Lecture16
Set of Operations on an Array of Structures Database.
Using the Selection Statement 'switch' and Infinite Loop for Menu of Operations.

An **array of structures** in C language can be used for storing and processing information of the same elements (structure objects) in a simple **database.** For creating a database using an array of structures we have to determine a set of operations on it. Usually this **set of operations on an array of structures database** can be represented as follows:

1. dynamic memory allocation for an array of n elements of the given structure.
2. input elements of array of structures from keyboard.
3. output elements of array on the screen.
4. searching an element in array.
5. modifying an element of array.
6. swapping two elements of array.
7. sorting elements of array.
8. writing (saving) elements of array in file.
9. reading (loading) elements of array from file.
10. freeing memory dynamically allocated for array.
11. appending an element to the end of array.
12. inserting an element into array.
13. deleting an element from array.
14. other operations related to determine some information (statistics) from database.

All these operations (options) can be implemented in a C language program for an array of structures database processing by creating corresponding number of functions (subprograms) and then by calling them from **main( )** function in some order. Usually order and number of these function calls (order and number of options needed to perform) depend on and are determined by user during a working session with database. What is why it is necessary to develop appropriate user interface for communication between user and program. Let's consider the process of developing a simple user interface in C language based on **using the selection statement 'switch' and infinite loop** that gives as possibility to output on the screen a simple **menu of operations** (options) during a working session with database. First of all it is necessary to consider the selection statement **switch.**

Selection statement 'switch' in C language.

- Enables the program to execute different statements based on an expression that can have more than two values. **Statement switch** is also called **multiple choice statement**.
- Before this, using the **if statement**, we are limited to evaluate an expression that could have only two values: TRUE or FALSE.
- If there are more than two values, we had to use **nested if statements.**
- The **switch statement** makes such nesting unnecessary.
- Is used together with **case and default keywords** and **break statement.**
- The **switch statement** permits the execution of more than one alternative (by not placing **break** statements) whereas the **if statement** does not.
- The **switch statement** construct has the following form:

```
switch(expression)
{       case  literal1  : statement(s);
                                  break;
        case  literal2  : statement(s);
                                  break;
        .....................
```

```
        case  literalN : statement(s);
                                 break;
        default : statement(s);
    }
    next statement;
```

- Evaluates the expression and compares its value of **int** or **char** type with the literals of **int** or **char** type following each **case** label.

1. If a match is found between expression and one of the literals, execution is transferred to the statement(s) following the **case** label.
2. If no match is found, execution is transferred to the statement(s) following the optional **default** label.
3. If no match is found and there is no **default** label, execution passes to the first statement following the **switch statement** closing brace, the next statement.
4. To ensure that only the statements associated with the matching template are executed, include a break statement where needed, which terminates the entire **switch statement.**

Using the switch statement and infinite loop for output the menu of operations.

Here is an example of the program which outputs on the screen the menu of operations for database processing:

```
#include<stdio.h>  // beginning of program
#include<conio.h>
#include<stdlib.h>
#include<string.h>
// here is a place for texts of all functions
int main( )   // function main( ) header
{
 int nm;
 while( 1 )  // infinite while loop
 {
  clrscr( );
  puts("\n \t \t  menu:\n");
  puts("\n  1. dynamic memory allocation ");
  puts("\n  2. input an array from keyboard");
  puts("\n  3. output an array on the screen");
  puts("\n  4. searching an element");
  puts("\n  5. modifying an element");
  puts("\n  6. swapping two elements");
  puts("\n  7. sorting elements");
  puts("\n  8. writing elements in file");
  puts("\n  9. reading elements from file");
  puts("\n10. freeing memory");
  puts("\n11. appending an element");
  puts("\n12. inserting an element");
  puts("\n13. deleting an element");
  puts("\n  0. exit\n");
  printf("enter number of option:    ");
  scanf("%d", &nm);
  switch(nm)  // switch statement
   {
    case   1:  //statements  getch( ); break;
    case   2:  //statements  getch( ); break;
    case   3:  //statements  getch( ); break;
    case   4:  //statements  getch( ); break;
    case   5:  //statements  getch( ); break;
    case   6:  //statements  getch( ); break;
    case   7:  //statements  getch( ); break;
    case   8:  //statements  getch( ); break;
    case   9:  //statements  getch( ); break;
    case 10:  //statements  getch( ); break;
    case 11:  //statements  getch( ); break;
    case 12:  //statements  getch( ); break;
    case 13:  //statements  getch( ); break;
    case   0:    return 0;
    default :
      puts("\n  wrong number of option");
                getch( ); break;

   }  // end of switch statement
  }    // end of while loop
}      // end of program
```

Lecturer: Mihail Kulev, 13.07.08, Sudan, Jonglei State, Bor town.