**Practice in online using following sites**

1. [**https://www.examveda.com/mcq-question-on-c-program/**](https://www.examveda.com/mcq-question-on-c-program/)
2. [**https://www.indiabix.com/computer-science/networking/**](https://www.indiabix.com/computer-science/networking/)

**Structure in c programming**

In C programming, a struct (or structure) is a collection of variables (can be of different types) under a single name.

**Create struct Variables**

When a struct type is declared, no storage or memory is allocated. To allocate memory of a given structure type and work with it, we need to create variables.

Here's how we create structure variables:

struct Person {

// code

};

int main () {

struct Person person1, person2, p[20];

return 0;

}

Another way of creating a struct variable is:

struct Person {

// code

} person1, person2, p[20];

In both cases,

* person1 and person2 are struct Person variables
* p[] is a struct Person array of size 20.

**Access Members of a Structure**

There are two types of operators used for accessing members of a structure.

1. . - Member operator
2. -> - Structure pointer operator (will be discussed in the next tutorial)

Suppose, you want to access the salary of person2. Here's how you can do it.

#include <stdio.h>

#include <string.h>

// create struct with person1 variable

struct Person {

char name[50];

int citNo;

float salary;

} person1;

int main() {

// assign value to name of person1

strcpy(person1.name, "George Orwell");

// assign values to other person1 variables

person1.citNo = 1984;

person1. salary = 2500;

// print struct variables

printf("Name: %s\n", person1.name);

printf("Citizenship No.: %d\n", person1.citNo);

printf("Salary: %.2f", person1.salary);

return 0;

}

**Keyword typedef**

We use the typedef keyword to create an alias name for data types. It is commonly used with structures to simplify the syntax of declaring variables.

For example, let us look at the following code:

struct Distance{

int feet;

float inch;

};

int main() {

struct Distance d1, d2;

}

We can use typedef to write an equivalent code with a simplified syntax:

typedef struct Distance {

int feet;

float inch;

} distances;

int main() {

distances d1, d2;

}

**Why structs in C?**

Suppose you want to store information about a person: his/her name, citizenship number, and salary. You can create different variables name, citNo and salary to store this information.

What if you need to store information of more than one person? Now, you need to create different variables for each information per person: name1, citNo1, salary1, name2, citNo2, salary2, etc.

A better approach would be to have a collection of all related information under a single name Person structure and use it for every person.

**Access members using Pointer**

To access members of a structure using pointers, we use the -> operator.

#include <stdio.h>

struct person

{

int age;

float weight;

};

int main()

{

struct person \*personPtr, person1;

personPtr = &person1;

printf("Enter age: ");

scanf("%d", &personPtr->age);

printf("Enter weight: ");

scanf("%f", &personPtr->weight);

printf("Displaying:\n");

printf("Age: %d\n", personPtr->age);

printf("weight: %f", personPtr->weight);

return 0;

}

By the way,

* personPtr->age is equivalent to (\*personPtr).age
* personPtr->weight is equivalent to (\*personPtr).weight

## Dynamic memory allocation of structs

Before you proceed this section, we recommend you to check [C dynamic memory allocation](https://www.programiz.com/c-programming/c-dynamic-memory-allocation).

Sometimes, the number of struct variables you declared may be insufficient. You may need to allocate memory during run-time. Here's how you can achieve this in C programming.

### Example: Dynamic memory allocation of structs

#include <stdio.h>

#include <stdlib.h>

struct person {

int age;

float weight;

char name[30];

};

int main()

{

struct person \*ptr;

int i, n;

printf("Enter the number of persons: ");

scanf("%d", &n);

// allocating memory for n numbers of struct person

ptr = (struct person\*) malloc(n \* sizeof(struct person));

for(i = 0; i < n; ++i)

{

printf("Enter first name and age respectively: ");

// To access members of 1st struct person,

// ptr->name and ptr->age is used

// To access members of 2nd struct person,

// (ptr+1)->name and (ptr+1)->age is used

scanf("%s %d", (ptr+i)->name, &(ptr+i)->age);

}

printf("Displaying Information:\n");

for(i = 0; i < n; ++i)

printf("Name: %s\tAge: %d\n", (ptr+i)->name, (ptr+i)->age);

return 0;

}

## Passing structs to functions

gets input from users in three way

**#include <stdio.h>**

**struct student {**

**char name[50];**

**int age;**

**};**

**// function prototype**

**void display(struct student s);**

**int main() {**

**struct student s1;**

**printf("Enter name: ");**

**// read string input from the user until \n is entered**

**// \n is discarded**

**//gets input from users in three way**

**// scanf("%[^\n]%\*c", s1.name);**

**fgets(s1.name,sizeof(s1.name),stdin);**

**// gets(s1.name)**

**printf("Enter age: ");**

**scanf("%d", &s1.age);**

**display(s1); // passing struct as an argument**

**return 0;**

**}**

**void display(struct student s) {**

**printf("\nDisplaying information\n");**

**printf("Name: %s", s.name);**

**printf("\nAge: %d", s.age);**

**}**

A union is a user-defined type similar to [structs in C](https://www.programiz.com/c-programming/c-structures) except for one key difference.

Structures allocate enough space to store all their members, whereas **unions can only hold one member value at a time**.

let's take an example to demonstrate the difference between unions and structures:

#include <stdio.h>

union unionJob

{

//defining a union

char name[32];

float salary;

int workerNo;

} uJob;

struct structJob

{

char name[32];

float salary;

int workerNo;

} sJob;

int main()

{

printf("size of union = %d bytes", sizeof(uJob));

printf("\nsize of structure = %d bytes", sizeof(sJob));

return 0;

}

size of union = 32

size of structure = 40

**Why this difference in the size of union and structure variables?**

Here, the size of sJob is 40 bytes because

* the size of name[32] is 32 bytes
* the size of salary is 4 bytes
* the size of workerNo is 4 bytes

However, the size of uJob is 32 bytes. It's because the size of a union variable will always be the size of its largest element. In the above example, the size of its largest element, (name[32]), is 32 bytes.

With a union, all members share **the same memory**.

#include <stdio.h>

union Job {

float salary;

int workerNo;

} j;

int main() {

j.salary = 12.3;

// when j.workerNo is assigned a value,

// j.salary will no longer hold 12.3

j.workerNo = 100;

printf("Salary = %x\n", &j.salary);

printf("Number of workers = %x", &j.workerNo);

return 0;

}

Output: Salary = 404034

Number of workers = 404034

That’s why output will show only for workerNo

# C File Handling

A file is a container in computer storage devices used for storing data.

## Why files are needed?

* When a program is terminated, the entire data is lost. Storing in a file will preserve your data even if the program terminates.
* If you have to enter a large number of data, it will take a lot of time to enter them all.  
  However, if you have a file containing all the data, you can easily access the contents of the file using a few commands in C.
* You can easily move your data from one computer to another without any changes.

## Types of Files

When dealing with files, there are two types of files you should know about:

1. Text files
2. Binary files

### 1. Text files

Text files are the normal **.txt** files. You can easily create text files using any simple text editors such as Notepad.

When you open those files, you'll see all the contents within the file as plain text. You can easily edit or delete the contents.

They take minimum effort to maintain, are easily readable, and provide the least security and takes bigger storage space.

### 2. Binary files

Binary files are mostly the **.bin** files in your computer.

Instead of storing data in plain text, they store it in the binary form (0's and 1's).

They can hold a higher amount of data, are not readable easily, and provides better security than text files.

## File Operations

In C, you can perform four major operations on files, either text or binary:

1. Creating a new file
2. Opening an existing file
3. Closing a file
4. Reading from and writing information to a file

## Working with files

When working with files, you need to declare a pointer of type file. This declaration is needed for communication between the file and the program.

FILE \*fptr;

| Opening Modes in Standard I/O | | |
| --- | --- | --- |
| Mode | Meaning of Mode | During Inexistence of file |
| r | Open for reading. | If the file does not exist, fopen() returns NULL. |
| rb | Open for reading in binary mode. | If the file does not exist, fopen() returns NULL. |
| w | Open for writing. | If the file exists, its contents are overwritten. If the file does not exist, it will be created. |
| wb | Open for writing in binary mode. | If the file exists, its contents are overwritten. If the file does not exist, it will be created. |
| a | Open for append. Data is added to the end of the file. | If the file does not exist, it will be created. |
| ab | Open for append in binary mode. Data is added to the end of the file. | If the file does not exist, it will be created. |
| r+ | Open for both reading and writing. | If the file does not exist, fopen() returns NULL. |
| rb+ | Open for both reading and writing in binary mode. | If the file does not exist, fopen() returns NULL. |
| w+ | Open for both reading and writing. | If the file exists, its contents are overwritten. If the file does not exist, it will be created. |
| wb+ | Open for both reading and writing in binary mode. | If the file exists, its contents are overwritten. If the file does not exist, it will be created. |
| a+ | Open for both reading and appending. | If the file does not exist, it will be created. |
| ab+ | Open for both reading and appending in binary mode. | If the file does not exist, it will be created. |

## Reading and writing to a text file

For reading and writing to a text file, we use the functions fprintf() and fscanf().

They are just the file versions of printf() and scanf(). The only difference is that fprintf() and fscanf() expects a pointer to the structure FILE.

### Example 1: Write to a text file

#include <stdio.h>

#include <stdlib.h>

int main()

{

int num;

FILE \*fptr;

// use appropriate location if you are using MacOS or Linux

fptr = fopen("C:\\program.txt","w");

if(fptr == NULL)

{

printf("Error!");

exit(1);

}

printf("Enter num: ");

scanf("%d",&num);

fprintf(fptr,"%d",num);

fclose(fptr);

return 0;

}

This program takes a number from the user and stores in the file program.txt.

After you compile and run this program, you can see a text file program.txt created in C drive of your computer. When you open the file, you can see the integer you entered.

### Example 2: Read from a text file

#include <stdio.h>

#include <stdlib.h>

int main()

{

int num;

FILE \*fptr;

if ((fptr = fopen("C:\\program.txt","r")) == NULL){

printf("Error! opening file");

// Program exits if the file pointer returns NULL.

exit(1);

}

fscanf(fptr,"%d", &num);

printf("Value of n=%d", num);

fclose(fptr);

return 0;

}

This program reads the integer present in the program.txt file and prints it onto the screen.

If you successfully created the file from **Example 1**, running this program will get you the integer you entered.

Other functions like fgetchar(), fputc() etc. can be used in a similar way.

What is stdlib. h in C? stdlib. h is a header file and also the Standard Library of C programming language that declares various utility functions for type conversions, memory allocation, algorithms, and other similar use cases.

**HTML,CSS,JAVASCRIPTS ,PHP**

JavaScript is one of the **3 languages** in web developing:

   1. [**HTML**](https://www.w3schools.com/html/default.asp) to define the content of web pages

   2. [**CSS**](https://www.w3schools.com/css/default.asp) to specify the layout of web pages

   3. **JavaScript** to program the behavior of web pages

**1.What is variable?**

## JavaScript Variables

In programming, a variable is a container (storage area) to hold data. For example,

let num = 5;

Here, num is a variable. It's storing **5**

### JavaScript var Vs let

|  |  |
| --- | --- |
| var | let |
| var is used in the older versions of JavaScript | let is the new way of declaring variables starting **ES6 (ES2015)**. |
| var is function scoped . | let is block scoped. |
| For example, var x; | For example, let y; |

Both var and let are used to declare variables. However, there are some differences between them.

## JavaScript Constants

 a constant is a type of variable whose value cannot be changed

const x = 5;

x = 10; // Error! constant cannot be changed.

console.log(x)

## JavaScript Data Types

|  |  |  |
| --- | --- | --- |
| Data Types | Description | Example |
| String | represents textual data | 'hello', "hello world!" etc |
| Number | an integer or a floating-point number | 3, 3.234, 3e-2 etc. |
| BigInt | an integer with arbitrary precision | 900719925124740999n , 1n etc. |
| Boolean | Any of two values: true or false | true and false |
| undefined | a data type whose variable is not initialized | let a; |
| null | denotes a null value | let a = null; |
| Symbol | data type whose instances are unique and immutable | let value = Symbol('hello'); |
| Object | key-value pairs of collection of data | let student = { }; |

There are eight basic data types in JavaScript. They are:

## JavaScript undefined

The undefined data type represents **value that is not assigned**. If a variable is declared but the value is not assigned, then the value of that variable will be undefined.

## JavaScript null

In JavaScript, null is a special value that represents **empty** or **unknown value**. For example,

const number = null;

## JavaScript Operator Types

Here is a list of different operators you will learn in this tutorial.

* Assignment Operators
* Arithmetic Operators
* Comparison Operators
* Logical Operators
* Bitwise Operators
* String Operators
* Other Operators

# JavaScript Comments

JavaScript comments are hints that a programmer can add to make their code easier to read and understand. They are completely ignored by JavaScript engines.

There are two ways to add comments to code:

* // - Single Line Comments
* /\* \*/ -Multi-line Comments

# JavaScript for loop

In programming, loops are used to repeat a block of code.

# JavaScript Recursion

Recursion is a process of calling itself. A function that calls itself is called a recursive function.

The syntax for recursive function is:

function recurse() {

// function code

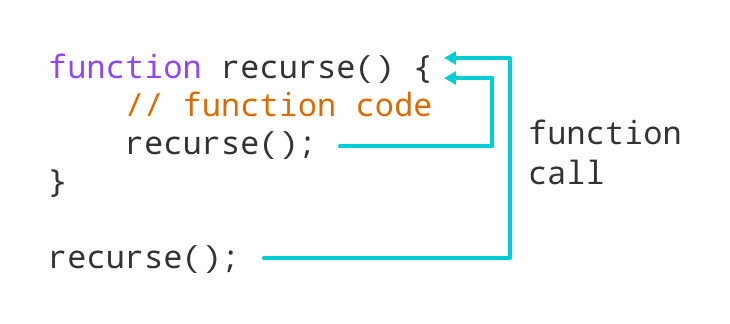
recurse();

// function code

}

recurse();

Here, the recurse() function is a recursive function. It is calling itself inside the function.



JavaScript object is a non-primitive data-type that allows you to store multiple collections of data.