Lecture 21

Structs Cont.

QUIZ

سورة طه پارہ: 16

قَالَ رَبِّ اشْرَحْ لِيْ صَدْرِيْ ۙ ﴿٢٥﴾

[قَالَ: اس نے کہا] [رَبِّ: اے رب !] [اشْرَحْ: کھول دے] [لِيْ: میرے لیے] [صَدْرِيْ : میرا سینہ]

وَيَسِّرْ لِيْٓ اَمْرِيْ ۙ ﴿٢٦﴾

[وَيَسِّرْ: اور آسان کر دے] [لِيْ: میرے لیے] [اَمْرِيْ : میرا کام]

وَاحْلُلْ عُقْدَةً مِّنْ لِّسَانِيْ ۙ ﴿٢٧﴾

[وَاحْلُلْ: اور کھول دے] [عُقْدَةً: گرہ] [مِّنْ: سے] [لِّسَانِيْ: میری زبان]

يَفْقَهُوْا قَوْلِيْ ۖ ﴿٢٨﴾

[يَفْقَهُوْا: وہ سمجھ سکیں] [قَوْلِيْ : میری بات]

QUESTIONS / FEEDBACK / CONCERNS

UBS

ITU INFORMATION TECHNOLOGY UNIVERSITY

SE SEC A
SLIDE OF FAME

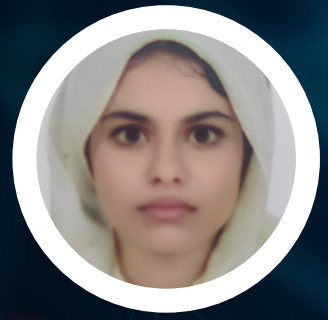
NO ONE
WEEK-1

Muhammad Daniyal Hammad (BSSE23046)
WEEK-2

Syed Hashim Abbas (BSSE23084)
WEEK-3

Umar Ahmad (BSSE23032)
WEEK-4

Umar Ahmad (BSSE23032)
WEEK-5



Fatima Noorulain BSSE23003
WEEK-6

Umar Ahmad (BSSE23032)
WEEK-7

YOUR NAME
WEEK-8

YOUR NAME
WEEK-9

YOUR NAME
WEEK-10

YOUR NAME
WEEK-11

YOUR NAME
WEEK-12

YOUR NAME
WEEK-13

YOUR NAME
WEEK-14

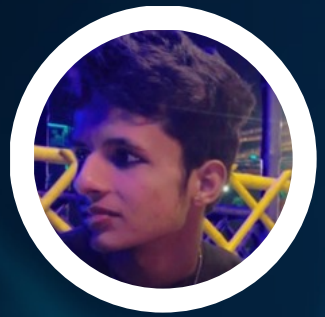
YOUR NAME
WEEK-15

UBS

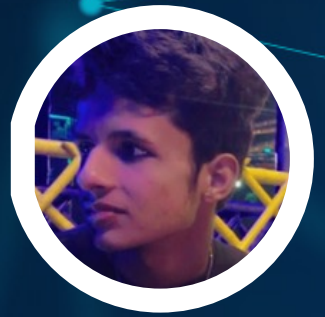
ITU INFORMATION TECHNOLOGY UNIVERSITY

SE SEC B
SLIDE OF FAME

Muhammad Mukarram
BSSE23029
WEEK-1



Muhammad Abdullah
(BSSE23087)
WEEK-2



Muhammad Abdullah
(BSSE23087)
WEEK-3

Fasiha Rohail
(BSSE23041)
WEEK-4

Muhammad Abdullah
(BSSE23087)
WEEK-5

Hazira Azam
BSSE23019
WEEK-6

Jamshaid Ahmed
BSSE23012
WEEK-7

YOUR NAME
WEEK-8

YOUR NAME
WEEK-9

YOUR NAME
WEEK-10

YOUR NAME
WEEK-11

YOUR NAME
WEEK-12

YOUR NAME
WEEK-13

YOUR NAME
WEEK-14

YOUR NAME
WEEK-15

UBS

ITU INFORMATION TECHNOLOGY UNIVERSITY

RECAP

GitHub

Tools (Cygwin, IDE, GitHub)

Approach towards a word problem

Flowcharts

Flowcharts Advantages & Disadvantages

Algorithms

Pseudocode

Numbers Systems (Decimal, Binary, Octal & Hexadecimal)

Ten’s Complement

Twos Complement

main function

Stream in and stream out operators

if else

Functions

Data Types

Arithmetic Operators

Relational Operators

Loops (While, for , do while)

Nested Loops

Switch cases

RECAP

Function Overloading

Scope of variables

Function Prototype and Definition

Default Value in parameters of functions

Parameters by value vs Parameters by Reference

Recursion

Arrays

2D Arrays / Multi Dimensional Arrays

Pointers

Structs

Filing

DMA

Template Functions

```
#include <iostream>
using namespace

template <typename T>
T myMax(T x, T y)
{
   return (x > y)? x: y;
}

int main()
{
  cout << myMax<int>(3, 7) << endl;  // Call myMax for int
  cout << myMax<double>(3.0, 7.0) << endl; // call myMax for double
  cout << myMax<char>('g', 'e') << endl;   // call myMax for char

  return 0;
}
```

https://www.geeksforgeeks.org/templates-cpp/

Template Functions

```
#include <iostream>
using namespace std;

template <class T>
void bubbleSort(T a[], int n) {
    for (int i = 0; i < n - 1; i++)
        for (int j = n - 1; i < j; j--)
            if (a[j] < a[j - 1])
            swap(a[j], a[j - 1]);
}

int main() {
    int a[5] = {10, 50, 30, 40, 20};
    int n = sizeof(a) / sizeof(a[0]);

    // calls template function
    bubbleSort<int>(a, n);

    cout << " Sorted array : ";
    for (int i = 0; i < n; i++)
        cout << a[i] << " ";
    cout << endl;
    return 0;
}
```

https://www.geeksforgeeks.org/templates-cpp/

WHAT ARE STRUCTS?

Struct

```
struct Person
{
    char name[50];
    int age;
    float salary;
};
```

https://www.programiz.com/cpp-programming/structure

Struct

```cpp
#include <iostream>
using namespace std;

struct Person
{
    char name[50];
    int age;
    float salary;
};

int main()
{
    Person p1;

    cout << "Enter Full name: ";
    cin.get(p1.name, 50);
    cout << "Enter age: ";
    cin >> p1.age;
    cout << "Enter salary: ";
    cin >> p1.salary;

    cout << "\nDisplaying Information." << endl;
    cout << "Name: " << p1.name << endl;
    cout <<"Age: " << p1.age << endl;
    cout << "Salary: " << p1.salary;

    return 0;
}
```

```
#include <iostream>
#include <iomanip>
#include <string>
using namespace std;
struct Representative  // Defining struct Representative
{
  string name;         // Name of a representative.
  double sales;        // Sales per month.
};
inline void print( const Representative& v)
{
   cout << fixed << setprecision(2)
        << left  << setw(20) << v.name
        << right << setw(10) << v.sales << endl;
}
int main()
{
   Representative rita, john;
   rita.name    = "Strom, Rita";
   rita.sales   = 37000.37;
   john.name    = "Quick, John";
   john.sales = 23001.23;

   rita.sales += 1700.11;              // More Sales
   cout << "  Representative             Sales\n"
        << "------------------------------" << endl;
   print( rita);
   print( john);
   cout << "\nTotal of sales: "
        << rita.sales + john.sales << endl;
   Representative *ptr = &john;          // Pointer ptr.
                                         // Who gets the
   if( john.sales < rita.sales)          // most sales?
     ptr = &rita;
   cout << "\nSalesman of the month: "
        << ptr->name << endl;    // Representative's name
                                 // pointed to by ptr.
   return 0;
}
```

STRUCTS

STRUCTS

```
struct tm
{
   int tm_sec;          // 0 - 59(60)
   int tm_min;          // 0 - 59
   int tm_hour;         // 0 - 23
   int tm_mday;         // Day of month: 1 - 31
   int tm_mon;          // Month: 0 - 11 (January == 0)
   int tm_year;         // Years since 1900 (Year - 1900)
   int tm_wday;         // Weekday: 0 - 6 (Sunday == 0)
   int tm_yday;         // Day of year: 0 - 365
   int tm_isdst;        // Flag for summer-time
};
```

Image Reference : A Complete Guide to Programming in C++

STRUCTS

```
#include <iostream>
#include <ctime>
using namespace std;

   struct tm *ptr;            // Pointer to struct tm.
   time_t sec;                // For seconds.
    . . .
   time(&sec);                // To get the present time.
   ptr = localtime(&sec);     // To initialize a struct of
                              // type tm and return a
                              // pointer to it.

   cout << "Today is the "       << ptr->tm_yday + 1
        << ". day of the year " << ptr->tm_year
        << endl;
    . . .
```

Example `struct` Declaration

```
struct StudentStruct
{
        int studentID;
        string name;
   short year;
   double gpa;
};
```

structure name

structure members

Note the required ;

`struct` Declaration Notes

- `struct` names commonly begin with an uppercase letter, *because they are a type*
- Multiple fields of same type can be declared in a comma-separated list

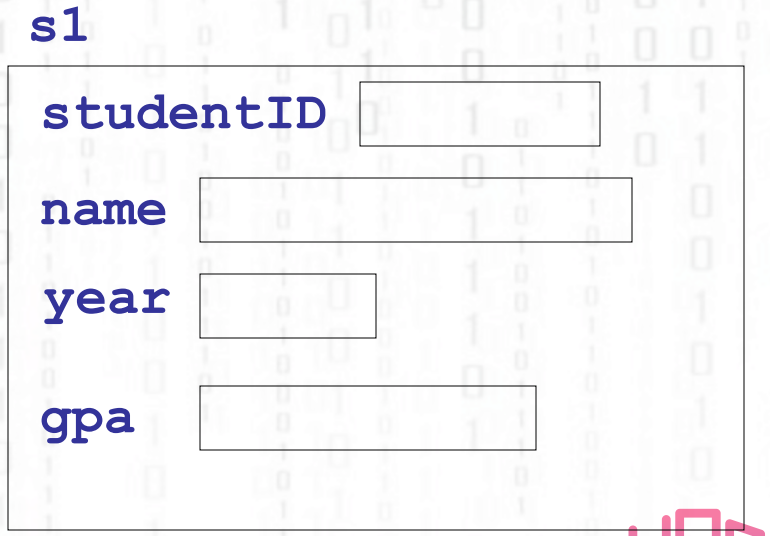
```
string name,address;
```

- Fields in a structure are all public by default

Defining Structure Variables

- A **`struct`** declaration does not allocate memory or create variables, *because they are a type*
- To define variables, use structure tag as the type name

```
StudentStruct s1;
```



Accessing Structure Members

- Use the name of the struct variable
  the name of the member
  separated by a dot .

```
getline(cin, s1.name);
cin >> s1.studentID;
s1.gpa = 3.75;
```

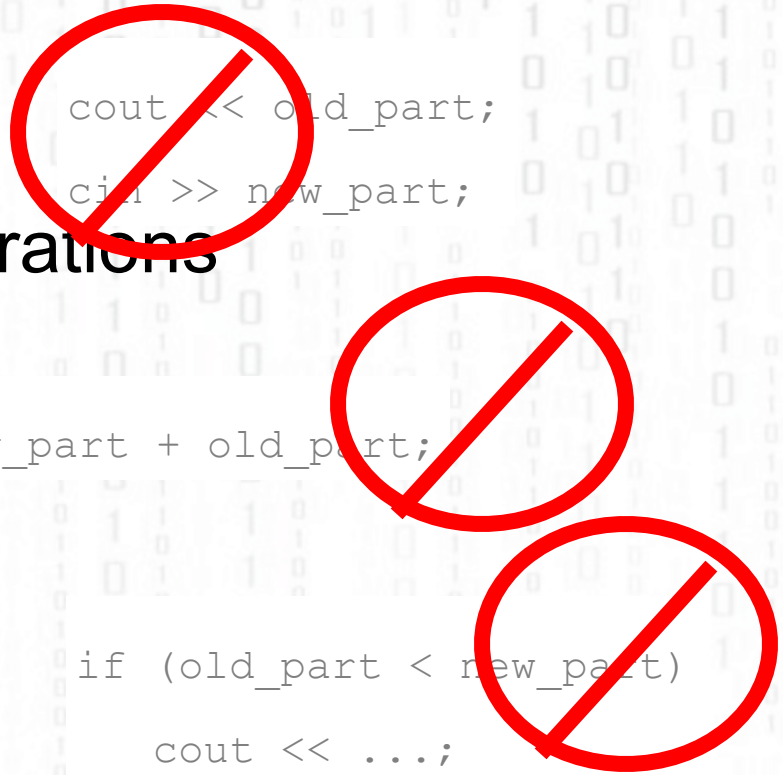
- The dot is called the member selector

Aggregate Operations with Structures

- Recall that arrays had no whole array operations (except passing reference to parameter)
- Structures DO have aggregate operators
  - assignment statement =
  - parameter (value or reference)
  - return a structure as a function type

Aggregate Operations with Structures

- Limitations on aggregate operations
  - no I/O
  - no arithmetic operations
  - no comparisons



Aggregate Operations with Structures

- `struct` variables must be initialized, compared, written one member at a time.

Displaying `struct` Members

To display the contents of a **`struct`** variable, you must display each field or member separately, using the dot operator

Wrong:

```
cout << s1; // won't work!
```

Correct:

```
cout << s1.studentID << endl;
cout << s1.name << endl;
cout << s1.year << endl;
cout << s1.gpa;
```

Comparing `struct` Members

- Similar to displaying a **`struct`**, you cannot compare two **`struct`** variables directly:

```
if (s1 >= s2) // won't work!
```

- Instead, compare member variables:

```
if (s1.gpa >= s2.gpa) // better
```

Initializing a Structure

Structure members should not be initialized in the structure declaration, because no memory has been allocated yet

```
struct Student          // Illegal
{                       // initialization
   int studentID = 1145;
   string name = "Alex";
   short year = 1;
   float gpa = 2.95;
};
```

Initializing a Structure (continued)

- Structure members are initialized at the time a structure variable is created
- You can initialize a structure variable's members with either
  - an initialization list
  - a constructor

Initialization List Example

Structure Declaration

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

Structure Variable

box


```
Dimensions box = {12,6,h};
```

Using a Constructor to Initialize Structure Members

- This is similar to a constructor for a class:
  - the name is the same as the name of the struct
  - it has no return type
  - it is used to initialize data members
- It is normally written inside the **`struct`** declaration

A Structure with a Constructor

```
struct Dimensions
{
   int length,
       width,
       height;

   // Constructor
   Dimensions(int L, int W, int H)
   {length = L; width = W; height = H;}
};
```

Nested Structures

A structure can have another structure as a member.

```
struct PersonInfo
{   string name,
           address,
           city;
};
struct Student
{   int            studentID;
    PersonInfo     pData;
    short          year;
    double         gpa;
};
```

Members of Nested Structures

Use the dot operator multiple times to access fields of nested structures

```
Student s5;
s5.pData.name = "Joanne";
s5.pData.city = "Tulsa";
```

Reference the nested structure's fields by the member variable name, not by the structure name

```
s5.PersonInfo.name = "Joanne"; //no!
```

Structures as Function Arguments

- You may pass *members* of **`struct`** variables to functions

```
computeGPA(s1.gpa);
```

- You may pass entire **`struct`** variables to functions by value or by reference

```
void CopyPart (PartStruct partOrig, PartStruct & partNew);
```

Notes on Passing Structures

- Using a value parameter for structure can slow down a program and waste space
- Using a reference parameter speeds up program execution, but it allows the function to modify data in the structure
- To save space and time while protecting structure data that should not be changed, use a **`const`** reference parameter

```
void showData(const StudentStruct &s)
                                  // header
```

Returning a Structure from a Function

- A function can return a **`struct`**

```
StudentStruct getStuData();  // prototype
s1 = getStuData();           // call
```

- The function must define a local structure variable
  - for internal use
  - to use with **`return`** statement

Returning a Structure Example

```
Student getStuData()
{ Student s;     // local variable
  cin >> s.studentID;
  cin.ignore();
  getline(cin, s.pData.name);
  getline(cin, s.pData.address);
  getline(cin, s.pData.city);
  cin >> s.year;
  cin >> s.gpa;
  return s;
}
```