Attendance record

- To check what devices students are using to join class
- To check attendance

https://forms.gle/XcQWvDuaCDxjwLeR8

Start class: **3:05pm**

DATA STRUCTURE AND PROGRAMMING I

Chapter 6- Data Structure (enumeration and structure)

Prepared by: Dr. VALY Dona and Mr. NOU Sotheany

Modified and Lectured by: Mr. BOU Channa

Lecture overview

❑ Overall lectures

1. Introduction to algorithm
2. Basic data types and statements
3. Control structures and Loop
4. Array
5. Sub-program
6. ***Data structure***

OBJECTIVE

- Understand and know how to define new variable with predefined values

- Understand and know how to create new data type that consist of many other variables

Defining variable with predefined values

- Introduction

Enumeration

Enumeration is a new data type and consisting a set of values defined by a programmer during creation.

Defining variable with predefined values

❑ Enumeration

- Enumeration is a new data type and consisting a set of values defined by a programmer during creation.
- The value can be name, month, day, color, or anything that could make a program code easy to read and maintain.
- **Example:**
  - Month of the year: `(January, February, March,..., December)`
  - Car's brand: `(Lexus, Mercedes Benz,...)`
  - Marital status: `(Single, Divorce, Married,...)`

Defining variable with predefined values

❑ Declaration

- To create an enumeration type:

  enum identifierName **{**value1, value2, …, valueN**}**

Examples:

enum day {Monday, Tuesday,…, Sunday}

enum color {red, blue, white, black}

Defining variable with predefined values

❑ Variable of enumerated type

- Declaration:
  - `Var identifier : name of enumerated type`
  - `Var d: day`
- Usage:
  - Can only assign a constant to a variable of the enumeration or
  - Compare a variable of the type
- Example:

```
enum day {Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday}
Var d: day
d ← Monday
if (d == Wednesday) then
        …
end if
```

Defining variable with predefined values

❑ Conversion of enumerated type

- The value of enumerated type is considered as integer when display.
  - It is in the order start from 0
- Example:

```
var d: day
for(d ← Monday; d<=Sunday; d ← d + 1) do
    write(d)
end for
```

Defining variable with predefined values

❑ Example: Using enumeration type

```
enum color{black, white, red, yellow, blue}
Procedure display_color(c: color)
        switch(c) do
                black: write(“Black color”)
                white: write(“white color”)
                red: write(“red color”)
                yellow: write(“blue color”)
        end switch
End procedure
```

```
Var tmp: color
begin
        tmp ← black
        display_color(tmp)
end
```

Defining variable with predefined values

- Example 1

```c
#include<stdio.h>

enum day{
    Monday, Tuesday, Wednesday,
    Thursday, Friday, Saturday, Sunday
};
enum color{
    Red, Blue, Green, Black, White
};

main(){

    enum day d1;
    enum color c1;

    c1 = White;
    d1 = Monday;

    printf("Day: %d\n", d1);
    printf("Color: %d\n", c1);

}
```

Defining variable with predefined values

- Example 2

```c
#include<stdio.h>

enum day{
    Monday, Tuesday, Wednesday,
    Thursday, Friday, Saturday, Sunday
};
enum color{
    Red, Blue, Green, Black, White
};

char myday[][20]={
    "Monday", "Tuesday", "Wednesday",
    "Thursday", "Friday", "Saturday", "Sunday"
};

main(){

    enum day d1;
    enum color c1;

    c1 = White;
    d1 = Monday;

    printf("%d\n", d1);
    printf("%s", myday[d1]);
}
```

Structure

- Allow programmers to create a new data type which consists of other defined data types (int, float, character, string, …)

- It could consist of the other structure or itself as well!

Create new data type with structure

❑ Structure

- **Problem**: Information of a student (surname, name, age, sex). Suppose that we have 100 students, which data structure can be used to store these data?
- **Solution**: Array for each attribute

```
const (N : integer) ← 100
var name[N], surname[N] : sequence of character
Var age[N] : integer
Var sex[N] : char
```

- **Inconvenient**: The information are scattered from each other in many arrays.
  - We are difficult to identify all information of a student.

Create new data type with structure

❑ Definition

- **Structure** a user-defined data type which allows us to combine data of different types together. Structure helps to construct a complex data type which is more meaningful

- Each element/variable/data in the structure consists in the structure is called
  - "**field**" or
  - "**attribute**" or
  - "**member**"

Create new data type with structure

❑ Declaration

- Before we can create a variable of a type structure, we need to create the structure type first.
- **Syntax to create a structure type:**

```
struct identifier
        attr1: type
        attr2: type
        …
        attrN: type
end struct
```

- After we define a structure type, we can declare variables of structure type

Must have the same name

```
Var varName: identifier
```

Create new data type with structure

❑ Example

- Suppose we create a structure for storing a student information

```
struct student
        surname, name: string
        age: integer
        sex: char
end struct
Var s: student
```

Syntax in C program:

- Example:

```
struct StudentInfo{
    int Id;
    int age;
    char Gender;
    double CGA;
};
```

The "StudentInfo" structure has 4 members of different types.

- Example:

```
struct StudentGrade{
    char Name[15];
    char Course[9];
    int Lab[5];
    int Homework[3];
    int Exam[2];
};
```

The "StudentGrade" structure has 5 members of different array types.

struct StudentInfo s;

Create new data type with structure

❑ How to access to element in the structure

- The elements of array are accessed by index ***while*** the elements of structure are accessed by `.` (period)

- **Syntax**:

```
structureVariableName.attributeName
```

Create new data type with structure

❑ Example

- Suppose we create a structure for storing a student

```
struct student
        surname, name: string
        age: integer
        sex: char
end struct

Var s: student
s.name ← “Sok”
read(s.surname)
s.age ← 20
s.age ← s.age * 2
```

Create new data type with structure

❑ Example

- Suppose we create a structure for storing information of 100 students

```
struct student
        surname, name: sequence of character
        age: integer
        sex: character
end struct

Var s[100]: student
For(i ← 0; i<100; i++) do
        read(s[i].surname)
        read(s[i].name)
        read(s[i].sex)
        read(s[i].age)
End for
```

Break
Back: 4:30pm

Using structure in sub-program

❑ Example

- Create a subprogram to find the different age between two students

```
function diffAge(s1: student, s2: student): integer
begin function
   if (s1.age > s2.age) then
      return (s1.age – s2.age)
   else
      return (s2.age – s1.age)
   end if
end function
```

Structure in C programming

❑ Examples

```
struct Student{
   char stud_name[30];------- 30 bytes
   int roll_number;--------- 02 bytes
   float percentage;-------- 04 bytes
};                          sum = 36 bytes
```

Here the variable of **Student** structure is allocated with 36 bytes of memory.

```c
#include<stdio.h>

struct Point
{
    int x, y;
};

int main()
{
    struct Point p1 = {0, 1};

    // Accesing members of point p1
    p1.x = 20;
    printf ("x = %d, y = %d", p1.x, p1.y);
```

Structure in C programming

❑ Using **typedef** to rename name of data type

- Create a structure

```
struct myStruct{
    int one;
    int two;
};
```

- Then to create a variable:

struct myStruct ms;

- Create a structure and give it a short name

```
typedef struct{
    int one;
    int two;
}myStruct;
```

- Then to create a variable, we don't need to use the keyword struct

myStruct ms;

Create new data type with structure

❑ Using **typedef**

```
typedef struct S {
    int x;
} S;
```

```
S s1;
```

```
struct S {
    int x;
};

typedef struct S S;
```

```
S s1;
```

Examples

enumeration test1.c

```c
#include<stdio.h>


enum Color{
    red, blue, black, white, violet, yellow
};

enum account{
    gold, VIP, normal, silver
};

main(){

    enum Color c1;
    enum account acc1;

    c1 = violet;
    printf("%d\n\n" , c1);

//    for(int k=0; k<10; k=k+1){
//
//    }

    for(c1=red; c1<=yellow; c1=c1+1){
        //printf("%d ",c1);
        if(c1==red){
            printf("Dress red color\n");
            //activate your function
        }else if(c1==yellow){
            printf("Dress yellow color\n");
        }
    }
}
```

```
"C:\!Data\AIgo2021\LabCTest\Datastructure\enumeration test

4

Dress red color
Dress yellow color

Process returned 0
Press any key to co
```

Examples

eration test1.c | structure test1.c

```c
#include<stdio.h>

//Define a structure named Student (id, name, age)
struct Student{
   int ID;
   char name[30];
   char major[30];
   int age;
   int year;
};

main(){

    struct Student s1;

    printf("Enter student ID: ");
    scanf("%d", &s1.ID);

    printf("Enter your name: ");
    scanf("%s", &s1.name);

    printf("\n\t*** Summary student information:\n");
    printf("Name: %s\n", s1.name);
    printf("Student ID: %d\n", s1.ID);


}
```

Select "C:\!Data\Algo2021\LabCTest\Datastructure\structure test1.exe"

```
Enter student ID: 123
Enter your name: Dara

        *** Summary student information:
Name: Dara
Student ID: 123

Process returned 0 (0x0)   execution time
Press any key to continue.
```

Examples

neration test1.c | *structure test1.c

```c
#include<stdio.h>

//Define a structure named Student (id, name, age)
struct Student{
    int ID;
    char name[30];
    char major[30];
    int age;
    int year;
};

main(){
    struct Student studentList[10];
    //studentList[0].name

    int a;
    a=5;
    a=10;

    while(2>0){
        struct Student s1;

        printf("\nEnter student ID: ");
        scanf("%d", &s1.ID);

        printf("Enter your name: ");
        scanf("%s", &s1.name);

        printf("\n\t*** Summary student information:\n");
        printf("Name: %s\n", s1.name);
        printf("Student ID: %.3d\n", s1.ID);

    }

    //cin>>s1.ID;   C++
}
```

"C:\!Data\Algo2021\LabCTest\Datastructure\structure test1.exe"

```
Enter student ID: 23
Enter your name: Dara

        *** Summary student information:
Name: Dara
Student ID: 023

Enter student ID: 123
Enter your name: Dara

        *** Summary student information:
Name: Dara
Student ID: 123

Enter student ID: 1
Enter your name: Jack
```

Examples

eration test1.c | *structure test1.c

```c
#include<stdio.h>

//Define a structure named Student (id, name, age)
struct Student{
    int ID;
    char name[30];
    char major[30];
    int age;
    int year;
};

main(){
    struct Student st[5];

    for(int k=0; k<5; k=k+1){
        printf("\n\n***Info student #%d\n", k+1);
        printf("Enter your name: ");
        scanf("%s", &st[k].name);

        printf("Enter your student ID: ");
        scanf("%d", &st[k].ID);

        printf("Enter your age: ");
        scanf("%d", &st[k].age);
    }

    printf("\n\n**** Student info's summary\n");
    for(int p=0; p<5; p=p+1){
        printf("%.3d\t%12s\t%d\n", st[p].ID, st[p].name, st[p].age);

    }

//    while(2>0){
//        struct Student s1;
```

```
Select "C:\!Data\Algo2021\LabCTest\Datastructure\structure test1.exe"
Enter your student ID: 123
Enter your age: 90


***Info student #5
Enter your name: Seyha
Enter your student ID: 4
Enter your age: 32


**** Student info's summary
012                SOk     20
001               Dara     30
321                Sao     30
123            Chettra     90
004              Seyha     32
```

An example to create an **Enumeration** and a **Structure** in C programming

```c
#include<stdio.h>
#include<string.h>

enum Day{Mon, Tue, Wed, Thu};
struct Student{
    int id;
    char name[100];
    char gender;
};
//typedef struct Student stud;

typedef struct Data{
    char model[100];
    int producedYear

}Data;

int main(){

    enum Day d;
    struct Student s1;
    Data data;

    d=Thu;
}
```

An example to create **Structure** (normal and nested structure) and using it in C programming

```c
#include<stdio.h>
struct Employee{
    char name[20];
    int id;
    float salary;
    struct date{
        int day;
        int month;
        int year;
    }date;
};
typedef struct Employee MyEmployee;

main(){
    MyEmployee emp={"John",100, 500, {12,3,2000}};
    printf("Employee name: %s\n", emp.name);
    printf("Employee ID: %d\n", emp.id);
    printf("Employee salary: %.2f\n", emp.salary);
    printf("Employee's start date: %d-%d-%d\n", \
           emp.date.day, emp.date.month, emp.date.year);
}
```

```
Employee name: John
Employee ID: 100
Employee salary: 500.00
Employee's start date: 12-3-2000

Process returned 0 (0x0)   execut
```

HOW TO:

- ✓ Create a structure, and nested structure
- ✓ Rename structure using **typedef**
- ✓ Create variable of type structure and Initialize data.
- ✓ Use and access data in a structure
- ✓ Access data in a nested structure

```c
#include<stdio.h>
struct Employee{
    char name[20];
    int id;
    float salary;
    struct date{
        int day;
        int month;
        int year;
    }date;
}emp={"John",100, 500, {12,3,2000}};

main(){
    printf("Employee name: %s\n", emp.name);
    printf("Employee ID: %d\n", emp.id);
    printf("Employee salary: %.2f\n", emp.salary);
    printf("Employee's start date: %d-%d-%d\n", \
           emp.date.day, emp.date.month, emp.date.year);
}
```

```
Employee name: John
Employee ID: 100
Employee salary: 500.00
Employee's start date: 12-3-2000

Process returned 0 (0x0)   execut
```

An example to create **Structure** (normal and nested structure) and using it in C programming

HOW TO:

- ✓ Create a structure, and nested structure
- ✓ **Initialize data to structure immediately while creating**
- ✓ Use and access data in a structure
- ✓ Access data in a nested structure

Break: 10mn
Start: 10:10am

Create new data type with structure

❑ Imbrication of structure / nested structure

- Suppose that in the structure type of student, we need to specify the date of birth for each student
- So we can declare type structure as the attribute of other structure
- Example:

```
struct date
        day, month, year: integer
end struct

struct student
        surname, name: string
        dob: date
        age: integer
end struct
```

How to use:

```
Var s: student
s.dob.year ← 1997
```

Practices

❑ Writing algorithms for the following exercises

Student
id
name
score

1. Create a student structure then create an array of 5 elements.
2. Write a procedure to display the information of all students.
3. Write a function to find the highest score of 5 students.
4. Write a procedure to display the information of student who has the highest.
5. Write a procedure to display the information of student who has the lowest score.