PROGRAMMING METHODOLOGY

Lab 8: Abstract Datatype STRUCT

1 Introduction

In this lab tutorial, we'll consider an abstract datatype, called ***struct***. Structures – sometimes referred to as aggregates – are collections of related variables under one name. Structures may contain variables of many different data types – in contrast to arrays, which contain only elements of the same data type.

2 Structure Definition

Structures are derived data types – they're constructed using objects of other types. To define a structure, you must use the struct statement. The struct statement defines a new data type, with more than one member. The format of the struct statement is as follows:

```
struct [structure_tag] {
      datatype definition;
      datatype definition;
      …
      datatype definition;
} [one_or_more_structure_variables];
```

The structure tag is optional and each member definition is a normal variable definition, or any other valid variable definition. At the end of the structure's definition, before the final semicolon, you can specify one or more structure variables but it is optional. For example, declaring a Book structure as in the following figure.

```
struct Book {
      char title[100];
      char author[100];
      int book_id;
} book1;
```

Accessing Structure Members

To access any member of a structure, we use the member access operator (.). The member access operator is coded as a period between the structure variable name and the structure member that we wish to access. You would use the keyword struct to define variables of structure type.

The program below will illustrate how-to declare and use struct.

```
1   #include <stdio.h>
2   #include <string.h>
3   #define SIZE 100
4
5   struct Book {
6     char  title[SIZE];
7     char  author[SIZE];
8     int   book_id;
9   };
10
11  int main(int argc, char const *argv[])
12  {
13    struct Book book1, book2;
14
15    // book 1 specification
16    strcpy(book1.title, "C Programming Language");
17    strcpy(book1.author, "Brian W. Kernighan");
18    book1.book_id = 1001;
19
20    // book 2 specification
21    strcpy(book2.title, "C: How to Program");
22    strcpy(book2.author, "Paul Deitel");
23    book2.book_id = 1002;
24
25    // print book1 info
26    printf( "Book1 title : %s\n", book1.title);
27    printf( "Book1 author : %s\n", book1.author);
28    printf( "Book1 book_id : %d\n", book1.book_id);
29
30    // print book2 info
31    printf( "Book2 title : %s\n", book2.title);
32    printf( "Book2 author : %s\n", book2.author);
33    printf( "Book2 book_id : %d\n", book2.book_id);
34
35    return 0;
36  }
```

Structures as Function Arguments

You can pass a structure as a function argument in the same way as you pass any other variable or pointer:

```c
#include <stdio.h>
#include <string.h>
#define SIZE 100

struct Book {
  char  title[SIZE];
  char  author[SIZE];
  int   book_id;
};

void printInfo(struct Book);

int main(int argc, char const *argv[])
{
  struct Book book1, book2;

  // book 1 specification
  strcpy(book1.title, "C Programming Language");
  strcpy(book1.author, "Brian W. Kernighan");
  book1.book_id = 1001;

  // // book 2 specification
  strcpy(book2.title, "C: How to Program");
  strcpy(book2.author, "Paul Deitel");
  book2.book_id = 1002;

  // print book1 info
  printInfo(book1);

  // print book2 info
  printInfo(book2);

  return 0;
}

void printInfo(struct Book book)
{
  printf( "Title : %s\n", book.title);
  printf( "Author : %s\n", book.author);
  printf( "book_id : %d\n", book.book_id);
}
```

3 Exercises

1. Define an Employee structure contains the following properties: id, name, sex, birthyear, phone number, salary.
2. Define an array contains 10 employees.
3. Define a function to print all employee information.
4. Find an employee by providing his/her id.
5. Count the number of male/female employee.
6. Sort the employee array in ascending order of birthyear.

7. Sort the employee array in ascending order of salary.
8. Find an employee has the maximum salary.
9. Find a youngest employee.
10. Delete a specific employee.

4 Reference

[1] Brian W. Kernighan & Dennis Ritchie (1988). *C Programming Language, 2nd Edition*. Prentice Hall.

[2] Paul Deitel & Harvey Deitel (2008). *C: How to Program, 7th Edition*. Prentice Hall.

[3] *C Programming Tutorial* (2014). Tutorials Point.

[4] *C Programming* (2013). Wikibooks.