VTC Academy
Think Ahead, Beyond Boundaries

BASIC PROGRAMMING LANGUAGE

LESSON 11 - 12

User Defined Data Type

VTC Academy

@ 2021 VTC Academy

CONTENT

VTC Academy
Think Ahead, Beyond Boundaries

VTC Academy

@ 2021 VTC Academy

Introduction to User Defined Data Types

- Fundamental data types are basic built-in data types of C programming language.
- There are three fundamental data types in C programming. They are an integer data type, floating data type and character data type.
- User defined data types in C are data types which created by developers.
- Examples of such data types are structure, union and enumeration.

@ 2021 VTC Academy

Structure

- Arrays allow to define type of variables that can hold several data items of the same kind.
- Similarly structure is another user defined data type available in C that allows to combine data items of different kinds.
- Structures are used to represent a record.
- Suppose you want to keep track of your books in a library. You might want to track the following attributes about each book:
  - `isbn`
  - `title`
  - `author`
  - `price`

@ 2021 VTC Academy

Defining a Structure

- 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 name]
{
    member definition;
    member definition;
    ...
    member definition;
} [one or more structure variables];
```


@ 2021 VTC Academy

Defining a Structure

VTC Academy
Think Ahead, Beyond Boundaries

- Example of Book structure:

```
struct Book {
    char isbn[14];
    char title[50];
    char author[50];
    float price;
} book;
```

```
struct Book {
    char isbn[14];
    char title[50];
    char author[50];
    float price;
};
struct Book book;
```

VTC Academy

@ 2021 VTC Academy

Accessing Structure Elements

- 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

VTC Academy

@ 2021 VTC Academy

Accessing Structure Elements

VTC Academy
Think Ahead, Beyond Boundaries

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

struct Books{
    char isbn[15];
    char title[50];
    char author[50];
    float price;
};

int main( ) {
   struct Books book1;        /* Declare Book1 of type Book */
   struct Books book2;        /* Declare Book2 of type Book */

   /* book 1 specification */
   strcpy(book1.isbn, "978-0131103627");
   strcpy(book1.title, "The C Programming Language");
   strcpy(book1.author, "Dennis M. Ritchie");
   book1.price = 52.89;

   /* book 2 specification */
   strcpy(book2.isbn, "978-0789751980");
   strcpy(book2.title, "C Programming Absolute Beginner's Guide");
   strcpy(book2.author, "Dean Miller");
   book2.price = 24.32;
```

```c
   /* print Book1 info */
   printf( "Book 1 isbn: %s\n", book1.isbn);
   printf( "Book 1 title: %s\n", book1.title);
   printf( "Book 1 author: %s\n", book1.author);
   printf( "Book 1 price: %.2f\n", book1.price);

   /* print Book2 info */
   printf("Book 2 isbn: %s\n", book2.isbn);
   printf("Book 2 title: %s\n", book2.title);
   printf("Book 2 author: %s\n", book2.author);
   printf("Book 2 price: %.2f\n", book2.price);

   return 0;
}
```

VTC Academy

@ 2021 VTC Academy

Initializing & Assignment with Structures

VTC Academy
Think Ahead, Beyond Boundaries

- You can be initialized at the time of declaration:

```
struct Book book1 = {"978-0131103627", "The C Programming
Language", "Dennis M. Ritchie", 52.89};
struct Book book2;
```

- Possible to assign the values of one structure variable to another variable of the same type using a simple assignment statement:

```
book2 = book1;
```

VTC Academy

@ 2021 VTC Academy

Pointer to Structures

VTC Academy
Think Ahead, Beyond Boundaries

- You can create pointers to structs.
- Structure pointers are declared by placing an asterisk (*) in front of the structure variable's name. For example:

```
struct Person {
    char name[30];
    int age;
} *person;
```

```
struct Person *person1, person2;
person1 = &person2;
```

- To access members of a structure using pointers, use the –> operator:

```
person -> age = 30;
printf("Age: %d", person->age);
```


@ 2021 VTC Academy

Structure as Function Arguments

VTC Academy Think Ahead, Beyond Boundaries

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

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

struct Books{
    char isbn[15];
    char title[50];
    char author[50];
    float price;
};

void printBook(struct Books book);

int main( ) {
   struct Books book1;        /* Declare Book1 of type Book */
   struct Books book2;        /* Declare Book2 of type Book */
```


@ 2021 VTC Academy

Structure as Function Arguments

VTC Academy
Think Ahead, Beyond Boundaries

```
    /* book 1 specification */
    strcpy(book1.isbn, "978-0131103627");
    strcpy(book1.title, "The C Programming Language");
    strcpy(book1.author, "Dennis M. Ritchie");
    book1.price = 52.89;

    /* book 2 specification */
    strcpy(book2.isbn, "978-0789751980");
    strcpy(book2.title, "C Programming Absolute Beginner's Guide");
    strcpy(book2.author, "Dean Miller");
    book2.price = 24.32;

    /* print Book1 info */
    printBook(book1);
    /* print Book2 info */
    printBook(book2);

    return 0;
}

void printBook(struct Books book){
    printf( "Book isbn: %s\n", book.isbn);
    printf( "Book title: %s\n", book.title);
    printf( "Book author: %s\n", book.author);
    printf( "Book price: %.2f\n", book.price);
```


@ 2021 VTC Academy

Passing struct By Reference

VTC Academy Think Ahead, Beyond Boundaries

- You can also pass structs by reference. During pass by reference, the memory addresses of struct variables are passed to the function.
- Structure pointers passed as arguments to functions enable the functions to modify the structure elements directly.
- Example

```
struct Book {
    int isbn;
    char title[50];
    char author[50];
    char subject[100];
};
```

```
void printbook(struct Book *book) {
    printf("Book isbn: %d\n", book->isbn);
    printf("Book title: %s\n", book->title);
    printf("Book author: %s\n", book->author);
    printf("Book subject: %s\n", book->subject);
}
```


@ 2021 VTC Academy

Array of Structures

- Structure arrays are initialized by enclosing the list of values of its elements within a pair of braces.
- Example:

```
struct Book books[50];
```

- To the access the variable named author of the fourth element of the array books:

```
books[4].title
```

VTC Academy

@ 2021 VTC Academy

Array of Structures

- A common use of structures is in arrays of structures.
- A structure is first defined, and then an array variable of that type is declared.
- Example:

```
struct Book books[] = {
    {"978-013110", "The C Programming Language", "Dennis M. Ritchie", 52.89},
    {"978-078975", "C Programming For Beginner", "Dean Miller", 24.32}
};
```


@ 2021 VTC Academy

Array of Structures Example

VTC Academy
Think Ahead, Beyond Boundaries

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

struct Books{
    char isbn[15];
    char title[51];
    char author[51];
    float price;
};

void printBook(struct Books book);
struct Books getBook();
void getString(char *str, int length);
void printLine();
void printTitle();

int main( ) {
    struct Books books[3] = {
        {"9780131103627","The C Programming Language","Dennis Ritchie",52.89},
        {"9780789751980","C Programming for Beginner","Dean Miller",24.32}
    };
    int i, count = 3;
```

VTC Academy

@ 2021 VTC Academy

Array of Structures Example

VTC Academy Think Ahead, Beyond Boundaries

```
    printf("Input book 3:\n");
    books[2] = getBook();

    printTitle();
    for(i=0; i<count; i++){
        printBook(books[i]);
    }
    printLine();

    return 0;
}

struct Books getBook(){
    struct Books book;
    printf("Input Book isbn: ");
    getString(book.isbn, 14);
    printf("Input Book title: ");
    getString(book.title, 50);
    printf("Input Book author: ");
    getString(book.author, 50);
    printf("Input Book price: ");
    scanf("%f", &book.price);
    return book;
}
```

VTC Academy

@ 2021 VTC Academy

Array of Structures Example

VTC Academy
Think Ahead, Beyond Boundaries

```
void printBook(struct Books book){
    printf( "| %-14s | %-26s | %-20s | %6.2f |\n",
           book.isbn, book.title, book.author, book.price);
}
void printLine(){
    printf( "+-%-14s-+-%-26s-+-%-20s-+-%-6s-+\n", "--------------",
        "--------------------------", "--------------------", "------");
}
void printTitle(){
    printLine();
    printf( "| %-14s | %-26s | %-20s | %-6s |\n", "isbn", "Title",
        "Author", "Price");
    printLine();
}
void getString(char *str, int length){
    //clear keyboard buffer on UNIX
    fseek(stdin, 0, SEEK_END);
    //clear keyboard buffer on Windows
    fflush(stdin);
    //input string
    fgets(str, length, stdin);
    str[strlen(str)-1] = '\0';
    //clear keyboard buffer on UNIX
    fseek(stdin, 0, SEEK_END);
    //clear keyboard buffer on Windows
    fflush(stdin);
}
```

VTC Academy

@ 2021 VTC Academy

Dynamic Memory Allocation of Structs

VTC Academy
Think Ahead, Beyond Boundaries

- Sometimes, the number of struct variables you declared may be insufficient. You may need to allocate memory during run-time.
- To solve this problem, you can use pointer to structs and memory allocation functions.
- For example:

```
struct Person {
    char name[30];
    int age;
} *ptr;
ptr = (struct Person*) malloc(n * sizeof(struct Person));
```


@ 2021 VTC Academy

Dynamic Memory Allocation of Structs

VTC Academy
Think Ahead, Beyond Boundaries

- Input:

```
for (i = 0; i < n; i++) {
   printf("Enter name: ");
   scanf("%s", (ptr + i)->name);
   printf("Enter age: ");
   scanf("%d", &(ptr + i)->age);
}
```

- Output:

```
for (i = 0; i < n; i++) {
    printf("Name: %s\tAge: %d\n", (ptr + i)->name, (ptr + i)->age);
}
```

VTC Academy

@ 2021 VTC Academy

Dynamic Memory Allocation of Structs

VTC Academy
Think Ahead, Beyond Boundaries

Implement the example using array with structs with dynamic memory allocation

VTC Academy

@ 2021 VTC Academy

typedef Keyword

- A new data type name can be defined by using the keyword `typedef`.
- Does not create a new data type, but defines a new name for an existing type.
- `typedef` cannot be used with storage classes.
- Syntax:

```
typedef struct Book{
    int isbn;
    char title[50];
    char author[30];
    float price;
} Book;
```

```
// declare a variable of Books type,
// do not repeat struct keyword
Book book;
```

VTC Academy

@ 2021 VTC Academy

typedef vs #define

- `#define` is a C-directive which is also used to define the aliases for various data types similar to typedef but with the following differences:
  - `typedef` is limited to giving symbolic names to types only where as `#define` can be used to define alias for values as well, q., you can define 1 as ONE etc.
  - `typedef` interpretation is performed by the compiler whereas #define statements are processed by the pre-processor.

@ 2021 VTC Academy

typedef vs #define

VTC Academy
Think Ahead, Beyond Boundaries

```c
#include <stdio.h>

#define TRUE  1
#define FALSE 0

int main( ) {
   printf( "Value of TRUE : %d\n", TRUE);
   printf( "Value of FALSE : %d\n", FALSE);

   return 0;
}
```

VTC Academy

@ 2021 VTC Academy

Summary

- Structure is another user defined data type available in C that allows to combine data items of different kinds.
- The struct statement defines a new data type, with more than one member.
- To access any member of a structure, we use the member access operator (.)
- A common use of structures is in arrays of structures
- You can use pointer to structs and dynamic memory allocation with structs.
- A new data type name can be defined by using the keyword `typedef`.

@ 2021 VTC Academy

VTC Academy
Think Ahead, Beyond Boundaries

VTC Academy

@ 2021 VTC Academy