Data Types in C++

- Scalar (or Basic) Data Types (atomic values)
    - Arithmetic types
        - Integers
            - short, int, long
            - char, bool
        - Floating points
            - float, double, long double
- Composite (or Aggregate) Types:
    - Arrays: ordered sequence of values of the same type
    - Structures: named components of various types

Structures

- Used to represent a relationship between values of different types
- Example: student
    - ID Number
    - Name
    - Age
    - Major
    - Address
- (The values are related because they belong to the same student)

- Define the student as a struct in C++:

```
struct Student {
     int idNumber;
     string name;
     int age;
     string major;
};
```

- NOTE: Semicolon after the last brace!
- A struct is a data type, by convention the name is capitalized.
- The components are called “members” (or “fields”).

Structures

- So far we have defined a new data type, but we haven’t defined any variables of that type.
- To define a variable of type Student:

```
            Student csStudent;
```

- Can define multiple variables of type Student:

```
        Student student1, student2, gradStudent;
```

- Each one has its own set of the member variables in the Student data type
- Each variable of type student has its own set of the member variables from the Student data type

```
Student student1, student2;

Student1 has its own idNumber, name, age, major
Student2 has its own idNumber, name, age, major
```

Accessing Structure Members

- Use dot notation to access members of a struct variable:

```
student1.age = 18;
student2.idNumber = 123456;
cin >> gradStudent.name;
gradStudent.major = "Rocket Science ";
```

- Member variables of structures can be used just like regular variables of the same type.

```
student1.age++; //happy birthday
myFunc(student2.idNumber);
if (student1.age==student2.age) {
         ...
     }
```

Structures: operations

- Valid operations over entire structs:
    - assignment: `student1 = student2;`
    - function call: `myFunc(gradStudent,x);`

- Invalid operations over structs:
    - comparison: `student1 == student2`
    - output: `cout << student1;`
    - input: `cin >> student2;`
    - Must do these member by member

Structures: output

- Output the members one at a time:

```
cout << student1.idNumber << " ";
cout << student1.name << " ";
cout << student1.age << " ";
cout << student1.major << endl;

Output:
11122 Chris Johnson 19 Football
```

- Comparing two structs:

```
if (student1.idNumber == student2.idNumber &&
     student1.name == student2.name &&
     student1.age == student2.age &&
     student1.major == student2.major)
...
```

Initializing structures

- Struct variable can be initialized when it is defined:

```
Student student1 = {123456, "John Smith",22, "Math"};
```

- Must give values in order of the struct declaration.
- Can NOT initialize members in structure declaration, only variable definition:

```
struct StudentA {
int id = 123456; //ILLEGAL
string name = “John Smith”; //ILLEGAL
}
```

Nested Structures

- You can nest one structure inside another.

```
struct Address {
     string street;
     string city;
     string state;
     int zip;
};

struct Student {
     int idNumber;
     string name;
     Address homeAddress;
 };
```

- Use dot operator multiple times to get into the nested structure:

```
Student student1;
student1.name = "Bob Lambert";
student1.homeAddress.city = "San Angelo";
student1.homeAddress.state = "TX";
```

- Or set up address structure separately:

```
Address a1;
a1.street = "101 Main St. ";
a1.city = "San Angelo";
a1.state = "TX";
a1.zip = 76903;
student1.name = "Bob Lambert";
student1.homeAddress = a1;
```

Example – simple use of a struct

```
// Shows simple use of a struct.
// Husain Gholoom

//================================================= includes
#include <iostream>
using namespace std;

//========================================= define new types
struct Time {
    int hours;
    int minutes;
    int seconds;
};

//===================================================== main
int main() {

    Time t;
    int toSeconds ;
    cout << "Enter No Of Hours  :  " ;
    cin >> t.hours ;
    cout << "Enter No Of Minutes: " ;
    cin >> t.minutes ;
    cout << "Enter No Of Seconds : " ;
    cin >> t.seconds ;

    toSeconds = 3600*t.hours + 60*t.minutes + t.seconds ;
    cout <<endl<<endl;
    cout << "Total seconds: " << toSeconds ;

    return 0;
}
```

Sample Output

```
Enter No Of Hours  :  1
Enter No Of Minutes: 1
Enter No Of Seconds : 1


Total seconds: 3661
```

Example –Passing Structure to a Function By Value

```
/*
 *    StructFunction.cpp
 *
 *    Author: Husain Gholoom
 *    Pass By Value
 */


#include<iostream>
#include<string>

#include<iomanip>
using namespace std;

struct Records
{
   string  Name ;
   int Salary;
   int Deductions;

};


void displayInfo(Records file);

int main()
{
      Records employee;
      employee.Name="Allison";
      employee.Salary= 2750;
      employee.Deductions = 350;
      displayInfo(employee);
  return 0;
}



void displayInfo(Records file )
      {
               cout << "Here are the records you entered:\n\n";
               cout << "NAME: " << file.Name << endl;
               cout << setw(6) << "Salary: " << file.Salary << endl;
               cout << setw(6) << "Deductions: " << file.Deductions << endl;
               cout << endl;
               cout << setw(6) << "Net Salary: " << file.Salary - file.Deductions << endl;


      }
```

Sample Output

```
Here are the records you entered:

NAME: Allison
Salary: 2750
Deductions: 350

Net Salary: 2400
```

What Happens IF you change the value of file.Name ??????????

```
void displayInfo(Records file )

      {
          file.Name = "Ray";    ?????????
         cout << "Here are the records you entered:\n\n";
         cout << "NAME: " << file.Name << endl;
         cout << setw(6) << "Salary: " << file.Salary << endl;
         cout << setw(6) << "Deductions: " << file.Deductions << endl;
         cout << endl;
         cout << setw(6) << "Net Salary: " << file.Salary - file.Deductions << endl;


      }
```

Example –Passing Structure to a Function By Reference

```
/*
 *    StructFunction.cpp
 *
 *    Author: Husain Gholoom
 *    Pass By Reference
 */


#include<iostream>
#include<string>
#include<iomanip>
using namespace std;

struct Records
{
   string  Name ;
   int Salary;
   int Deductions;

};


void getInfo(Records &file);
void displayInfo(Records &file);

int main()     {
      Records employee;
      getInfo(employee);
      displayInfo(employee);
  return 0;
}


void getInfo( Records &file)     {

      cout << "Name: ";
      cin >> file.Name;
      cout << setw(6) << "Salary: ";
      cin >> file.Salary;
      cout << "Deductions: ";
      cin >> file.Deductions;
      cout << endl;

}
```

```
void displayInfo(Records &file )    {

     cout << "Here are the Record you Entered:\n\n";
     cout << "NAME: " << file.Name << endl;
     cout << setw(6) << "Salary: " << file.Salary << endl;
     cout << setw(6) << "Deductions: " << file.Deductions << endl;
     cout << endl;
     cout << setw(6) << "Net Salary: " << file.Salary - file.Deductions << endl;


}
```

Sample Output

```
Name: Allison
Salary: 2850
Deductions: 375

Here are the Record you Entered:

NAME: Allison
Salary: 2850
Deductions: 375

Net Salary: 2475
```

Example - Nested Structs

```
// Nested Structures
// Husain Gholoom

#include <iostream>
using namespace std;

struct GradeRec
{
	float percent;
	char grade;
};

struct StudentRec
{
	string lastName;
	string firstName;
	int age;
	GradeRec courseGrade;
};

int  main()
{
	StudentRec student;
	cout << "Enter first name: ";
	cin >> student.firstName;
	cout << "Enter last name: ";
	cin >> student.lastName;
	cout << "Enter age: ";
	cin >> student.age;
	cout << "Enter overall percent: ";
	cin >> student.courseGrade.percent;
	if(student.courseGrade.percent >= 90)
		{
			student.courseGrade.grade = 'A';
		}
	else if(student.courseGrade.percent >= 80)
		{
			student.courseGrade.grade = 'B';
		}

	 else if(student.courseGrade.percent >= 70)
		{
			student.courseGrade.grade = 'C';
		}
	else if(student.courseGrade.percent >= 60)
		{
			student.courseGrade.grade = 'D';
		}

	else
		{
			student.courseGrade.grade = 'F';
		}


	cout << "\n\nHello " << student.firstName << ' ' << student.lastName ;

	cout << "\nCongratulations. You are   " << student.age  << ".\n";

	cout << "Your overall percent score is "
		<< student.courseGrade.percent << " Your earned a grade of    "
		<< student.courseGrade.grade;
}
```

Sample Output

```
Enter first name: Allison
Enter last name: Ray
Enter age: 21
Enter overall percent: 98


Hello Allison Ray.
Congratulations.  You are 21.
Your overall percent score is 98.  You earned a grade of A
```

Arrays of Structures

- Define the student as a struct :

```
struct Student {
    int idNumber;
    string name;
    int age;
    string major;
};
```

- You can store values of structure types in arrays.

```
Student roster[40]; //holds 40 Student structs
```

- Each student is accessible via the subscript notation.

```
roster[0] = student1;
```

- Members of structure accessible via dot notation

```
cout << roster[0].name << endl;
```

- Arrays processed in loops:

```
Student roster[40];

//input
for (int i=0; i<40; i++) {
     cout << "Enter the name, age, idNumber and "
          << "major of the next student: \n";
     cin >> roster[i].name >> roster[i].age
          >> roster[i].idNumber >> roster[i].major;
}

//output all the id numbers and names
for (int i=0; i<40; i++) {
cout << roster[i].idNumber << endl;
cout << roster[i].name << endl;
}
```

OR

```
Student roster[40] = {
     {123456,"John Smith",22, "Math"} ,
     {444555,"Lisa Simpson",18, "Biology"},
     {999999,"Tony Jackson",25, "Physics"},
     {887766,"Melissa Brown",20, "Engineering"}
   };
```

Example Using Struct – Arrays and Functions

```
/*
 * StructArrays.cpp
 *
 *  Author: Husain Gholoom
 */


#include<iostream>
#include<string>

#include<iomanip>
using namespace std;

struct Records
{
   string Name;
   int Salary;
   int Deductions;

};

void getInfo(Records file[], int SIZE);
void displayInfo(Records file[], int SIZE);

const int SIZE = 5;
int numRec = 0;

int main()
{

   // Declare an array of objects.
   Records files[SIZE];

   cout << "Max Number of Records  you can enter is " << SIZE << ".\n";
   cout << "\nHow many records do you want to enter?: ";
   cin >> numRec;
   cout << endl;

   if (numRec <= SIZE)
   {
      getInfo(files, SIZE);
      cout << endl;
   }
   else
   {
      cout << "You can only enter less than " << SIZE << " records!\n\n";
      exit(0);
   }

   displayInfo(files, SIZE);
   return 0;             }

void getInfo( Records file[], int SIZE)
{
   for (int i = 0; i < numRec; i++)
   {
      cout << "Name: ";
      cin >> file[i].Name;
      cout << setw(6) << "Salary: ";
      cin >> file[i].Salary;
      cout << "Deductions: ";
      cin >> file[i].Deductions;

      cout << endl;
   }
}


void displayInfo(Records file[], int SIZE)
{
   int count = 1;
   cout << "Here are the records you entered:\n\n";

   for (int i = 0; i < numRec; i++)
      {
         cout << "Record #  " << count << ":\n";
         cout << "NAME: " << file[i].Name << endl;
         cout << setw(6) << "Salary: " << file[i].Salary << endl;
         cout << setw(6) << "Deductions: " << file[i].Deductions << endl;
         cout << endl;
         cout << setw(6) << "Net Salary: " << file[i].Salary -
                    file[i].Deductions << endl;
         count++;
      }
}
```

Sample Output

```
Max Number of Records  you can enter is 5.

How many records do you want to enter?: 1

Name: Allison
Salary: 2780
Deductions: 350


Here are the records you entered:

Record #  1:
NAME: Allison
Salary: 2780
Deductions: 350

Net Salary: 2430
```

Structures and files

```
#include <iostream>
#include <fstream>

using namespace std;

struct StudentRecord {
        string sname;
        int finalGrade;
};

char calcAward(StudentRecord rec);

int main() {

        ifstream
        ifs("studentRecords.txt");

        if (ifs.fail()) {
                cout << "Error opening student records file
                           (studentRecords.txt)"
                                << endl;
                return 1;
        }

        StudentRecord record;

        ifs >> record.sname;
        ifs >> record.finalGrade;
        while (!ifs.eof()) {

                cout << "\n\nStudent  :   " << record.sname
                        << "\nrecord  ... processing ... processing" << endl;

                char grade = calcAward(record);
                cout << record.sname << "   got    "
         <<  record.finalGrade
                   " points  which is:   " << grade;

                ifs >> record.sname;
                ifs >> record.finalGrade;
        }
        cout << "\n\n";
 return 0;
}

char calcAward(StudentRecord rec) {
        char grade = 'N';
        if (rec.finalGrade >= 90) {
                grade = 'A';
        } else if (rec.finalGrade >= 80) {
                grade = 'B';
        } else if (rec.finalGrade >= 70) {
                grade = 'C';
        } else if (rec.finalGrade >= 60) {
                grade = 'D';
        } else
                grade = 'F';

        return grade;
}
```

For Example :

Assume that the studentRecords.txt has the following Data :

```
Allison  95
Ray   90
Sam 28
```

Sample run

```
Student  :  Allison
Record  … processing … processing
Allison  Got  95  points   which  is : A

Student  :  Ray
Record  … processing … processing
Ray  Got  90  points   which  is : A

Student  :  Sam
Record  … processing … processing
Sam  Got  28  points   which  is : F
```