**C++ Structures**

In this article, you'll learn about structures in C++ programming; what is it, how to define it and use it in your program.

Structure is a collection of variables of different data types under a single name. It is similar to a [class](https://www.programiz.com/cpp-programming/object-class) in that, both holds a collecion of data of different data types.

**For example:**You want to store some information about a person: his/her name, citizenship number and salary. You can easily create different variables name, citNo, salary to store these information separately.

However, in the future, you would want to store information about multiple persons. Now, you'd need to create different variables for each information per person: name1, citNo1, salary1, name2, citNo2, salary2

You can easily visualize how big and messy the code would look. Also, since no relation between the variables (information) would exist, it's going to be a daunting task.

A better approach will be to have a collection of all related information under a single name Person, and use it for every person. Now, the code looks much cleaner, readable and efficient as well.

This collection of all related information under a single name Person is a structure.

**How to declare a structure in C++ programming?**

The struct keyword defines a structure type followed by an identifier (name of the structure).

Then inside the curly braces, you can declare one or more members (declare variables inside curly braces) of that structure. For example:

**struct** Person

{

char name[50];

int age;

float salary;

};

Here a structure person is defined which has three members: name, age and salary.

When a structure is created, no memory is allocated.

The structure definition is only the blueprint for the creating of variables. You can imagine it as a datatype. When you define an integer as below:

int foo;

The int specifies that, variable foo can hold integer element only. Similarly, structure definition only specifies that, what property a structure variable holds when it is defined.

**Note:**Remember to end the declaration with a semicolon **(;)**

**How to define a structure variable?**

Once you declare a structure person as above. You can define a structure variable as:

Person bill;

Here, a structure variable bill is defined which is of type structure Person.

When structure variable is defined, only then the required memory is allocated by the compiler.

Considering you have either 32-bit or 64-bit system, the memory of float is 4 bytes, memory of int is 4 bytes and memory of char is 1 byte.

Hence, 58 bytes of memory is allocated for structure variable bill.

**How to access members of a structure?**

The members of structure variable is accessed using a **dot (.)** operator.

Suppose, you want to access age of structure variable bill and assign it 50 to it. You can perform this task by using following code below:

bill.age = 50;

**Example: C++ Structure**

C++ Program to assign data to members of a structure variable and display it.

#include <iostream>

using namespace std;

struct Person

{

char name[50];

int age;

float salary;

};

int main()

{

Person p1;

cout << "Enter Full name: ";

cin.get(p1.name, 50);

cout << "Enter age: ";

cin >> p1.age;

cout << "Enter salary: ";

cin >> p1.salary;

cout << "\nDisplaying Information." << endl;

cout << "Name: " << p1.name << endl;

cout <<"Age: " << p1.age << endl;

cout << "Salary: " << p1.salary;

return 0;

}

**Output**

Enter Full name: Magdalena Dankova

Enter age: 27

Enter salary: 1024.4

Displaying Information.

Name: Magdalena Dankova

Age: 27

Salary: 1024.4

Here a structure Person is declared which has three members name, age and salary.

Inside main() [function](https://www.programiz.com/cpp-programming/function), a structure variable p1 is defined. Then, the user is asked to enter information and data entered by user is displayed.

You should also check out these structure related tutorials:

* [How to pass structures to functions?](https://www.programiz.com/cpp-programming/structure-function)
* [How to use pointers with structures?](https://www.programiz.com/cpp-programming/structure-pointer)

# C++ Structure and Function

#### In this article, you'll find relevant examples to pass structures as an argument to a function, and use them in your program.

[Structure](https://www.programiz.com/cpp-programming/structure) variables can be passed to a [function](https://www.programiz.com/cpp-programming/function) and returned in a similar way as normal arguments.

## Passing structure to function in C++

A structure variable can be passed to a function in similar way as normal argument. Consider this example:

### Example 1: C++ Structure and Function

#include <iostream>

using namespace std;

struct Person

{

char name[50];

int age;

float salary;

};

void displayData(Person); // Function declaration

int main()

{

Person p;

cout << "Enter Full name: ";

cin.get(p.name, 50);

cout << "Enter age: ";

cin >> p.age;

cout << "Enter salary: ";

cin >> p.salary;

// Function call with structure variable as an argument

displayData(p);

return 0;

}

void displayData(Person p)

{

cout << "\nDisplaying Information." << endl;

cout << "Name: " << p.name << endl;

cout <<"Age: " << p.age << endl;

cout << "Salary: " << p.salary;

}

**Output**

Enter Full name: Bill Jobs

Enter age: 55

Enter salary: 34233.4

Displaying Information.

Name: Bill Jobs

Age: 55

Salary: 34233.4

In this program, user is asked to enter the name, age and salary of a Person inside main() function.

Then, the structure variable p is to passed to a function using.

displayData(p);

The return type of displayData() is void and a single argument of type structure Person is passed.

Then the members of structure p is displayed from this function.

### Example 2: Returning structure from function in C++

#include <iostream>

using namespace std;

struct Person {

char name[50];

int age;

float salary;

};

Person getData(Person);

void displayData(Person);

int main()

{

Person p;

p = getData(p);

displayData(p);

return 0;

}

Person getData(Person p) {

cout << "Enter Full name: ";

cin.get(p.name, 50);

cout << "Enter age: ";

cin >> p.age;

cout << "Enter salary: ";

cin >> p.salary;

return p;

}

void displayData(Person p)

{

cout << "\nDisplaying Information." << endl;

cout << "Name: " << p.name << endl;

cout <<"Age: " << p.age << endl;

cout << "Salary: " << p.salary;

}

The output of this program is same as program above.

In this program, the structure variable p of type structure Person is defined under main() function.

The structure variable p is passed to getData() function which takes input from user which is then returned to main function.

p = getData(p);

**Note:** The value of all members of a structure variable can be assigned to another structure using assignment operator **=** if both structure variables are of same type. You don't need to manually assign each members.

Then the structure variable p is passed to displayData() function, which displays the information.

# C++ Pointers to Structure

#### In this article, you'll find relevant examples that will help you to work with pointers to access data within a structure.

A [pointer](https://www.programiz.com/cpp-programming/pointers) variable can be created not only for native types like (int, float, double etc.) but they can also be created for user defined types like [structure](https://www.programiz.com/cpp-programming/structure).

If you do not know what pointers are, visit [C++ pointers](https://www.programiz.com/cpp-programming/pointers).

Here is how you can create pointer for structures:

#include <iostream>

using namespace std;

struct temp {

int i;

float f;

};

int main() {

temp \*ptr;

return 0;

}

This program creates a pointer ptr of type structure temp.

## Example: Pointers to Structure

#include <iostream>

using namespace std;

struct Distance

{

int feet;

float inch;

};

int main()

{

Distance \*ptr, d;

ptr = &d;

cout << "Enter feet: ";

cin >> (\*ptr).feet;

cout << "Enter inch: ";

cin >> (\*ptr).inch;

cout << "Displaying information." << endl;

cout << "Distance = " << (\*ptr).feet << " feet " << (\*ptr).inch << " inches";

return 0;

}

**Output**

Enter feet: 4

Enter inch: 3.5

Displaying information.

Distance = 4 feet 3.5 inches

In this program, a pointer variable ptr and normal variable d of type structure Distance is defined.

The address of variable d is stored to pointer variable, that is, ptr is pointing to variable d. Then, the member function of variable d is accessed using pointer.

**Note:** Since pointer ptr is pointing to variable d in this program, (\*ptr).inch and d.inch is exact same cell. Similarly, (\*ptr).feet and d.feet is exact same cell.

The syntax to access member function using pointer is ugly and there is alternative notation **->** which is more common.

ptr->feet is same as (\*ptr).feet

ptr->inch is same as (\*ptr).inch

# C++ Enumeration

#### In this article, you will learn to work with enumeration (enum). Also, you will learn where enums are commonly used in C++ programming.

An enumeration is a user-defined data type that consists of integral constants. To define an enumeration, keyword **enum** is used.

enum season { spring, summer, autumn, winter };

Here, the name of the enumeration is season.

And, spring, summer and winter are values of type season.

By default, spring is 0, summer is 1 and so on. You can change the default value of an enum element during declaration (if necessary).

enum season

{ spring = 0,

summer = 4,

autumn = 8,

winter = 12

};

## Enumerated Type Declaration

When you create an enumerated type, only blueprint for the variable is created. Here's how you can create variables of enum type.

enum boolean { false, true };

// inside function

enum boolean check;

Here, a variable check of type **enum boolean** is created.

Here is another way to declare same check variable using different syntax.

enum boolean

{

false, true

} check;

### Example 1: Enumeration Type

#include <iostream>

using namespace std;

enum week { Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday };

int main()

{

week today;

today = Wednesday;

cout << "Day " << today+1;

return 0;

}

**Output**

Day 4

### Example2: Changing Default Value of Enums

#include <iostream>

using namespace std;

enum seasons { spring = 34, summer = 4, autumn = 9, winter = 32};

int main() {

seasons s;

s = summer;

cout << "Summer = " << s << endl;

return 0;

}

**Output**

Summer = 4

## Why enums are used in C++ programming?

An enum variable takes only one value out of many possible values. Example to demonstrate it,

#include <iostream>

using namespace std;

enum suit {

club = 0,

diamonds = 10,

hearts = 20,

spades = 3

} card;

int main()

{

card = club;

cout << "Size of enum variable " << sizeof(card) << " bytes.";

return 0;

}

**Output**

Size of enum variable 4 bytes.

It's because the size of an integer is 4 bytes.;

This makes enum a good choice to work with flags.

You can accomplish the same task using [C++ structures](https://www.programiz.com/cpp-programming/structure). However, working with enums gives you efficiency along with flexibility.

### How to use enums for flags?

Let us take an example,

enum designFlags {

ITALICS = 1,

BOLD = 2,

UNDERLINE = 4

} button;

Suppose you are designing a button for Windows application. You can set flags ITALICS, BOLD and UNDERLINE to work with text.

There is a reason why all the integral constants are power of 2 in above pseudocode.

// In binary

ITALICS = 00000001

BOLD = 00000010

UNDERLINE = 00000100

Since, the integral constants are power of 2, you can combine two or more flags at once without overlapping using bitwise OR **|** operator. This allows you to choose two or more flags at once. For example,

#include <iostream>

using namespace std;

enum designFlags {

BOLD = 1,

ITALICS = 2,

UNDERLINE = 4

};

int main()

{

int myDesign = BOLD | UNDERLINE;

// 00000001

// | 00000100

// \_\_\_\_\_\_\_\_\_\_\_

// 00000101

cout << myDesign;

return 0;

}

Output

5

When the output is 5, you always know that bold and underline is used.

Also, you can add flag to your requirements.

if (myDesign & ITALICS) {

// code for italics

}

Here, we have added italics to our design. Note, only code for italics is written inside the [if statement](https://www.programiz.com/cpp-programming/if-else).

You can accomplish almost anything in C++ programming without using enumerations. However, they can be pretty handy in certain situations. That's what differentiates good programmers from great programmers.