested Structures

An example of geometric shape:

```
struct Point {
        int x;
        int y;
};

struct Rectangle {
        struct Point pt1;
        struct Point pt2;
};

struct Rectangle rect = {
        {0, 0},
        {4, 3}
};
```

Structures and Unions

Nested Structures

An example of person representation:

```
struct Person {
        Text additionalName;
        PostalAddress address;
        Organization affiliation;
        EducationalOrganizaton alumniOf;
        Text award;
        Date birthDate;
        Place birthPlace;
        Text familyName;
        Text givenName;
        QuantitativeValue weight;
};

struct Text {
        char content[100];
};
```

```
struct QuantitativeValue {
        Number maxValue;
        Number minValue;
        Text unitCode;
        Number value;
        Text unitText;
};
```

Structures and Unions

Unions

A union is a variable that may hold (at different times) objects of different types and sizes, with the compiler keeping track of size and alignment requirements.

Structures and Unions

Unions

Unions provide a way to manipulate different kinds of data in a single area of storage, without embedding any machine-dependent information in the program.

Structures and Unions

Unions

a union - a single variable that can legitimately hold any of one of several types.

Structures and Unions

Declaration

A person weight could either be an integer value or a float value.

```
union Weight {
        short iVal;
        float fVal;
};

union Weight personWeight;
```

A **personWeight** can hold either short integer or a float value at a time.

Structures and Unions

Declaration

A person weight could either be an integer value or a float value.

```
union Weight {
        short iVal;
        float fVal;
};

union Weight personWeight;
```

Same dot(.) operator(as in Structures) is used to access the member of a union.

personWeight.iVal and *personWeight.fVal*

Structures and Unions

Declaration

A Bank account with multiple type interests representation using union:

```
struct Saving {
    float interestRate;
    double balance;
};
```

```
struct Current {
    double balance;
};
```

```
struct Fixed {
    float interestRate;
    double balance;
    long matureDate;
};
```

```
struct Account {
    char name[64];
    char accountNumber[63];

    int type;
    union {
        struct Saving saving;
        struct Current current;
        struct Fixed fixed;
    } accountDetails;

};
```

Structures and Unions

Declaration

A Bank account with multiple type interests representation using union:

Why do we need a union for such Account representation ?

Structures and Unions

Enumerations

>> Enumeration represents a list of constant integer values.

>> Enumeration provides an alternative way to #define with the generated integer values for the programmer.

```
#define KEYWORD 01
#define EXTERNAL 02
#define STATIC 04
```

```
enum { KEYWORD = 01, EXTERNAL = 02, STATIC = 04 };
```

Structures and Unions

Enumerations

Enumeration usage :

```
enum Boolean { NO, YES };
```

```
enum WeekDays { SUNDAY = 1, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY };
```

```
enum Months { JAN = 1, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC };
```

Structures and Unions

Enumerations

Some Enumeration examples :

```c
#include<stdio.h>

enum Boolean {
    NO, YES
};

enum Month { JAN = 1, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC };

enum HairColor { BLACK, WHITE, BROWN, RED };

int main() {
    enum Boolean value = YES;
    printf("YES: %d\n", value);

    enum Month month = DEC;
    printf("Month: %d\n", month);
    return 0;
}
```

Thank you.

Questions ?