Programming Using
C
My Notes

PreparedBy

Rakesh Deep

2020

SYLLABUS

OBJECTIVES:

_ TolearnbasicsofCprogramminglanguage.

_ Tobeabletodeveloplogicstocreateprograms/applicationsinC.

Unit-1

Introduction: Introduction to Programming Language, Introduction to C Programming, Keywords & Identifiers, Constants, Variables, Input and Output Operations, Compilation andpre-processing, **Data types**: Different data types, Data types qualifier, modifiers, Memory representation, size and range, **Operators:** Operators (Arithmetic, Relational, Logical, Bitwise, Assignment & compound assignment, Increment & Decrement, Conditional), Operator types (unary, binary, ternary). Expressions, Order of expression (Precedence and associativity)

Control structures: Decision Making and Branching (Simple IF Statement, IF…ELSE Statement, Nesting IF… ELSE Statement, ELSE IF Ladder), Selection control structure (Switch Statement).

Unit-2

Loops: The WHILE Statement, The DO…WHILE Statement, The FOR Statement, Jumps in Loops,**Array**: Concept ofArray,ArrayDeclaration,typesofarray(oneandmultiple dimension), Character Arrays and Strings, Subscript and pointer representation of array, Array of Pointers, Limitation of array, **Pointers**: Concept of Pointer (null pointer, wild pointer, dangling pointer, generic pointer), Pointer Expressions, Accessing the Address of a Variable, Declaring Pointer Variables, Initializations of Pointer Variable, Accessing a Variable through its Pointer, Pointer arithmetic.

Unit-3

Storage class: Types (auto, register, static, extern), scope rules, declaration and definition.

Function:Function&types(Userdefinedfunction,libraryfunction)FunctionDefinition,Declaration,FunctionCalls,Headerfileandlibrary,FunctionArguments,stringhandling function (strlen, strcmp, strcpy, strncpy, strcat, strstr), Function recursion, Functions ReturningPointers,PointerstoFunctions,Commandlinearguments,Applicationofpointer(dynamic memory allocation).

Unit-4

Structure and Union: Defining, Declaring, Accessing, Initialization Structure, nested structure, self-referential structure, bit-field, Arrays of Structures, Structures and Functions, Unions, difference between structure and union, active data member, structure within union, Self-referential Structure,

File: File Management in C, Defining and Opening a File, File opening modes (read, write, append), Closing a File, File operations, file and stream, Error Handling During I/O Operations, sequential and random access file, low level and high level file.

Text Books:

1. E.Balagurusamy,"ProgramminginANSIC",4/e,(TMH)

ReferenceBooks:

1. B.Kernighan&DennisRitchie,"TheCProgrammingLanguage",2/ePHI
2. PaulDeitel,HarveyDeitel,"C:HowtoProgram",8/e,PrenticeHall.
3. P.C.Sethi,P.K.Behera,"ProgrammingusingC",KalyaniPublisher,Ludhiana

UNIT-1

ProgramminginC

Cprogrammingisapopularcomputerprogramminglanguagewhichiswidely usedforsystem and application software. Despite being fairly old programming language, C programming is widely used because of its efficiency and control. This tutorial is intended for beginners who does not have any prior knowledge or have very little knowledge of computer programming. All basic features of C programming language are included in detail with explanation and output to give you solid platform to understand C programming.

CProgrammingKeywordsandIdentifiers

Characterset

Character set are the set of alphabets, letters and some special characters that are valid in C language.

Alphabets:

Uppercase:ABC .....XYZ

Lowercase:ab c .....x y z

Digits:

0 1 2 3 4 5 68 9

Special Characters:

SpecialCharactersinClanguage

,	<	>	.	_	(	)	;	$	:	%	[	]	#	?	
'	&	{	}	"	^	!	*	/	|	-	\	~	+		

WhitespaceCharacters:

blankspace,newline,horizontaltab,carriagereturnandform feed

Keywords:

Keywordsarethereservedwordsusedinprogramming.Eachkeywordshasfixed meaning and that cannot be changed by user. For example:

```
intmoney;
```

Here,intis akeywordthat indicates,*'money'*is oftype integer.

As, C programming is case sensitive, all keywords must be written in lowercase. Here is the list of all keywords predefined by ANSI C.

KeywordsinCLanguage

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
continue	for	signed	void
do	if	static	while
default	goto	sizeof	volatile
const	float	short	unsigned

Besidesthesekeywords, therearesomeadditionalkeywordssupported byTurbo C.

AdditionalKeywordsforBorlandC

asm	far	interrupt	pascal	near	huge	cdecl

All these keywords, their syntax and application will be discussed in their respective topics. However, if you want brief information about these keywords without going further visit page: list of all C keywords.

Identifiers

In C programming, identifiers are names given to C entities, such as variables, functions, structures etc. Identifier are created to give unique name to C entities to identify it during the execution of program. For example:

```
intmoney;
intmango_tree;
```

Here,*money*isaidentifierwhichdenotesavariableoftypeinteger.Similarly, *mango_tree*isanotheridentifier, whichdenotesanothervariableoftype integer.

Rulesforwritingidentifier

1. An identifier can be composed of letters (both uppercase and lowercase letters), digitsand underscore '_' only.
2. The first letter of identifier should be either a letter or an underscore. But, it is discouragedto startanidentifier namewith anunderscorethoughit is legal. Itis because, identifier that starts with underscore can conflict with system names. In such cases, compilerwillcomplainaboutit.Somesystemnamesthatstartwithunderscoreare *_fileno*,*_iob*,*_wfopen*etc.
3. There is no rule for the length of an identifier. However, the first 31 characters of an identifier arediscriminated by the compiler. So, the first 31 letters of two identifiers in a program should be different.

TipsforGood ProgrammingPractice:

Programmer can choose the name of identifier whatever they want. However, if the programmer choose meaningful name for an identifier, it will be easy to understand and work on, particularly in case of large program.

C Programming Variables and Constants

Variables

Variables are memory location in computer's memory to store data. To indicate the memory location, each variable should be given a unique name called identifier. Variable names are just the symbolic representation of a memory location. Examples of variable name: *sum*, *car_no*, *count* etc.

int *num*;

Here,*num*isa variableofinteger type.

RulesforwritingvariablenameinC

1. Variable name can be composed of letters (both uppercase and lowercase letters), digits and underscore '_' only.
2. Thefirstletterofavariableshouldbeeitheraletteroranunderscore.But,itis discouragedtostartvariablenamewithanunderscorethoughitislegal.Itisbecause,

variable name that starts with underscore can conflict with system names and compiler may complain.

3. Thereisnoruleforthelengthof lengthofavariable.However,thefirst31characters ofavariable arediscriminated by the compiler. So,the first 31 letters of two variables in a program should be different.

InCprogramming, youhavetodeclarevariablebeforeusingit intheprogram.

Constants

Constantsarethetermsthatcan'tbechangedduringtheexecutionofaprogram.For example: 1, 2.5, "Programming is easy." etc. In C, constants can be classified as:

Integerconstants

Integerconstantsarethenumericconstants(constantassociatedwithnumber)withoutany fractional part or exponential part. There are three types of integer constants in C language: decimal constant(base 10), octal constant(base 8) and hexadecimal constant(base 16) .

Decimaldigits: 01 2 34 5 67 8 9

Octaldigits: 0 1 23 4 5 6 7

Hexadecimaldigits:0123456789ABCDEF. For

example:

```
Decimalconstants:0,-9,22 etc
Octal constants: 021, 077, 033 etc
Hexadecimalconstants:0x7f,0x2a,0x521etc
```

Notes:

1. Youcanusesmallcaps*a*,*b*,*c*,*d*,*e*,*f*insteadofuppercaseletterswhilewritinga hexadecimal constant.
2. Everyoctalconstantstartswith0andhexadecimalconstantstartswith0xinC programming.

Floating-pointconstants

Floatingpointconstantsarethenumericconstantsthathaseitherfractionalformor exponent form. For example:

```
-2.0
0.0000234
```

```
-0.22E-5
```

Note: Here,E-5represents10^{-5}.Thus,`-0.22E-5=-0.0000022`.

Characterconstants

Character constantsarethe constantwhichusesingle quotationaroundcharacters.For example: 'a', 'l', 'm', 'F' etc.

EscapeSequences

Sometimes, it is necessary to use newline(enter), tab, quotation mark etc. in the program which either cannot be typed or has special meaning in C programming. In such cases, escape sequence are used. For example: `\n`is used for newline. The backslash(\ ) causes "escape" from the normal way the characters are interpreted by the compiler.

Escape Sequences

EscapeSequences	**Character**
\b	Backspace
\f	Formfeed
\n	Newline
\r	Return
\t	Horizontaltab
\v	Verticaltab
\\	Backslash
\'	Singlequotation mark
\"	Doublequotationmark
\?	Questionmark
\0	Nullcharacter

Stringconstants

Stringconstantsaretheconstantswhichareenclosedinapairofdouble-quotemarks.For example:

```
"good"                  //stringconstant
""                     //nullstringconstant
"      "               //stringconstantofsixwhitespace
"x"                    //stringconstanthavingsinglecharacter.
"Earth is round\n"         //prints string with newline
```

Enumeration constants

Keywordenumisusedtodeclareenumerationtypes.For example:

```
enumcolor{yellow,green,black,white};
```

Here, the variable name is color and yellow, green, black and white are the enumeration constants having value 0, 1, 2 and 3 respectively by default. For more information about enumeration, visit page: Enumeration Types.

CProgrammingDataTypes

In C, variable(data) should be declared before it can be used in program. Data types arethe keywords, which are used for assigning a type to a variable.

Datatypesin C

1. FundamentalDataTypes
    - Integertypes
    - FloatingType
    - Charactertypes
2. DerivedDataTypes
    - Arrays
    - Pointers
    - Structures
    - Enumeration

Syntaxfordeclarationofavariable

```
data_typevariable_name;
```

Integerdatatypes

Keywordintisusedfordeclaringthevariablewithintegertype.Forexample:

```
intvar1;
```

Here,*var1*isavariableoftype integer.

Thesizeofintiseither2bytes(InolderPC's)or4bytes.Ifyouconsideraninteger havingsizeof4byte(equalto32bits),itcantake2^{32}distinctstatesas:-2^{31},$-2^{31}+1$,...,-2, -1, 0, 1, 2, ..., $2^{31}-2$, $2^{31}-1$

Similarly,intof2bytes,itcantake 2^{16} distinctstatesfrom $-2^{15}$to$2^{15}-1$.If youtrytostore larger number than $2^{31}-1$, i.e,+2147483647 and smaller number than -2^{31}, i.e, -2147483648,program will not run correctly.

Floating types

Variablesoffloatingtypescanholdrealvalues(numbers)suchas:2.34,-9.382etc. Keywords either float or double is used fordeclaring floating type variable. For example:

```
float var2;
doublevar3;
```

Here,both*var2* and *var3*arefloating typevariables.

InC,floatingvaluescanberepresented inexponentialformaswell.For example:

```
floatvar3=22.442e2
```

Differencebetweenfloatanddouble

Generally the size of float(Single precision float data type) is 4 bytes and that of double(Double precision float data type) is 8 bytes. Floating point variables has a precision of 6 digits whereas the the precision of double is 14 digits.

Note: Precision describes the number of significant decimal places that a floating values carries.

Charactertypes

Keywordcharisused fordeclaringthevariableof charactertype.For example:

```
charvar4='h';
```

Here, *var4*isavariableoftypecharacterwhichisstoringacharacter'h'.

Thesizeofcharis1byte.ThecharacterdatatypeconsistsofASCIIcharacters.Each character is given a specific value. For example:

```
For,'a',value=97
For, 'b', value=98
For, 'A', value=65
For, '&', value=33
For, '2', value=49
```

Hereis thelistof allASCIIcharacters inC language.

Qualifiers

Qualifiersaltersthemeaningofbasedatatypes toyieldanewdatatype.

Size qualifiers:

Sizequalifiersaltersthesizeofbasicdatatype.Thekeywordslongandshortaretwosize qualifiers. For example:

```
longinti;
```

The size of int is either 2 bytes or 4 bytes but, when long keyword is used, that variable will be either 4 bytes of 8 bytes. Learn more about long keyword in C programming. If the larger size ofvariable is not needed then, short keyword can be used in similar manner as long keyword.

Signqualifiers:

Whether a variable can hold only positive value or both values is specified by sign qualifiers. Keywords signed and unsigned are used for sign qualifiers.

```
unsignedinta;
//unsignedvariablecanholdzeroandpositivevaluesonly
```

It is not necessary to define variable using keyword signed because, a variable is signed by default. Sign qualifiers can be applied to only int and char data types. For a intvariable of size 4 bytes it can hold data from -2^{31} to $2^{31}-1$ but, if that variable is defined unsigned, it can hold data from 0 to 2^{32} -1.

Constantqualifiers

Constantqualifierscanbedeclaredwithkeywordconst.Anobjectdeclaredbyconst cannot be modified.

```
constintp=20;
```

Thevalueof*p*cannotbechangedintheprogram.

Volatile qualifiers:

Avariableshouldbedeclaredvolatilewheneveritsvaluecanbechangedbysome external sources outside program. Keyword volatile is used to indicate volatile variable.

CProgrammingInputOutput(I/O)

ANSI standard has defined many library functions for input and output in C language. Functions `printf()`and `scanf()`are the most commonly used to display out and take input respectively. Let us consider an example:

```
#include <stdio.h>      //Thisisneededtorunprintf()function. int
main()
{
    printf("CProgramming");//displaysthecontentinsidequotation return
    0;
}
```

Output

```
C Programming
```

ExplanationofHowthisprogram works

1. Everyprogramstartsfrommain()function.
2. `printf()` isalibraryfunctiontodisplayoutputwhichonlyworksif `#include<stdio.h>`isincludedatthebeginning.
3. Here, `stdio.h`is a header file (standard input output header file) and `#include`is command to paste the code from the header file when necessary. When compiler encounters `printf()` function and doesn't find `stdio.h`header file, compiler shows error.
4. Code`return 0;`indicates the end of program. You can ignore this statement but, it is good programming practice to use `return 0;`.

<u>I/Oofintegersin C</u>

```
#include<stdio.h>in
t main()
{
    int c=5;
    printf("Number=%d",c);
    return 0;
}
```

Output

```
Number=5
```

Inside quotation of `printf()` there, is a conversion format string `"%d"`(for integer). If this conversion format string matches with remaining argument,i.e, *c* in this case, valueof *c* is displayed.

```
#include<stdio.h>in
t main()
{
    intc;
    printf("Enteranumber\n");
    scanf("%d",&c);
    printf("Number=%d",c);
    return 0;
}
```

Output

```
Enteranumber 4
Number=4
```

The`scanf()`functionis used to take input from user. In this program, the user is asked a input and value is stored in variable *c*. Note the '&' sign before *c*. &c denotes the address of *c* and value is stored in that address.

I/Oof floatsinC

```
#include<stdio.h>
int main(){
    floata;
    printf("Entervalue:");
    scanf("%f",&a);
    printf("Value=%f",a);     //%fisusedforfloatsinsteadof%d return
    0;
}
```

Output

```
Entervalue:23.45
Value=23.450000
```

Conversionformatstring"`%f`"isusedforfloatstotakeinputandtodisplayfloating value of a variable.

I/OofcharactersandASCIIcode

```
#include<stdio.h>
int main(){
    charvar1;
    printf("Entercharacter:");
    scanf("%c",&var1);
    printf("Youentered%c.",var1);
    return 0;
}
```

Output

```
Entercharacter:g
You entered g.
```

Conversionformatstring"%c"isusedincaseof characters.

ASCIIcode

When character is typed in the above program, the character itself is not recorded a numeric value(ASCII value) is stored. And when we displayed that value by using "%c", that character is displayed.

```
#include<stdio.h>
int main(){
    charvar1;
    printf("Entercharacter:");
    scanf("%c",&var1);
    printf("Youentered%c.\n",var1);
/*\nprintsthenextline(performsworkofenter).*/ printf("ASCII value
    of %d",var1);
    return0;
}
```

Output

```
Entercharacter:
g
103
```

When,*'g'*is entered,ASCIIvalue 103 isstored instead of*g*.

YoucandisplaycharacterifyouknowASCIIcodeonly.Thisisshownbyfollowing example.

```
#include<stdio.h>
int main(){
    intvar1=69;
    printf("CharacterofASCIIvalue69:%c",var1);
    return 0;
}
```

Output

```
CharacterofASCIIvalue69:E
```

The ASCII value of 'A' is 65, 'B' is 66 and so on to 'Z' is 90. Similarly ASCII value of 'a' is 97, 'b' is 98 and so on to 'z' is 122.

MoreaboutInput/Output offloatsandInteger

VariationsinOutputforintegeranfloats

Integerandfloating-pointscanbedisplayedindifferentformatsinCprogrammingas:

```
#include<stdio.h>in
t main(){
    printf("Case1:%6d\n",9876);
/*Printsthenumberrightjustifiedwithin6columns*/ printf("Case
    2:%3d\n",9876);
/*Printsthenumbertoberightjustifiedto3columnsbut,thereare
4digitssonumberisnotrightjustified*/ printf("Case
    3:%.2f\n",987.6543);
/*Printsthenumberroundedtotwodecimalplaces*/
    printf("Case 4:%.f\n",987.6543);
/*Printsthenumberroundedto0decimalplace,i.e,roundedto integer */
    printf("Case5:%e\n",987.6543);
/*Printsthenumberinexponentialnotation(scientificnotation)*/ return 0;
}
```

Output

```
Case1:9876
Case2:9876
Case3:987.65
Case4:988
Case5:9.876543e+002
```

VariationsinInputforintegerand floats

```
#include<stdio.h>
int main(){
    int a,b;
    floatc,d;
```

```
    printf("Entertwointgers:");
/*Twointegerscanbetakenfromuseratonceasbelow*/
    scanf("%d%d",&a,&b);
    printf("Enterintgerandfloatingpointnumbers:");
/*Integerandfloatingpointnumbercanbetakenatoncefromuseras below*/
    scanf("%d%f",&a,&c);
    return 0;
}
```

Similarly,anynumberofinputcan betakenatoncefrom user.

CProgrammingOperators

Operatorsarethesymbolwhichoperatesonvalueoravariable.Forexample:+isa operator to perform addition.

C programming language has wide range of operators to perform various operations. For better understanding of operators, these operators can be classified as:

OperatorsinC programming
ArithmeticOperators
IncrementandDecrementOperators
AssignmentOperators
RelationalOperators
LogicalOperators
ConditionalOperators
BitwiseOperators
SpecialOperators

ArithmeticOperators

Operator	**Meaningof Operator**
+	additionorunaryplus
-	subtractionorunary minus
*	multiplication
/	division
%	remainderafterdivision(modulodivision)

Exampleofworkingof arithmeticoperators

```
/*Programtodemonstratetheworkingofarithmeticoperatorsin C.*/ #include
<stdio.h>
intmain(){
    inta=9,b=4,c;
    c=a+b;
    printf("a+b=%d\n",c);
    c=a-b;
    printf("a-b=%d\n",c);
    c=a*b;
    printf("a*b=%d\n",c);
    c=a/b;
    printf("a/b=%d\n",c);
    c=a%b;
    printf("Remainderwhenadividedbyb=%d\n",c);
    return 0;
}
a+b=13
a-b=5
a*b=36
a/b=2
Remainderwhenadividedby b=1
```

Explanation

Here,theoperators+, -and*performednormallyasyouexpected.Innormalcalculation, `9/4`equals to 2.25. But, the output is 2 in this program. It is because, a and b are both integers. So, the output is also integer and the compiler neglects the term after decimal pointandshows answer 2 instead of 2.25. And, finally `a%b`is 1,i.e. ,when `a=9`is divided by `b=4`, remainder is 1.

```
Supposea=5.0,b=2.0,c=5andd=2 In C
programming,
a/b=2.5
a/d=2.5
c/b=2.5
c/d=2
```

Note:%operatorcanonlybeusedwithintegers.

Incrementanddecrementoperators

InC,++and--are called increment and decrement operators respectively. Both of these operators are unary operators, i.e, used on single operand. ++adds 1 to operand and -- subtracts 1 to operand respectively. For example:

```
Let a=5 and b=10
a++;//abecomes6 a--
;//abecomes5
```

```
++a;//abecomes6
--a;//abecomes5
```

Differencebetween++ and--operatoraspostfixandprefix

When`i++`isused as prefix(like: `++var`), `++var`will increment the value of *var* and then return it but, if `++`is used as postfix(like: var++), operator will return the value of operand first and then only increment it. This can be demonstrated by an example:

```
#include<stdio.h>
int main(){
    intc=2,d=2;
    printf("%d\n",c++);//thisstatementdisplays2then,onlyc incremented by 1
to 3.
    printf("%d",++c);    //this statement increments 1 to c then, only c
is displayed.
    return0;
}
```

Output

```
2
4
```

AssignmentOperators

The most common assignment operator is `=`. This operator assigns the value in right side to the left side. For example:

```
var=5//5isassignedtovar
a=c;    //valueofcisassignedtoa 5=c;
        // Error! 5 is a constant.
```

Operator	Example	Same as
=	a=b	a=b
+=	a+=b	a=a+b
-=	a-=b	a=a-b
*=	a*=b	a=a*b
/=	a/=b	a=a/b
%=	a%=b	a=a%b

RelationalOperator

Relational operators checks relationship between two operands. If the relation is true, itreturns value 1 and if the relation is false, it returns value 0. For example:

```
a>b
```

Here,`>`is a relationaloperator.If*a*is greaterthan*b*, *a*>*b*returns1 ifnotthen, itreturns0

Relationaloperatorsareused indecision makingandloopsinC programming.

Operator	Meaningof Operator	Example
==	Equalto	5==3returnsfalse(0)
>	Greaterthan	5>3returnstrue(1)
<	Lessthan	5<3returnsfalse(0)
!=	Notequalto	5!=3returns true(1)
>=	Greaterthanorequal to	5>=3returnstrue(1)
<=	Lessthanorequalto	5<=3returnfalse(0)

LogicalOperators

Logicaloperatorsareusedtocombineexpressionscontainingrelationoperators.InC, there are 3 logical operators:

Operator	Meaning of Operator	Example
&&	LogialAND	Ifc=5andd=2then,((c==5)&&(d>5))returns false.
‖	LogicalOR	Ifc=5andd=2then,((c==5)‖(d>5))returnstrue.
!	LogicalNOT	Ifc=5then,!(c==5)returnsfalse.

Explanation

For expression, *((c==5) && (d>5))* to be true, both *c==5* and *d>5* should be true but, (d>5)isfalseinthegivenexample.So,theexpressionisfalse.Forexpression `((c==5)||(d>5))` to be true, either the expression should be true. Since, `(c==5)` is true. So, the expression is true. Since, expression `(c==5)` is true, `!(c==5)` is false.

ConditionalOperator

Conditionaloperatortakesthreeoperandsandconsistsoftwosymbols?and:. Conditional operators are used for decision making in C. For example:

```
c=(c>0)?10:-10;
```

If*c*is greaterthan 0, valueof *c*will be10 but, if*c*is lessthan 0, value of*c*will be-10.

BitwiseOperators

Abitwiseoperatorworksoneachbitofdata.Bitwiseoperatorsareusedinbitlevel programming.

Operators	**Meaningof operators**
&	Bitwise AND
|	Bitwise OR
^	Bitwiseexclusive OR
~	Bitwisecomplement
<<	Shiftleft
>>	Shift right

Bitwise operator is advance topic in programming . Learn more about bitwise operator inC programming.

OtherOperators

CommaOperator

Commaoperatorsareusedtolinkrelatedexpressionstogether.For example:

```
inta,c=5,d;
```

Thesizeof operator

Itisaunaryoperatorwhichisusedinfindingthesizeofdatatype,constant,arrays, structure etc. For example:

```
#include<stdio.h>
int main(){
    int a;
    float b;
    doublec;
    char d;
    printf("Size of int=%d bytes\n",sizeof(a));
    printf("Size of float=%d bytes\n",sizeof(b));
    printf("Sizeofdouble=%dbytes\n",sizeof(c));
    printf("Size of char=%d byte\n",sizeof(d));
    return 0;
}
```

Output

```
Size of int=4 bytes
Size of float=4 bytes
Sizeofdouble=8bytes
Size of char=1 byte
```

Conditionaloperators(?:)

ConditionaloperatorsareusedindecisionmakinginCprogramming,i.e,executesdifferent statements according to test condition whether it is either true or false.

Syntaxofconditionaloperators

```
conditional_expression?expression1:expression2
```

Ifthetestconditionistrue,`expression1` isreturnedandiffalse`expression2` is returned.

Exampleofconditionaloperator

```
#include<stdio.h>
int main(){
   charfeb;
   intdays;
   printf("Enterliftheyearisleapyearotherwiseenter0:"); scanf("%c",&feb);
   days=(feb=='l')?29:28;
   /*Iftestcondition(feb=='l')istrue,dayswillbeequalto29. */
```

```
   /*Iftestcondition(feb=='l')isfalse,dayswillbeequalto28.
*/
   printf("NumberofdaysinFebruary=%d",days); return
   0;
}
```

Output

```
Enterliftheyearisleapyearotherwiseentern:l Number of
days in February = 29
```

Other operators suchas&(reference operator),*(dereference operator) and->(member selection) operator will be discussed in pointer chapter.

CProgrammingIntroductionExamples

This page contains example and source code on very basic features of C programming language. To understand the examples on this page, you should have knowledge of following topics:

1. VariablesandConstants
2. DataTypes
3. InputandOutputinCprogramming
4. Operators

CProgrammingIntroductionExamples

1.CProgramtoPrint aSentence

Tounderstandthisexample,youshouldhaveknowledgeoffollowing C programming topics:

CProgrammingInputOutput (I/O)

SourceCode

```
/*CProgramtoprintasentence.*/
#include <stdio.h>
intmain()
{
   printf("CProgramming");/*printf()printsthecontentinside quotation */
   return0;
}
```

Output

```
C Programming
```

CProgrammingIntroductionExamples

EveryCprogramstartsexecutingcodefrom`main()` function.Insidemain(),thereisa `printf()` functionwhichprintsthecontentinsidethequotationmarkwhichis"C Programming"

in this case.

2.ProgramtoPrintaIntegerEnteredbyaUser

Tounderstandthisexample,youshouldhaveknowledgeoffollowing C programming topics:

SourceCode

```
#include<stdio.h>
int main()
{
    int num;
    printf("Enterainteger:");
    scanf("%d",&num);/* Storing a integer entered by user in variable
num */
    printf("Youentered:%d",num);
    return 0;
}
```

Output

```
Enterainteger:25 You
entered: 25
```

In this program, a variable *num* is declared of type integer using keyword **int**. The `printf()` function prints the content inside quotation mark which is "Enter a integer: ". Then, the `scanf()` takes integer value from user and stores it in variable *num*. Finally, the value entered by user is displayed in the screen using `printf()`.

3.CProgramtoAddTwoIntegersEnteredbyUser

In this program, user is asked to enter two integers and this program will add these twointegers and display it.

SourceCode

```
/*C programming source code to add and display the sum of two integers
entered by user */

#include<stdio.h>
int main( )
```

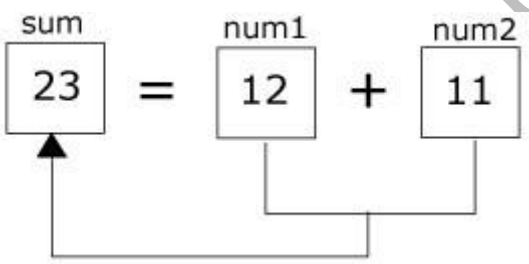
CProgrammingIntroductionExamples

```
{
    int num1, num2, sum;
    printf("Entertwointegers:");
    scanf("%d%d",&num1,&num2);/*Storesthetwointegerenteredby user in
variable num1 and num2 */

    sum=num1+num2;       /* Performs addition and stores it in variable
sum */
    printf("Sum:%d",sum);/*Displayssum*/ return 0;
}
```

Output

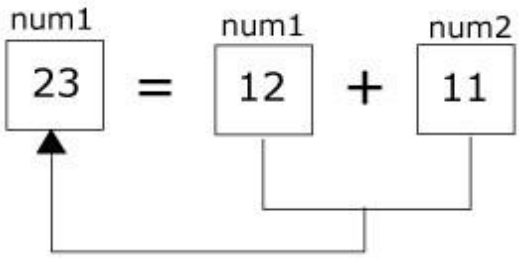
```
Entertwointegers:12 11
Sum: 23
```

In this program, user is asked to enter two integers. The two integers entered by user will bestoredin variables *num1* and *num2* respectively. This is done using `scanf()`function. Then, + operator is used for adding variables *num1* and *num2* and this value is assignedto variable sum.



Finally,thesum isdisplayedand programis terminated.

There are three variables used for performing this task but, you can perform this same taskusing 2 variables only. It can done by assiging the sum of `num1`and `num2`to one of these variables as shown in figure below.



CProgrammingIntroductionExamples

```
/* C programming source code to add and display the sum of two integers
entered by user using two variables only. */

#include<stdio.h>
int main( )
{
    int num1, num2, sum;
    printf("Enteratwointegers:");
    scanf("%d %d",&num1,&num2);
    num1=num1+num2;/*Addsvariablesnum1andnum2andstoresitin num1 */
    printf("Sum:%d",num1);/*Displaysvalueofnum1*/ return
    0;
}
```

Thissourcecodeabovecalculatesthesumoftwointegersanddisplays usingonlytwo variables

4.CProgramtoMultiplytwoFloatingPointNumbers

In this program, user is asked to enter two floating point numbers and this program willmulitply these two numbers and display it.

SourceCode

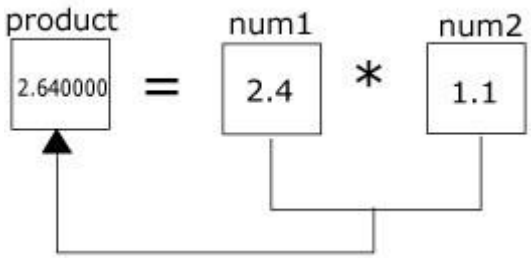
```
/*Cprogramtomultiplyanddisplaytheproductoftwofloatingpoint numbers
entered by user. */

#include<stdio.h>
int main( )
{
    float num1, num2, product;
    printf("Entertwonumbers:");
    scanf("%f %f",&num1,&num2);         /* Stores the twofloating point
numbers entered by user in variable num1 and num2 respectively */
    product=num1*num2;/*Performsmultiplicationandstoresit*/
    printf("Product: %f",product);
    return0;
}
```

Output

```
Entertwonumbers:2.4 1.1
Product:2.640000
```

In this program, user is asked to enter two floating point numbers. These two numbers entered by user will be stored in variables *num1* and *num2* respectively. This is done using`scanf()`function.Then,*operatorisusedformultiplyingvariablesandthisvalue

isstoredinvariable`product`.



Then,theproductisdisplayedandprogramisterminated.

5.CProgramtoFindASCIIValueofCharacterEnteredbyUser

Every character in C programming is given an integer value to represent it. That integer value is known as ASCII value of that character. For example: ASCII value of 'a' is 97. Here is the complete list of ASCII value of characters in C programming. When a character is stored in variable of type **`char`**, the ASCII value of character is stored instead of that character itself character itself. For example: If you try to store character 'a' in a char type variable, ASCII value of that character is stored which is 97.

In,thisprogramuserisaskedtoenteracharacterandthisprogramwilldisplaytheASCII value of that character.

SourceCode

```
/*SourcecodetofindASCIIvalueofacharacterenteredbyuser */

#include<stdio.h>
int main(){
    char c;
    printf("Enteracharacter:");
    scanf("%c",&c);        /*Takesacharacterfromuser*/
    printf("ASCII value of %c = %d",c,c);
    return0;
}
```

Output

```
Enter a character: G
ASCIIvalueofG=71
```

In this program, user is asked to enter a character and this character will be stored in variable *c*, i.e., the ASCII value of that character is stored in variable *c*. When, this value is displayed usingconversion format string %c, theactual variable is displayed but, when this variable is displayed using format string %d, the ASCII value of that character is displayed.

CProgrammingIntroductionExamples

6.CProgramtoFindQuotientandRemainderofTwoIntegersEnteredbyUser

Inthisprogram,userisaskedtoentertwointegers(dividendanddivisor)andthisprogram will compute the quotient and remainder and display it.

SourceCode

```
/*CProgramtocomputeremainderandquotient*/ #include

<stdio.h>
intmain(){
    intdividend,divisor,quotient,remainder;
    printf("Enter dividend: ");
    scanf("%d",&dividend);
    printf("Enterdivisor:");
    scanf("%d",&divisor);
    quotient=dividend/divisor;            /*Computesquotient*/
    remainder=dividend%divisor;/*Computesremainder*/
    printf("Quotient = %d\n",quotient);
    printf("Remainder=%d",remainder);
    return 0;
}
```

Output

```
Enterdividend:25
Enterdivisor:4
Quotient=6
Remainder=1
```

Explanation

This program takes two integers(dividend and divisor) from user and stores it in variable *dividend* and *divisor*. Then, quotient and remainder is calculated and stored in variable *quotient* and *remainder*. Operator / is used for calculation of quotient and % is used for calculatingremainder.Learnmoreabout divison(/)andmodulodivision(%)operatorinCprogramming

Youcanalso programcanbeperformedusingonly twovariables as:

```
/* C Program to compute and display remainder and quotientusing only two
variables */

#include<stdio.h>
int main(){
  int dividend, divisor;
  printf("Enterdividend:");
  scanf("%d",&dividend);
```

```
  printf("Enterdivisor:");
  scanf("%d",&divisor);
  printf("Quotient=%d\n",dividend/divisor);/*Computesanddisplays
quotient */
  printf("Remainder=%d",dividend%divisor);/*Computesanddisplays
remainder */
  return0;
}
```

Output of this program is same as program above but, only two variables are used in this case instead of four variables.

7.CProgramtoFindSizeofint,float,doubleandcharofYour System

The size of a character is always 1 byte but, size of int, float and double variables differs from system to system. This program will compute the size of int, float, double and char of you system using sizeof operator. The syntax of size of operator is:

```
temp=sizeof(operand);
/*Here,tempisavariableoftypeinteger,i.e,sizeof()operator returns integer
   value. */
```

SourceCode

```
/*Thisprogramcomputesthesizeofvariableusingsizeofoperator.*/ #include

<stdio.h>
intmain(){
    int a;
    float b;
    doublec;
    char d;
    printf("Size of int: %d bytes\n",sizeof(a));
    printf("Size of float: %d bytes\n",sizeof(b));
    printf("Sizeofdouble:%dbytes\n",sizeof(c));
    printf("Size of char: %d byte\n",sizeof(d));
    return 0;
}
```

Output

```
Size of int: 4 bytes
Size of float: 4 bytes
Sizeofdouble:8bytes
Size of char: 1 byte
```

Note: Youmay getdifferentoutput dependinguponyour system.

CProgrammingIntroductionExamples

In this program, 4 variables *a*, *b*, *c* and *d* are declared of type int, float, double and char respectively. Then, the size of these variables is computed using sizeof operator and displayed.

8.CProgramtoDemonstratetheWorkingofKeyword long

Keyword long is used for altering the size of data type. For example: the size of `int`is either 2 bytes or 4 bytes but, when long keyword is used, the size of `long int`will be either4 bytesor8 bytes.Also, you can use`longlongint`. Thesizeof `longlongint` is generally 8 bytes. This program will demonstrate the size of keyword long for my system. It may be different in your system.

SourceCode

```
#include<stdio.h>
int main(){
    inta;
    longintb;                   /*intisoptional. */
    long long int c;             /* int is optional. */
    printf("Size of int = %d bytes\n",sizeof(a));
    printf("Size of long int = %ld bytes\n",sizeof(b));
    printf("Sizeoflonglongint=%ldbytes",sizeof(c)); return
    0;
}
```

Output

```
Sizeofint=4bytes
Sizeoflongint=4bytes
Sizeoflonglongint=8bytes
```

Inthisprogram,thesizeofoperatorisusedforfindingthesizeof`int`,`longint`and `longlongint`.

Thus,`int`and `long int`for my system can hold values from -2^{31} to $2^{31}-1$. If I have to work on data outside this range, I have to use `long long int`, which can hold values from -2^{63} to $2^{63}-1$.

Similarly,thelong keywordcan beuseddouble andfloats types.

9.CProgramtoSwapTwonumbersEnteredbyUser

Thisprogramasksusertoentertwonumbersandthisprogramwillswapthevalueof

thesetwo numbers.

SourceCodetoSwapTwoNumbers

```
#include<stdio.h>
int main(){
      float a, b, temp;
      printf("Entervalueofa:");
      scanf("%f",&a);
      printf("Entervalueofb:");
      scanf("%f",&b);
      temp=a;      /*Valueofaisstoredinvariabletemp*/
      a=b;         /*Valueofbisstoredinvariablea*/
      b = temp;     /* Value of temp(which contains initial value of a)
is stored in variable b*/
      printf("\nAfterswapping,valueofa=%.2f\n",a);
      printf("After swapping, value of b = %.2f", b);
      return 0;
}
```

Output

```
Entervalueofa:1.20
Entervalueofb: 2.45

Afterswapping,valueofa=2.45 After
swapping, value of b = 1.2
```

CProgrammingbreakandcontinueStatement

There are two statements built in C programming, `break;`and `continue;`to alter the normal flow of a program. Loops perform a set of repetitive task until text expression becomes false but it is sometimes desirable to skip some statement/s inside loop or terminatethe loop immediately without checking thetest expression. Insuch cases, break andcontinuestatementsareused. The `break;`statement is also used in `switch`statement to exit switch statement.

break Statement

InCprogramming,breakisusedinterminatingtheloopimmediatelyafteritis encountered. The break statement is used with conditional if statement.

Syntaxofbreakstatement

```
break;
```

The break statement can be used in terminating all three loops for, while and do...whileloops.

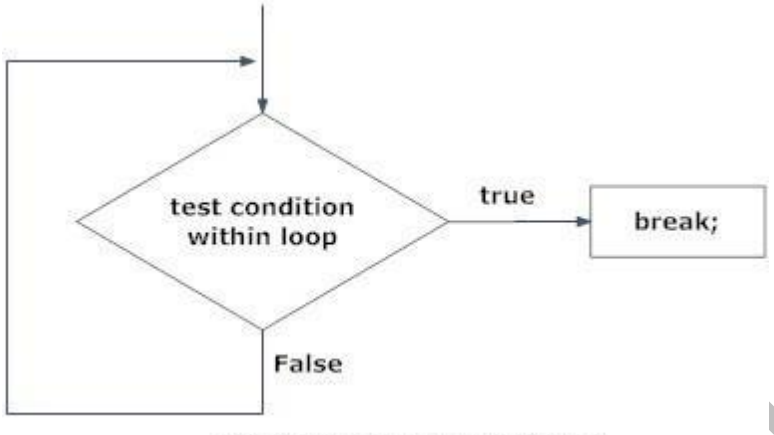


Figure: Flowchart of break statement

Thefigurebelow explainsthe working ofbreakstatement in allthreetypeof loops.

```
while (test expression) {
    statement/s
    if (test expression) {
        break;
    }
    statement/s
}
```

```
do {
    statement/s
    if (test expression) {
        break;
    }
    statement/s
}
while (test expression);
```

```
for (intial expression; test expression; update expression) {
    statement/s
    if (test expression) {
        break;
    }
    statements/
}
```

NOTE: The break statment may also be used inside body of else statement.

<u>Exampleofbreakstatement</u>

Write a C program to find average of maximum of *n* positive numbers entered by user. But, if the input is negative, display the average(excluding the average of negative input) and end the program.

```
/*Cprogramtodemonstratetheworkingofbreakstatementby terminating a loop,
if user inputs negative number*/
#include<stdio.h>
int main(){
   floatnum,average,sum;
   int i,n;
   printf("Maximumno.ofinputs\n"); scanf("%d",&n);
   for(i=1;i<=n;++i)
{
       printf("Entern%d:",i);
       scanf("%f",&num);
       if(num<0.0)
       break;                      //forloopbreaksifnum<0.0
       sum=sum+num;
}
  average=sum/(i-1);
  printf("Average=%.2f",average);
  return 0;
}
```

Output

```
Maximumno.ofinputs 4
Entern1: 1.5
Entern2:12.5
Entern3: 7.2
Entern4:-1
Average=7.07
```

In this program, when the user inputs number less than zero, the loop is terminated using break statement with executing the statement below it i.e., without executing `sum=sum+num`.

In C, break statements are also used in switch...case statement. You will study it in Cswitch...case statement chapter.

continue Statement

It is sometimes desirable to skip some statements inside the loop. In such cases, continue statements are used.

SyntaxofcontinueStatement

```
continue;
```

Justlikebreak, continue is alsousedwith conditionalif statement.

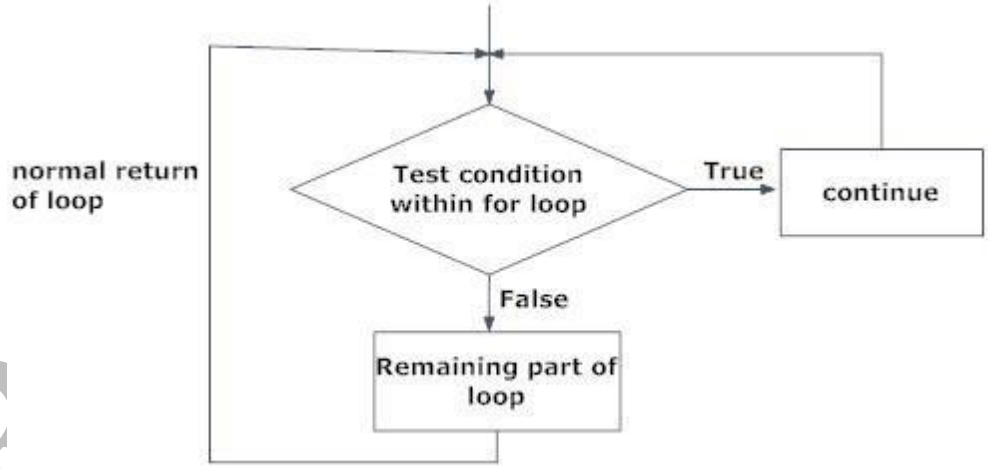


Fig: Flowchart of continue statement

For better understanding of how continue statements works in C programming. Analyze the figure below which bypasses some code/s inside loops using continue statement.

```
while (test expression) {
    statement/s
    if (test expression) {
        continue;
    }
    statement/s
}
```

```
do {
    statement/s
    if (test expression) {
        continue;
    }
    statement/s
}
while (test expression);
```

```
for (intial expression; test expression; update expression) {
    statement/s
    if (test expression) {
        continue;
    }
    statements/
}
```

NOTE: The continue statment may also be used inside body of else statement.

<u>Exampleofcontinuestatement</u>

Writea Cprogram tofind theproductof4 integers entered by a user. Ifuser enters 0 skip it.

```
//programtodemonstratetheworkingofcontinuestatementinC programming
#include<stdio.h>
int main(){
    int i,num,product;
    for(i=1,product=1;i<=4;++i){
        printf("Enternum%d:",i);
        scanf("%d",&num);
        if(num==0)
            continue;/ *In this program, when num equals to zero, it
skips the statement product*=num and continue the loop. */
        product*=num;
}
    printf("product=%d",product);
return0;
}
```

Output

```
Enter num1:3
Enter num2:0
Enternum3:-5
Enter num4:2
product=-30
```

CProgrammingswitchStatement

Decision making are needed when, the program encounters the situation to choose a particular statement among many statements. If a programmer has to choose one block of statement among many alternatives, nested if...else can be used but, this makes programming logic complex. This type of problem can be handled in C programming using `switch`statement.

Syntaxof switch...case

```
switch (n) {
caseconstant1:
   code/stobeexecutedifnequalstoconstant1; break;
caseconstant2:
   code/stobeexecutedifnequalstoconstant2; break;
   .
   .
   .
default:
   code/stobeexecutedifndoesn'tmatchtoanycases;
}
```

The value of *n* is either an integer or a character in above syntax. If the value of *n*matches constant in `case`, the relevant codes are executed and control moves out of the `switch`statement. If the *n* doesn't matches any of the constant in case, then the default codes are executed and control moves out of `switch`statement.

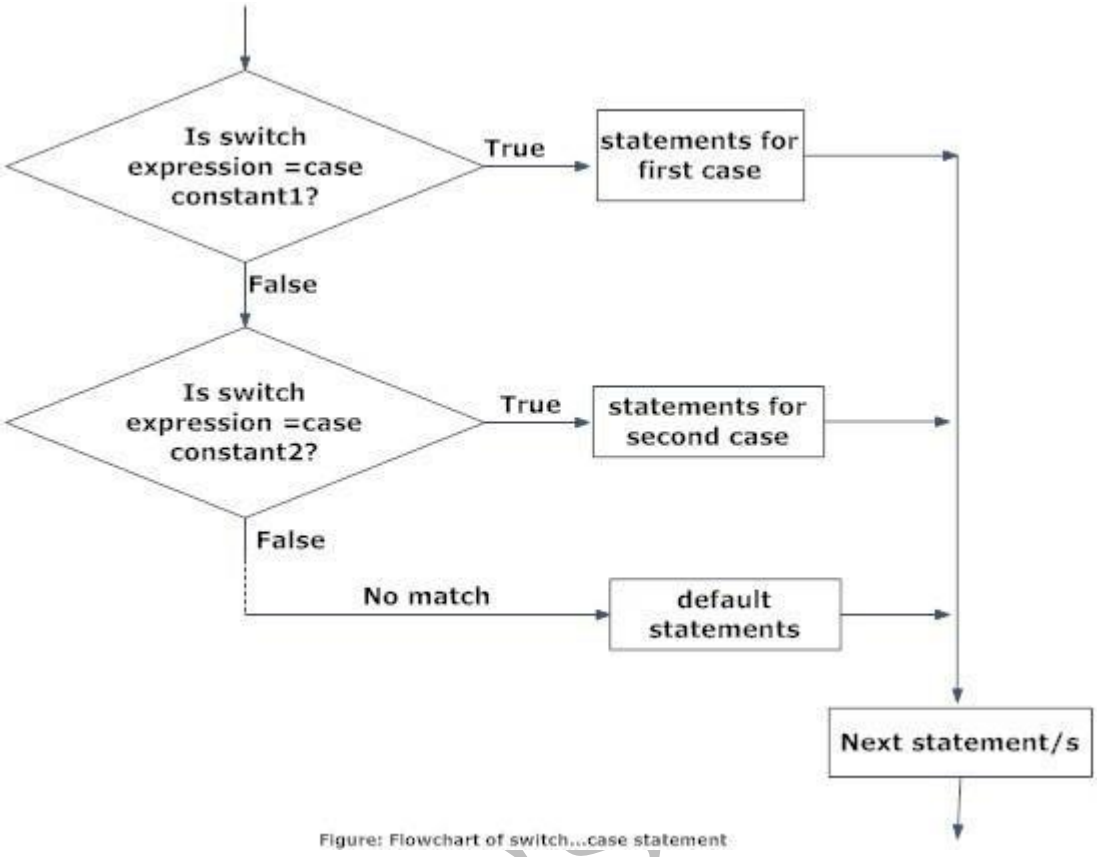


Figure: Flowchart of switch...case statement

<u>Exampleofswitch...casestatement</u>

Writeaprogramthatasksuseranarithmeticoperator('+','-','*'or'/')andtwo operands and perform the corresponding calculation on the operands.

```
/*Cprogramtodemonstratetheworkingofswitch...casestatement*/
/*CProgramtocreateasimplecalculatorforaddition,subtraction, multiplication
   and division */

#include<stdio.h>
int main() {
    char o;
    floatnum1,num2;
    printf("Selectanoperatoreither+or-or*or/\n"); scanf("%c",&o);
    printf("Entertwooperands:");
    scanf("%f%f",&num1,&num2);
    switch(o) {
        case'+':
            printf("%.1f+%.1f=%.1f",num1,num2,num1+num2); break;
        case'-':
            printf("%.1f-%.1f=%.1f",num1,num2,num1-num2); break;
        case'*':
```

```
            printf("%.1f*%.1f=%.1f",num1,num2,num1*num2); break;
        case'/':
            printf("%.1f/%.1f=%.1f",num1,num2,num1/num2); break;
        default:
            /*Ifoperatorisotherthan+,-,*or/,errormessageis
shown*/
            printf("Error!operatorisnotcorrect");
            break;
    }
    return0;
}
```

Output

```
Enteroperatoreither+or-or*or/
*
Entertwooperands:2.3 4.5
2.3*4.5=10.3
```

The`break`statementattheendofeach`case`cause`switch`statementtoexit.Ifbreak statement is not used, all statements below that case statement are also executed.

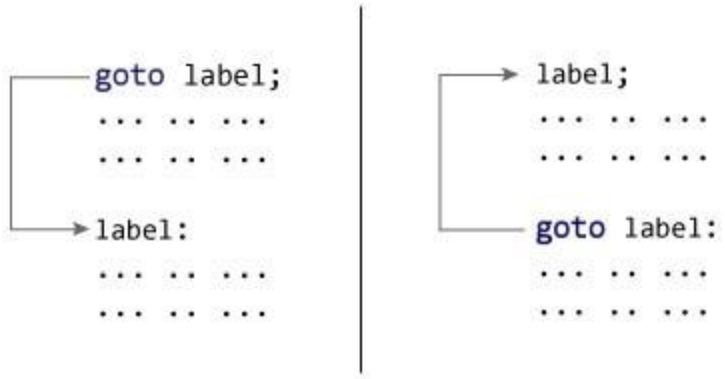
CProgramminggoto Statement

In C programming, goto statement is used for altering the normal sequence of program execution by transferring control to some other part of the program.

Syntaxofgotostatement

```
gotolabel;
.............
.............
.............
label:
statement;
```

In this syntax, label is an identifier. When, the control of program reaches to goto statement, the control of the program will jump to the `label:`and executes the code below it.



Exampleofgotostatement

```
/*Cprogramtodemonstratetheworkingofgotostatement. */
/*Thisprogramcalculatestheaverageofnumbersenteredbyuser.*/
/*Ifuserentersnegativenumber,itignoresthatnumberand calculates
   the average of number entered before it.*/

#include<stdio.h>
int main(){
   floatnum,average,sum;
   int i,n;
   printf("Maximumno.ofinputs:"); scanf("%d",&n);
   for(i=1;i<=n;++i){
       printf("Entern%d:",i);
       scanf("%f",&num);
       if(num<0.0)
       goto jump;              /* control of the program moves to label
jump */
       sum=sum+num;
}
jump:
  average=sum/(i-1);
  printf("Average:%.2f",average);
  return 0;
}
```

Output

```
Maximumno.ofinputs:4
Enter n1: 1.5
Entern2:12.5
Entern3: 7.2
Entern4:-1
Average:7.07
```

Though goto statement is included in ANSI standard of C, use of goto statement should be reduced as much as possible in a program.

Reasonstoavoid gotostatement

Though, using goto statement give power to jump to any part of program, using goto statement makes the logic of the program complex and tangled. In modern programming, goto statement is considered a harmful construct and a bad programming practice.

The goto statement can be replaced in most of C program with the use of break and continue statements. In fact, any program in C programming can be perfectly written without the use of goto statement. All programmer should try to avoid goto statement as possible as they can.

CProgrammingif,if..elseandNestedif...else Statement

The`if`,`if...else`andnested`if...else`statementareusedtomakeone-timedecisions in C Programming, that is, to execute some code/s and ignore some code/s depending upon the test expression.

C if Statement

```
if(testexpression){
       statement/stobeexecutediftestexpressionistrue;
}
```

The`if`statementchecks whetherthetextexpressioninsideparenthesis()istrueornot.If the test expression is true, statement/s inside the body of `if`statement is executed but if

Flowchartofif statement

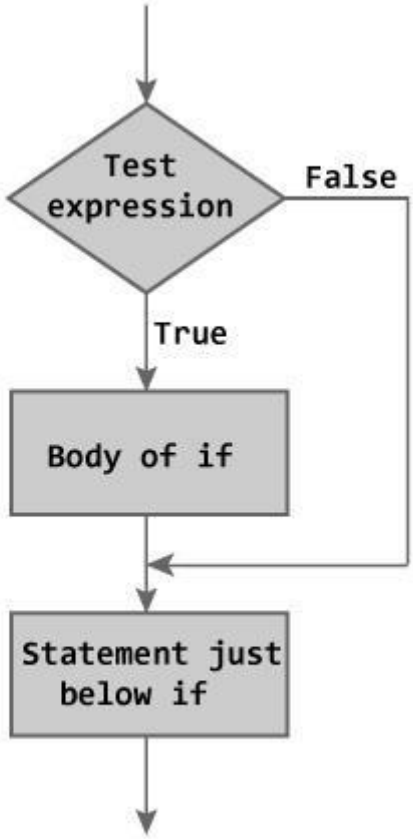


Figure: Flowchart of if Statement

Example1:Cifstatement

Write a C program to print the number entered by user only if the number entered is negative.

```
#include<stdio.h>
      intmain(){
      int num;
      printf("Enteranumbertocheck.\n");
      scanf("%d",&num);
      if(num<0) {         /* checking whether number is less than 0
ornot. */
            printf("Number=%d\n",num);
      }
/*Iftestconditionistrue,statementabovewillbeexecuted, otherwise it will
not be executed */
```

```
      printf("TheifstatementinCprogrammingiseasy."); return 0;
}
```

Output1

```
Enteranumbertocheck.
-2
Number=-2
TheifstatementinCprogrammingiseasy.
```

Whenuser enters -2 then, the test expression `(num<0)` becomes true. Hence, `Number = -2` is displayed in the screen.

Output2

```
Enteranumbertocheck. 5
TheifstatementinCprogrammingiseasy.
```

Whentheuserenters5then,thetestexpression `(num<0)` becomesfalse.So,the statement/s inside body of `if` is skipped and only the statement below it is executed.

Cif...elsestatement

The `if...else` statement is used if the programmer wants to execute some statement/s when the test expression is true and execute some other statement/s if the test expression is false.

Syntaxofif...else

```
if(testexpression){
     statementstobeexecutediftestexpressionistrue;
}
else{
     statementstobeexecutediftestexpressionisfalse;
}
```

Flowchartofif...elsestatement

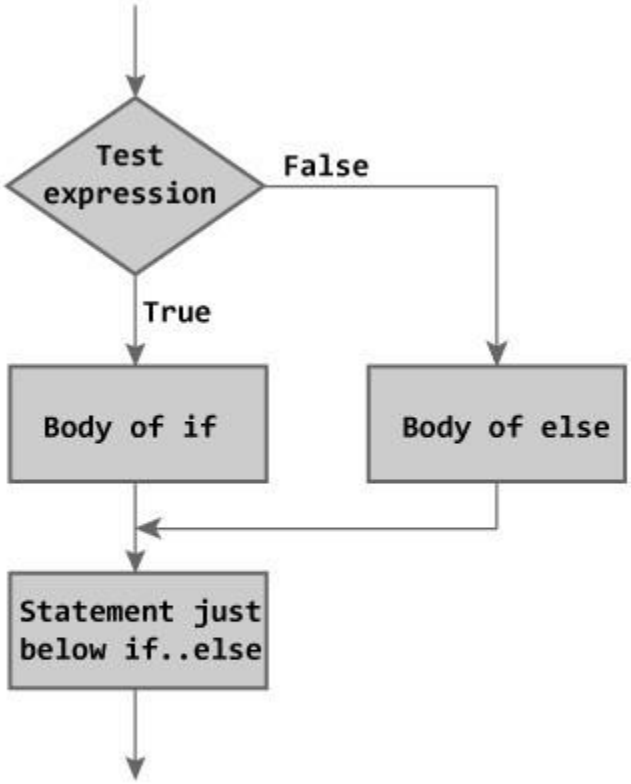


Figure: Flowchart of if...else Statement

<u>Example2:Cif...elsestatement</u>

WriteaCprogram to checkwhetheranumberentered byuserisevenorodd

```
#include<stdio.h>
int main(){
      int num;
      printf("Enteranumberyouwanttocheck.\n"); scanf("%d",&num);
      if((num%2)==0)          //checkingwhetherremainderis0ornot.
           printf("%d is even.",num);
```

```
      else
           printf("%disodd.",num);
      return0;
}
```

Output1

```
Enteranumberyouwanttocheck. 25
25isodd.
```

Output2

```
Enteranumberyouwanttocheck. 2
2is even.
```

Nestedif...elsestatement(if...elseif.....elseStatement)

Thenested`if...else`statementisusedwhenprogramrequiresmorethanonetest expression.

Syntaxofnestedif...elsestatement.

```
if(testexpression1){
     statement/stobeexecutediftestexpression1istrue;
     }
     elseif(testexpression2){
          statement/stobeexecutediftestexpression1isfalseand2
istrue;
     }
     elseif(testexpression3){
         statement/s to be executed if text expression1 and 2 are false
and 3 is true;
     }
          .
          .
          .
     else{
          statementstobeexecutedifalltestexpressionsare
false;
     }
```

Hownestedif...elseworks?

The nested `if...else`statement has more than one test expression. If the first test expression is true, it executes the code inside the braces{ } just below it. But if the first test expression is false, it checks the second test expression. If the second test expression istrue,itexecutesthestatement/sinsidethebraces{}justbelowit.Thisprocess

continues.Ifallthetestexpressionarefalse,code/sinside`else`isexecutedandthe control of program jumps below the `nested if...else`

TheANSIstandard specifiesthat 15 levels ofnesting may becontinued.

Example3:Cnestedifelsestatement

Writea Cprogramtorelatetwointegersenteredby userusing =or>or<sign.

```
#include<stdio.h>
int main(){
     intnumb1,numb2;
     printf("Entertwointegerstocheck\n"); scanf("%d
     %d",&numb1,&numb2);
     if(numb1==numb2)//checkingwhethertwointegersareequal.
          printf("Result: %d = %d",numb1,numb2);
     else
        if(numb1>numb2)//checkingwhethernumb1isgreaterthannumb2.
          printf("Result: %d > %d",numb1,numb2);
        else
          printf("Result:%d>%d",numb2,numb1);
return0;
}
```

Output1

```
Entertwointegerstocheck. 5
3
Result:5>3
```

Output2

```
Entertwointegerstocheck.
-4
-4
Result:-4=-4
```

UNIT-2

CProgrammingforLoop

CProgrammingLoops

Loops cause program to execute the certain block of code repeatedly until test conditionis false. Loops are used in performing repetitive task in programming. Consider these scenarios:

- Youwanttoexecutesomecode/s100times.
- Youwanttoexecutesomecode/scertainnumberoftimesdependinguponinputfrom user.

Thesetypesoftaskcanbesolvedinprogrammingusingloops. There

are 3 types of loops in C programming:

1. forloop
2. while loop
3. do...whileloop

forLoopSyntax

```
for(initializationstatement;testexpression;updatestatement){
       code/s to be executed;
}
```

HowforloopworksinCprogramming?

The initialization statement is executed only once at the beginning of the for loop. Then the test expression is checked by the program. If the test expression is false, for loop is terminated. But if test expression is true then the code/s inside body of `for`loop is executed andthen update expressionisupdated.Thisprocessrepeatsuntiltestexpression is false.

Thisflowchart describesthe workingofforloopin C programming.

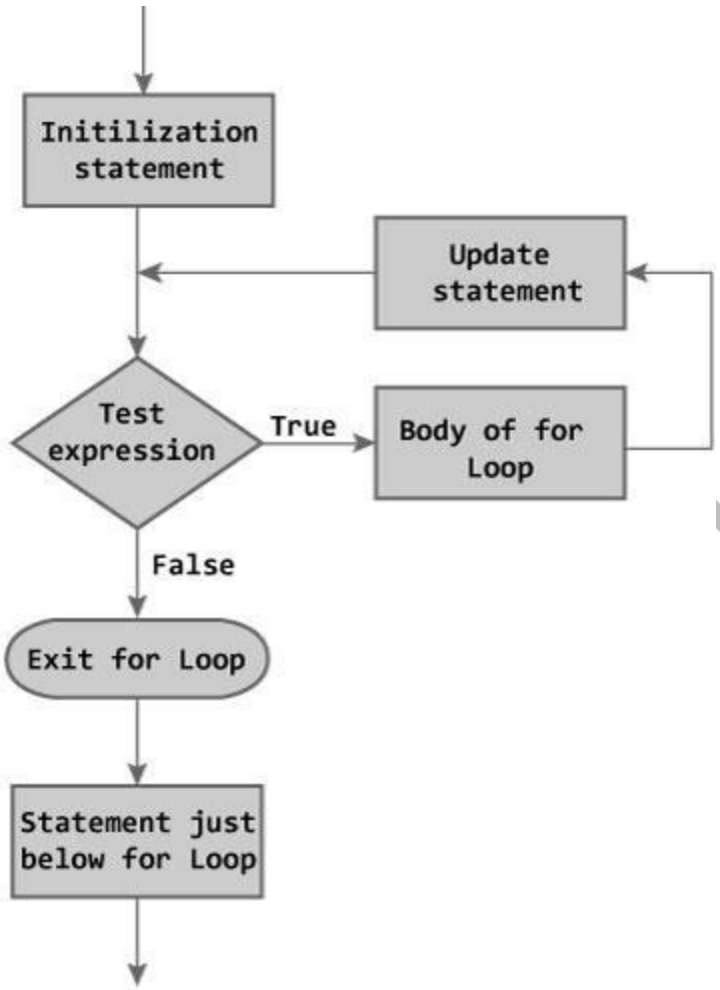


Figure: Flowchart of for Loop

<u>forloopexample</u>

Write a program to find the sum of first*n* natural numbers where nis entered byuser. Note: 1,2,3... are called natural numbers.

```
#include<stdio.h>
int main(){
    intn,count,sum=0;
    printf("Enterthevalueofn.\n"); scanf("%d",&n);
    for(count=1;count<=n;++count)//forloopterminatesifcount>n
    {
        sum+=count;     /*thisstatementisequivalenttosum=sum+count
*/
    }
    printf("Sum=%d",sum);
    return 0;
```

```
}
```

Output

```
Enterthevalueofn. 19
Sum=190
```

In this program, the user is asked to enter the value of *n*. Suppose you entered 19 then, count is initialized to 1 at first. Then, the test expression in the for loop,i.e., `(count<= n)` becomestrue. So, the code in the body of for loop is executed which makes *sum* to 1. Then,theexpression`++count`isexecutedandagainthetestexpressionischecked,which becomes true. Again, the body of for loop is executed which makes *sum* to 3 and this process continues. When count is 20, the test condition becomes false and the for loop is terminated.

Note: Initial,testandupdateexpressionsareseparatedbysemicolon(;).

CprogrammingwhileanddO...whileLoopC programming loops

Loops causes program to execute the certain block of code repeatedly until some conditions are satisfied, i.e., loops are used in performing repetitive work inprogramming.

Suppose you want to execute some code/s 10 times. You can perform it by writing that code/s only one time and repeat the execution 10 times using loop.

Thereare3 types of loops in C programming:

1. forloop
2. while loop
3. do...whileloop

Syntaxofwhileloop

```
while (test expression) {
    statement/stobeexecuted.
}
```

The `while`loop checks whether thetest expression is true or not. If it is true, code/s inside the body of while loop is executed,that is, code/s inside the braces { } areexecuted. Then againthe testexpressionischecked whether testexpression istrueor not. This process continues until the test expression becomes false.

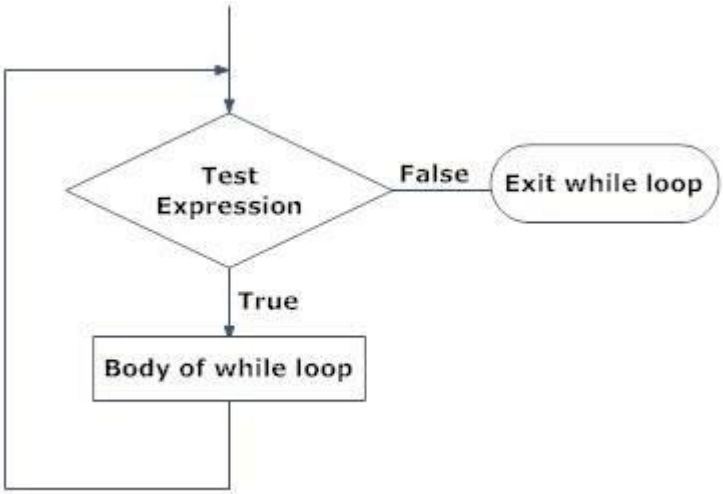


Figure: Flowchart of while loop

Exampleofwhileloop

Write a C program to find the factorial of a number, where the number is enteredby user. (Hints: factorial of `n = 1*2*3*...*n`

```
/*Cprogramtodemonstratetheworkingofwhileloop*/
#include <stdio.h>
     intmain(){
     int number,factorial;
     printf("Enteranumber.\n");
     scanf("%d",&number);
     factorial=1;
     while (number>0){     /* while loop continues util test condition
number>0 is true */
          factorial=factorial*number;
          --number;
}
printf("Factorial=%d",factorial);
return 0;
}
```

Output

```
Enteranumber.
5
Factorial=120
```

do...while loop

InC,do...whileloopisverysimilartowhileloop.Onlydifferencebetweenthesetwo loopsisthat,inwhile loops,testexpressionischeckedatfirstbut,indo...while loopcode

is executed at first then the condition is checked. So, the code are executed at least oncein do...while loops.

Syntaxofdo...whileloops

```
do{
   somecode/s;
}
while(testexpression);
```

At first codes inside body of do is executed. Then, the test expression is checked. If it is true, code/s inside body of do are executed again and the process continues until test expression becomes false(zero).

Notice,thereissemicolon intheend ofwhile ();in do...while loop.

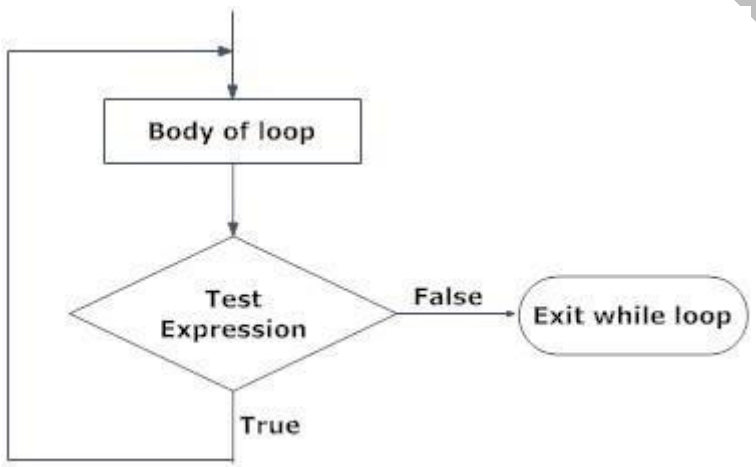


Figure: Flowchart of do...while loop

Exampleofdo...whileloop

WriteaCprogram toaddallthenumbersenteredbyauseruntiluserenters0.

```
/*Cprogramtodemonstratetheworkingofdo...whilestatement*/ #include
<stdio.h>
intmain(){
   intsum=0,num;
   do              /* Codes inside the body of do...while loops are at
least executed once. */
   {
        printf("Enteranumber\n"); scanf("%d",&num);
        sum+=num;
```

```
   }
   while(num!=0);
   printf("sum=%d",sum);
return0;
}
```

Output

```
Enteranumber 3
Enteranumber
-2
Enteranumber 0
sum=1
```

In this C program, user is asked a number and it is added with *sum*. Then, only the test condition in the do...while loop is checked. If the test condition is true,i.e, *num* is not equal to 0, the body of do...while loop is again executed until *num* equals to zero.

CProgrammingDecisionMakingandLoopsExamples

This page contains examples and source code on decision making in C programming (to choose a particular statement among many statements) and loops (to perform repeated task). To understand all the examples on this page, you should have knowledge of following topics:

1. if...elseStatement
2. for Loop
3. whileLoop
4. breakandContinueStatement
5. switch...case

DecisionMakingandLoopExamples

CProgrammingExamplesandSourceCode

CProgramtoCheckWhetheraNumberisEvenorOdd

Tounderstandthisexample,youshouldhaveknowledgeof following Cprogramming topics:

CProgrammingOperators
CProgrammingif,if..elseandNestedif...elseStatement

Numbersperfectlydivisibleby2areknownevennumbersandnumberswhicharenot divisibleby2arecalledoddnumbers.Thisprogramtakesanintegerfromuserand

CProgrammingExamplesandSourceCode

checkswhetherthat numberis even oroddand displays the result.

SourceCodetoCheckWhetheraNumberisEvenorOdd

```
/*Cprogramtocheckwhetheranumberenteredbyuserisevenorodd.
*/

#include<stdio.h>
int main(){
      int num;
      printf("Enteranintegeryouwanttocheck:");
      scanf("%d",&num);
      if((num%2)==0)      /*Checkingwhetherremainderis0ornot.*/
           printf("%d is even.",num);
      else
           printf("%disodd.",num); return
      0;
}
```

Output1

```
Enteranintegeryouwanttocheck: 25
25isodd.
```

Output2

```
Enteranintegeryouwanttocheck: 12
12is even.
```

In this program, user is asked to enter an integer which is stored in variable *num*. Then, the remainder is found when that number is divided by 2 and checked whether remainder is 0 or not. If remainder is 0 then, that number is even otherwise that number is odd. This task is performed using `if...else statement in C programming`and the result is displayed accordingly.

This program also can be solved using conditional operator[**?:**] which is the shorthand notation for if...else statement.

```
/*Cprogramtocheckwhetheranintegerisoddorevenusing conditional operator
*/

#include<stdio.h>
int main(){
    int num;
    printf("Enteranintegeryouwanttocheck:");
```

```
    scanf("%d",&num);
    ((num%2)==0)   ?   printf("%d   is   even.",num)   :   printf("%d   is
odd.",num);
    return0;
}
```

CProgramtoCheckWhetheraCharacterisVowelorconsonant

Alphabets a, e, i, o and u are known as vowels and all alphabets except these characters areknown as consonants.This program asks userto enteracharacterand checks whether that character isvowel or not.

SourceCodetoCheckWhetheraCharacterisVowelorconsonant

```
#include<stdio.h>
int main(){
  char c;
  printf("Enteranalphabet:");
  scanf("%c",&c);

if(c=='a'||c=='A'||c=='e'||c=='E'||c=='i'||c=='I'||c=='o'||c=='O'||c=='
u'||c=='U')
       printf("%cisavowel.",c);
  else
       printf("%cisaconsonant.",c); return
  0;
}
```

Output1

```
Enteranalphabet:i i
is a vowel.
```

Output2

```
Enteranalphabet:G G
is a consonant.
```

In this program, user is asked to enter a character which is stored in variable *c*. Then, this character is checked, whether it is any one of these ten characters a, A, e, E, i, I, o, O, u andUusinglogicalORoperator||. Ifthatcharacterisanyoneofthesetencharacters,that alphabet is a vowel if not that alphabet is a consonant.

This program also can be solved using conditional operator which is shorthand notation for if else statement.

```
/*Cprogramtocheckwhetheracharacterisvowelorconsonantusing conditional
operator */
```

```
#include<stdio.h>
int main(){
 char c;
 printf("Enteranalphabet:");
 scanf("%c",&c);

(c=='a'||c=='A'||c=='e'||c=='E'||c=='i'||c=='I'||c=='o'||c=='O'||c=='u'
||c=='U')   ?   printf("%c   is   a   vowel.",c)   :   printf("%c   is   a
consonant.",c);
  return0;
```

CProgramtoFindtheLargestNumberAmongThreeNumbersEnteredbyUser

In this program useris asked to enterthreenumbersand this program will findthelargest number among three numbers entered by user. This program can be solved in more than one way.

SourceCode1

```
/*Cprogramtofindlargestnumberusingifstatementonly*/ #include

<stdio.h>
intmain(){
      floata,b,c;
      printf("Enterthreenumbers:");
      scanf("%f %f %f", &a, &b, &c);
      if(a>=b && a>=c)
         printf("Largestnumber=%.2f",a); if(b>=a
      && b>=c)
         printf("Largestnumber=%.2f",b); if(c>=a
      && c>=b)
         printf("Largestnumber=%.2f",c); return
      0;
}
```

SourceCode2

```
/*Cprogramtofindlargestnumberusingif...elsestatement*/

#include<stdio.h>
int main(){
      floata,b,c;
      printf("Enterthreenumbers:");
      scanf("%f %f %f", &a, &b, &c);
      if (a>=b)
      {
          if(a>=c)
            printf("Largestnumber=%.2f",a);
          else
```

```
            printf("Largestnumber=%.2f",c);
      }
      else
      {
          if(b>=c)
            printf("Largestnumber=%.2f",b);
          else
            printf("Largestnumber=%.2f",c);
      }
      return0;
}
```

SourceCode3

```
/*CProgramtofindlargestnumberusingnestedif...elsestatement*/

#include<stdio.h>
int main(){
      floata,b,c;
      printf("Enterthreenumbers:");
      scanf("%f %f %f", &a, &b, &c);
      if(a>=b && a>=c)
         printf("Largestnumber=%.2f",a); else
      if(b>=a && b>=c)
         printf("Largestnumber=%.2f",b); else
         printf("Largestnumber=%.2f",c); return
      0;
}
```

Though the technique to solve this problem is different in these three examples, output of all these program is same.

```
Enterthreenumbers:12.2 13.452
10.193
Largestnumber=13.45
```

CprogramtoFindallRootsofaQuadratic equation

Suppose you want of find roots of a quadratic equation $ax^2+bx+c=0$ where a, b and c are coefficients. This program will ask the coefficients: a, b and c from user and displays the roots.

SourceCodetoFindRootsofQuadraticEquation

```
/*CProgramtofindrootsofaquadraticequationwhencoefficients are entered by
user. */
/*Libraryfunctionsqrt()computesthesquareroot. */

#include<stdio.h>
#include<math.h>/*Thisisneededtousesqrt()function.*/
intmain()
```

```
{
  floata,b,c,determinant,r1,r2,real,imag; printf("Enter
  coefficients a, b and c: "); scanf("%f%f%f",&a,&b,&c);
  determinant=b*b-4*a*c;
  if (determinant>0)
  {
      r1=(-b+sqrt(determinant))/(2*a);
      r2=(-b-sqrt(determinant))/(2*a);
      printf("Rootsare:%.2fand%.2f",r1,r2);
  }
  elseif(determinant==0)
  {
    r1=r2=-b/(2*a);
    printf("Rootsare:%.2fand%.2f",r1,r2);
  }
  else
  {
    real=-b/(2*a);
    imag=sqrt(-determinant)/(2*a);
    printf("Rootsare:%.2f+%.2fiand%.2f-%.2fi",real,imag,real, imag);
  }
  return0;
}
```

Output1

```
Entercoefficientsa,bandc:2.3 4
5.6
Rootsare:-0.87+1.30iand-0.87-1.30i
```

Output2

```
Entercoefficientsa,bandc:4 1
0
Rootsare:0.00and-0.25
```

Tosolvethisprogram,libraryfunction`sqrt()`isused.Thisfunctioncalculatesthe square root of a number. Learn more about,sqrt() function.

CProgramtoCheckWhethertheEnteredYearisLeapYearornot

All years which are perfectly divisible by 4 are leap years except for century years(years ending with 00) which is a leap year only it is perfectly divisible by 400. For example: 2012,2004,1968etcareleapyearbut,1971,2006etcarenotleapyear.Similarly,1200, 1600,2000, 2400 areleapyears but,1700, 1800,1900 etc arenot.

CProgrammingExamplesandSourceCode

This program asks user to enter a year and this program checks whether that year is leap year or not.

SourceCodetoCheckLeapYear

```
/* C program to check whether a year is leap year or not using if else
statement.*/

#include<stdio.h>
int main(){
      intyear;
      printf("Enterayear:");
      scanf("%d",&year);
      if(year%4 == 0)
      {
          if(year%100==0)/*Checkingforacenturyyear*/
          {
              if(year%400==0)
                 printf("%disaleapyear.",year); else
                 printf("%disnotaleapyear.",year);
          }
          else
             printf("%disaleapyear.",year );
      }
      else
         printf("%disnotaleapyear.",year); return 0;
}
```

Output2

```
Enteryear:1900
1900isnotaleapyear.
```

Output3

```
Enter year: 2012
2012isaleapyear.
```

CProgramtoCheckWhetheraNumberisPositiveorNegativeor Zero.

Thisprogramtakesanumberfromuser andchecks whetherthatnumberiseitherpositive or negative or zero.

SourceCode

CProgrammingExamplesandSourceCode

```
#include<stdio.h>
int main()
{
    floatnum;
    printf("Enteranumber:");
    scanf("%f",&num);
    if(num<=0)
    {
        if(num==0)
          printf("Youenteredzero.");
        else
          printf("%.2fisnegative.",num);
    }
    else
      printf("%.2fispositive.",num);
    return 0;
}
```

Thisprogram alsocan besolvedusing<u>nestedif elsestatement</u>.

```
/* C programming code to check whether a number is negative or positive
or zero using nested if...else statement. */

#include<stdio.h>
int main()
{
    floatnum;
    printf("Enteranumber:");
    scanf("%f",&num);
    if (num<0)      /*Checkingwhethernumislessthan0*/
      printf("%.2f is negative.",num);
    else if (num>0)   /*Checkingwhethernumisgreaterthanzero*/
      printf("%.2f is positive.",num);
    else
      printf("Youenteredzero."); return
    0;
}
```

Output1

```
Enteranumber:12.3
12.30ispositive.
```

Output2

```
Enteranumber:0
Youenteredzero.
```

<u>CProgramtoCheckerWhetheraCharacterisanAlphabetornot</u>

CProgrammingExamplesandSourceCode

Thisprogramtakesacharacterfromuserandchecks whetherthatcharacterisanalphabet or not.

SourceCodetoCheckCharacterisanalphabetornot

```
/*Cprogrammingcodetocheckwhetheracharacterisalphabetor not.*/

#include<stdio.h>
int main()
{
    char c;
    printf("Enteracharacter:");
    scanf("%c",&c);
    if((c>='a'&&c<='z')||(c>='A'&&c<='Z')) printf("%c is
       an alphabet.",c);
    else
       printf("%cisnotanalphabet.",c);
    return 0;
}
```

Output1

```
Enteracharacter:*
*isnotan alphabet
```

Output2

```
Enteracharacter:K K
is an alphabet
```

When acharacteris stored in avariable,ASCIIvalueofthat characteris stored instead of that characteritself. For example: If'a' is stored in avariable, ASCIIvalueof'a' which is 97 is stored. If you see the ASCII table, the lowercase alphabets are from 97 to 122 and uppercaseletterare from 65 to 90. Ifthe ASCIIvalueof number stored is between any of these two intervals then, that character will be an alphabet. In this program, instead of number 97, 122, 65 and 90; we have used 'a', 'z', 'A' and 'Z' respectively which isbasically the same thing.

Thisprogramalsocanbe performedusingstandardlibraryfunctionisalpha().

CProgramtoFindSumofNaturalNumbers

Positive integers 1, 2, 3,4... areknown as natural numbers. This program takes a positive integerfromuser(supposeuserentered*n*)then,thisprogramdisplaysthevalueof

1+2+3+... n.

SourceCodetoCalculateSumofNaturalNumbers

```
/*Thisprogramissolvedusingwhileloop.*/

#include<stdio.h>
int main()
{
    int n, count, sum=0;
    printf("Enteraninteger:");
    scanf("%d",&n);
    count=1;
    while(count<=n)        /*whileloopterminatesifcount>n*/
    {
        sum+=count;        /*sum=sum+count*/
        ++count;
    }
    printf("Sum=%d",sum);
    return 0;
}
```

SourceCodetoCalculateSumUsingforLoop

```
/*Thisprogramissolveusingforloop. */

#include<stdio.h>
int main()
{
    int n, count, sum=0;
    printf("Enteraninteger:");
    scanf("%d",&n);
    for(count=1;count<=n;++count)/*forloopterminatesifcount>n*/
    {
        sum+=count;                /*sum=sum+count*/
    }
    printf("Sum=%d",sum);
    return 0;
}
```

Output

```
Enteraninteger:100 Sum
= 5050
```

Both program above does exactly the same thing. Initially, the value of *count* is set to 1. Both program have test condition to perform looping iteration until condition `count<=n` becomes false and in each iteration `++count`is executed.

Thisprogramworksperfectlyforpositivenumberbut,ifuserentersnegativenumberor

0, `Sum = 0`is displayed but, it is better is display the error message in this case. The above program can be made user-friendly by adding <u>if else statement</u> as:

```
/* This program displays error message when user enters negative number
or 0 and displays the sum of natural numbers if user enters positive
number. */

#include<stdio.h>
int main()
{
    int n, count, sum=0;
    printf("Enteraninteger:");
    scanf("%d",&n);
    if(n<=0)
        printf("Error!!!!");
    else
    {
       for(count=1;count<=n;++count)/*forloopterminatesifcount>n
*/
       {
          sum+=count;                    /*sum=sum+count*/
       }
       printf("Sum=%d",sum);
    }
    return0;
}
```

Thisprogramalsocan be solvedusing <u>recursivefunction</u>.

<u>CProgramtoFindFactorialofa Number</u>

Foranypositivenumber*n*,itsfactorialisgivenby:

```
factorial=1*2*3*4. ....n
```

Ifanumberis negative,factorial doesnot existand factorialof 0is 1.

This program takes an integer from a user. If user enters negative integer, this program will display error message and if user enters non-negative integer, this program will display the factorial of that number.

<u>SourceCodetoFindFactorialofaNumber</u>

```
/*Cprogramtodisplayfactorialofanintegerifuserentersnon- negative
integer. */

#include<stdio.h>
int main()
{
    intn,count;
```

```
    unsignedlonglongintfactorial=1;
    printf("Enter an integer: ");
    scanf("%d",&n);
    if(n<0)
        printf("Error!!!Factorialofnegativenumberdoesn'texist."); else
    {
       for(count=1;count<=n;++count)        /*forloopterminatesif
count>n */
       {
          factorial*=count;                 /*factorial=factorial*count
*/
       }
    printf("Factorial=%lu",factorial);
    }
    return0;
}
```

Output1

```
Enteraninteger:-5
Error!!!Factorialofnegativenumberdoesn'texist.
```

Output2

```
Enteraninteger:10 Factorial
= 3628800
```

Herethetypeoffactorial variableisdeclared as: `unsignedlonglong`. Itisbecause,the factorial is always positive, so unsigned keyword is used and the factorial of a numbercanbeprettylarge.For example:thefactorialof 10is3628800thus,`long longkeyword`is used.

CprogramtoGenerateMultiplicationTable

Thisprogramasksusertoenteranintegerandthisprogramwillgeneratethe multiplication table upto 10.

SourceCodetoGenerateMultiplicationTable

```
/*Cprogramtofindmultiplicationtableupto10.*/ #include
<stdio.h>
intmain()
{
    intn,i;
    printf("Enteranintegertofindmultiplicationtable:");
    scanf("%d",&n);
```

```
    for(i=1;i<=10;++i)
    {
        printf("%d*%d=%d\n",n,i,n*i);
    }
    return0;
}
```

Output

```
Enteranintegertofindmultiplicationtable:9
9*1=9
9*2=18
9*3=27
9*4=36
9*5=45
9*6=54
9*7=63
9*8=72
9*9=81
9*10=90
```

This program generates multiplication table of an integer up 10. But, this program can be made more user friendly by giving option to user to enter number up to which, he/she want to generate multiplication table. The source code below will perform this task.

```
#include<stdio.h>
int main()
{
    intn,range,i;
    printf("Enteranintegertofindmultiplicationtable:"); scanf("%d",&n);
    printf("Enterrangeofmultiplicationtable:");
    scanf("%d",&range);
    for(i=1;i<=range;++i)
    {
        printf("%d*%d=%d\n",n,i,n*i);
    }
    return0;
}
```

Output

```
Enteranintegertofindmultiplicationtable:6 Enter
range of multiplication table: 4
6*1=6
```

```
6*2=12
6*3=18
6*4=24
```

CProgramtoDisplayFibonacci Series

SourcecodetodisplayFibonacciseriesuptonterms

```
/*DisplayingFibonaccisequenceuptonth termwheren is entered by user. */
#include<stdio.h>
int main()
{
  intcount,n,t1=0,t2=1,display=0;
  printf("Enter number of terms: ");
  scanf("%d",&n);
  printf("FibonacciSeries:%d+%d+",t1,t2);/*Displayingfirsttwo terms */
  count=2;     /*count=2becausefirsttwotermsarealreadydisplayed.
*/
  while(count<n)
  {
      display=t1+t2;
      t1=t2;
      t2=display;
      ++count;
      printf("%d+",display);
  }
  return0;
}
```

Output

```
Enternumberofterms: 10
FibonacciSeries:0+1+1+2+3+5+8+13+21+34+
```

Suppose, instead of number of terms, you want to display the Fibonacci series util the term is less than certain number entered by user. Then, this can be done using sourcecode below:

```
/*DisplayingFibonacciseriesuptocertainnumberenteredbyuser.*/

#include<stdio.h>
int main()
{
  intt1=0,t2=1,display=0,num;
  printf("Enter an integer: ");
  scanf("%d",&num);
  printf("FibonacciSeries:%d+%d+",t1,t2);/*Displayingfirsttwo terms */
  display=t1+t2;
  while(display<num)
  {
```

```
      printf("%d+",display);
      t1=t2;
      t2=display;
      display=t1+t2;
  }
  return0;
}
```

Output

```
Enteraninteger:200
FibonacciSeries:0+1+1+2+3+5+8+13+21+34+55+89+144+
```

CProgramtoFindHCFoftwoNumbers

The greatest common divisor (GCD) of two integers is a largest positive integer(non-zero) that divides the numbers without a remainder.Visit this page to learn more about GCD.

There are many ways to find the GCD of two numbers is C programming. All codes below will take two integers from user and displays the H.C.F of those two integers.

1.SourceCodetoFindHCForGCD

```
/*CProgramtoFindHighestCommonFactor.*/ #include
<stdio.h>
intmain()
{
    int num1, num2, i, hcf;
    printf("Enter two integers: ");
    scanf("%d %d", &num1, &num2);
    for(i=1;i<=num1||i<=num2;++i)
    {
        if(num1%i==0 && num2%i==0)    /* Checking whether i is a factor
of both number */
            hcf=i;
    }
    printf("H.C.Fof%dand%dis%d",num1,num2,hcf); return 0;
}
```

In this program, two integers are taken from user and stored in variable *num1* and *num2*. Then *i* is initialized to 1 and for loop is executed until *i* becomes equal to smallest of two numbers. In each looping iteration, it is checked whether *i* is factor of both numbers or not.If*i*isfactorofbothnumbers,itisstoredtohcf.Whenforloopiscompleted,the H.C.Fof thosetwo numbers will be stored in variable*hcf*.

CProgrammingExamplesandSourceCode

2.SourceCodetoFindHCForGCD

```
#include<stdio.h>
int main()
{
  int num1, num2, min,i;
  printf("Entertwointegers:");
  scanf("%d %d", &num1, &num2);
  min=(num1>num2)?num2:num1;/*minimumvalueisstoredinvariablemin
*/
  for(i=min;i>=1;--i)
  {
      if(num1%i==0&&num2%i==0)
      {
          printf("HCFof%dand%dis%d",num1,num2,i); break;
      }
  }
  return0;
}
```

This program is little optimized than the program above to find H.C.F. In this program, smallest of two integers entered by user is stored in variable *min*. Then *i* is initialized to *min* and for loop is executed. In each looping iteration, whether *i* is factor of these two numbers is checked. If *i* is a factor of these two numbers then, i will be the highest common divisor and loop is terminated using break statement.

3.SourceCodetoFindHCForGCD

```
#include<stdio.h>in
t main()
{
    intnum1,num2;
    printf("Entertwointegers:");
    scanf("%d %d",&num1,&num2);
    printf("HCFof%dand%dis",num1,num2);
    while(num1!=num2)
    {
        if(num1>num2)
            num1-=num2;
        else
            num2-=num1;
    }
    printf("%d",num1);
    return 0;
}
```

This is the best way to find HCF of two numbers. In this method, smaller number is subtracted from larger number and that number is stored in place of larger number. This process is continued until, two numbers become equal which will be HCF.

Allsourcecodesabovedisplaysthesame output.

CProgrammingExamplesandSourceCode

Output

```
Entertwointegers:14 35
HCFof14and35is7
```

CProgramtoFindLCMoftwonumbersenteredbyuser

LCM of two integers a and b is the lowest positive integer this is perfectly divisible by both a and b. Visit this page to learn more about LCM.

SourceCodetoComputeLCM

```
/*CprogramtofindLCMoftwopositiveintegersenteredbyuser */

#include<stdio.h>
int main()
{
  intnum1,num2, max;
  printf("Entertwopositiveintegers:");
  scanf("%d %d", &num1, &num2);
  max=(num1>num2) ? num1 : num2; /* maximum value is stored in variable
max */
  while(1)                    /*Alwaystrue. */
  {
      if(max%num1==0&&max%num2==0)
      {
          printf("LCMof%dand%dis%d",num1,num2,max); break;
                          /* while loop terminates. */
      }
      ++max;
  }
  return0;
}
```

In this program, user is asked to enter two positive integers which will be stored in variable *num1* and *num2* respectively and largest of two integers is assigned to variable *max*. Then, while loop is executed and in each iteration it is checked whether *max* is perfectlydivisiblebytwonumbersenteredbyuserornot.If *max*isnotperfeclydivisible, *max* is increased by 1 and this process goes not until max is perfectly divisible by both numbers. The test condition of while loop in above program is always true so, the loop is terminated using break statement.

CProgrammingExamplesandSourceCode

TheLCM of twonumbers also can befound using following formula:

```
LCM=(num1*num2)/GCD
```

Visitthis pageto learndifferentmethods forfindingGCD oftwo numbers.

```
#include<stdio.h>
int main()
{
    intn1,n2,temp1,temp2;
    printf("Entertwopositiveintegers:");
    scanf("%d %d",&n1,&n2);
    temp1=n1;
    temp2=n2;
    while(temp1!=temp2)
    {
        if(temp1>temp2)
            temp1-=temp2;
        else
            temp2-=temp1;
    }
    printf("LCM of two numbers %d and %d is %d", n1, n2,
(n1*n2)/temp1);
    return0;
}
```

Theoutputofthesetwoprogramsis same.

Output

```
Entertwopositivenumbers:15 9
LCMoftwonumbers15and9is45
```

CProgramtoCountNumberofDigitsofan Integer

This program takes an integer from user and calculates the number of digits in that integer. For example: If user enters 2319, the output of program will be 4 because it contains 4 digits.

CProgramtoFindNumberofDigitsina Number

```
#include<stdio.h>
int main()
{
  intn,count=0;
  printf("Enteraninteger:");
  scanf("%d", &n);
  while(n!=0)
```

```
  {
      n/=10;              /*n=n/10 */
      ++count;
  }
  printf("Numberofdigits:%d",count);
}
```

Output

```
Enteraninteger:34523
Number of digits: 5
```

This program takes an integer from user and stores that number in variable *n*. Suppose, user entered 34523. Then, while loop is executed because `n!=0`will be true in first iteration. The codes inside while loop will be executed. After first iteration, value of n will be 3452 and *count* will be 1. Similarly, in second iteration *n* will be equal to 345 and *count* will beequal to 2. This process goes on and after fourth iteration, n will beequal to 3 and *count* will be equal to 4. Then, in next iteration n will be equal to 0 and *count* will be equal to 5 and program will be terminated as `n!=0`becomes false.

CProgramtoReverseaNumber

Thisprogramtakesan integernumberfromuser andreversesthat number.

CProgramtoReverseanInteger

```
#include<stdio.h>
int main()
{
  int n, reverse=0, rem;
  printf("Enteraninteger:");
  scanf("%d", &n);
  while(n!=0)
  {
     rem=n%10;
     reverse=reverse*10+rem;
    n/=10;
  }
  printf("ReversedNumber=%d",reverse);
  return 0;
}
```

Output

```
Enteraninteger:2345
ReversedNumber=5432
```

CprogramtoCalculatethePowerofaNumber

This program below takes two integers from user, a base number and an exponent. For example: In case of 2^3, 2 is the base number and 3 is the exponent of that number. And this program calculates the value of $base^{exponent}$.

SourceCodetoCalculatePower

```
/*Cprogramtocalculatethepowerofaninteger*/ #include
<stdio.h>
intmain()
{
  intbase,exp;
  longlongintvalue=1;
  printf("Enterbasenumberandexponentrespectively:");
  scanf("%d%d", &base, &exp);
  while(exp!=0)
  {
      value*=base;/*value=value*base;*/
      --exp;
  }
  printf("Answer=%d",value);
}
```

Output

```
Enterbasenumberandexponentrespectively:3 4
Answer=81
```

This program takes base number and exponent from user and stores in variable *base* and *exp*respectively. Let us suppose, user entered 3 and 4 respectively. Then, `while`loop is executed withtestcondition `exp!=0`.Thistestconditionwillbe trueand*value*becomes3 after first iteration and valueof exp will be decreased to 3. This process goes on and after fourth iteration, *value* will be equal to 81(3*3*3*3) and *exp* will be equal to 0 and `while` loop will be terminated. Here, we have declared variable value of type `long long int` because power of a number can be very large.

This program can only calculate the power if base power and exponent are integers. Ifyou need to the calculate power of a floating point number then, you can use pow()function.

CProgramtoCheckWhetheraNumberisPalindromeorNot

This program takes an integer from user and that integer is reversed. If the reversed integeris equal to theinteger entered by userthen, that numberis apalindromeifnot that number is not a palindrome.

CProgramtoCheckPalindromeNumber

```
/*Cprogramtocheckwhetheranumberispalindromeornot */

#include<stdio.h>
int main()
{
  int n, reverse=0, rem,temp;
  printf("Enteraninteger:");
  scanf("%d", &n);
  temp=n;
  while(temp!=0)
  {
     rem=temp%10;
     reverse=reverse*10+rem;
     temp/=10;
  }
/*Checkingifnumberenteredbyuserandit'sreversenumberisequal.
*/
  if(reverse==n)
      printf("%disapalindrome.",n); else
      printf("%disnotapalindrome.",n); return
  0;
}
```

Output

```
Enteraninteger:12321 12321
is a palindrome.
```

CProgramtoCheckWhetheraNumberisPrimeorNot

A positive integer which is only divisible by 1 and iself is known as prime number. For example: 13 is a prime number because it is only divisible by 1 and 13 but, 15 is not prime number because it is divisible by 1, 3, 5 and 15.

CprogramtoCheckPrime Number

```
/*Cprogramtocheckwhetheranumberisprimeornot.*/

#include<stdio.h>
int main()
{
  intn,i,flag=0;
  printf("Enterapositiveinteger:");
  scanf("%d",&n);
  for(i=2;i<=n/2;++i)
  {
      if(n%i==0)
      {
          flag=1;
          break;
      }
  }
  if(flag==0)
      printf("%disaprimenumber.",n); else
      printf("%disnotaprimenumber.",n); return 0;
}
```

Output

```
Enterapositiveinteger:29
29isaprimenumber.
```

Thisprogramtakesapositiveintegerfromuserandstoresitinvariable *n*.Then, `for`loop is executed which checks whether the number entered by user is perfectly divisible by *i*or not starting with initial value of *i* equals to 2 and increasing the value of *i* in each iteration. If the number entered by user is perfectly divisible by *i*then, *flag* is set to 1 and that number will not be a prime number but, if the number is not perfectly divisible by *i* untiltest condition `i<=n/2`is true means, it is only divisible by 1 and that number itself and that number is a prime number.

Visit this page to learn, how you can display all prime numbers between two intervalsentered by user.

CProgrammingExamplesandSourceCode

CProgramtoDisplayPrimeNumbersBetweenTwo Intervals

This program asks user to enter two integers and this program will display all prime numbers between these intervals. If you don't know how to check whether a number is prime or not then, this program may seem little bit complex. You can visit this page to learn about prime numbers and how to check whether a number is prime or not in Cprogramming.

SourceCodetoDisplayPrimeNumbersBetweentwoIntervals

```
/* C program to display all prime numbers between Two interval entered
by user. */
#include<stdio.h>
int main()
{
  intn1,n2,i,j,flag;
  printf("Entertwonumbers(intevals):");
  scanf("%d %d", &n1, &n2);
  printf("Primenumbersbetween%dand%dare:",n1,n2);
  for(i=n1+1; i<n2; ++i)
  {
      flag=0;
      for(j=2;j<=i/2;++j)
      {
        if(i%j==0)
        {
          flag=1;
          break;
        }
      }
      if(flag==0)
        printf("%d",i);
  }
  return0;
}
```

Output

```
Entertwonumbers(intervals):20 50
Primenumbersbetween20and50are:23293137414347
```

In this program, it is assumed that, the user always enters smaller number first. This programwillnot performthetaskintendedifuserenterslargernumberfirst.Youcanadd the code to swap two numbers entered by user if user enters larger number first to make this program work properly.

Visit this page to learn, how you can display all prime numbers between two intervals bymaking user-defined function.

CProgrammingExamplesandSourceCode

CprogramtoCheckArmstrong Number

A positive integer is called an Armstrong number if the sum of cubes of individual digitis equal to that number itself. For example:

```
153=1*1*1+5*5*5+3*3*3//153isanArmstrongnumber.
12isnotequalto1*1*1+2*2*2//12isnotanArmstrongnumber.
```

SourceCodetoCheckArmstrongNumber

```
/* C program to check whether a number entered by user is Armstrong or
not. */

#include<stdio.h>
int main()
{
  intn,n1,rem,num=0;
  printf("Enterapositiveinteger:");
  scanf("%d", &n);
  n1=n;
  while(n1!=0)
  {
      rem=n1%10;
      num+=rem*rem*rem;
      n1/=10;
  }
  if(num==n)
    printf("%disanArmstrongnumber.",n); else
    printf("%disnotanArmstrongnumber.",n);
}
```

Output

```
Enterapositiveinteger:371
371isanArmstrongnumber.
```

CProgramtoDisplayArmstrongNumberBetweenTwoIntervals

This program asks user to enter two integers and this program will display all Armstrong numbers between these intervals. If you don't know how to check whether a number is Armstrong or not in programming then, this program may seem little bit complex. Visit this page to learn about Armstrong number and how to check it in C programming.

CprogramtoDisplayArmstrongNumberBetweenIntervals

```
/*SourceCodetodisplayArmstrongnumberbetweentwointervals    entered    by
user. */

#include<stdio.h>
```

```
intmain()
{
  int n1, n2, i, temp, num, rem;
  printf("Entertwonumbers(intervals):");
  scanf("%d %d", &n1, &n2);
  printf("Armstrongnumbersbetween%dan%dare:",n1,n2); for(i=n1+1; i<n2;
  ++i)
  {
      temp=i;
      num=0;
      while(temp!=0)
      {
          rem=(temp%10);
          num+=rem*rem*rem;
          temp/=10;
      }
      if(i==num)
      {
          printf("%d",i);
      }
  }
  return0;
}
```

Output

```
Entertwonumbers(intervals):100
400
Armstrongnumbersbetween100and400are:153370371
```

In this program, it is assumed that, the user always enters smaller number first. This programwillnot performthetaskintendedifuser enterslargernumberfirst.Youcanadd the code to swap two numbers entered by user if user enters larger number first to make this program work properly.

CprogramtoDisplayFactorsofa Number

Thisprogramtakesapositiveintegerfromanuseranddisplaysallthefactorsofthat number.

SourceCodetoDisplayFactorsofaNumber

```
/*Ctofindanddisplayallthefactorsofanumberenteredbyan user.. */

#include<stdio.h>
int main()
{
  int n,i;
```

```
  printf("Enterapositiveinteger:"); scanf("%d",&n);
  printf("Factorsof%dare:",n); for(i=1;i<=n;++i)
  {
      if(n%i==0)
         printf("%d",i);
  }
  return0;
}
```

Output

```
Enterapositiveinteger:60
Factorsof60are:1234561215203060
```

In this program, an integer entered by user is stored in variable *n*. Then, for loop is executed with initial condition `i=1`and checked whether *n* is perfectly divisible by *i* or not. If *n* is perfectly divisible by *i* then, *i* will be the factor of *n*. In each iteration, thevalue of *i* is updated(increased by 1). This process goes not until test condition `i<=n` becomes false,i.e., this program checks whether number entered by user *n* is perfectly divisible by all numbers from 1 to *n* and all displays factors of that number.

CprogramtoPrintPyramidsandTrianglesinCprogrammingusingLoops

CprogramtoMakeaSimpleCalculatortoAdd,Subtract,MultiplyorDivideUsingswitch...case

This program takes an arithmetic operator (+, -, *, /) and two operands from an user and performs the operation on those two operands depending upon the operator entered by user.

SourceCodetoMakeSimpleCalculatorinCprogramming

```
/*Sourcecodetocreateasimplecalculatorforaddition,subtraction,
multiplicationanddivisionusingswitch...casestatementinC
```

CProgrammingExamplesandSourceCode

```
programming.*/

#include<stdio.h>
int main()
{
    char o;
    floatnum1,num2;
    printf("Enteroperatoreither+or-or*ordivide:"); scanf("%c",&o);
    printf("Entertwooperands:");
    scanf("%f%f",&num1,&num2);
    switch(o) {
        case'+':
            printf("%.1f+%.1f=%.1f",num1,num2,num1+num2); break;
        case'-':
            printf("%.1f-%.1f=%.1f",num1,num2,num1-num2); break;
        case'*':
            printf("%.1f*%.1f=%.1f",num1,num2,num1*num2); break;
        case'/':
            printf("%.1f/%.1f=%.1f",num1,num2,num1/num2); break;
        default:
            /*Ifoperatorisotherthan+,-,*or/,errormessageis
shown*/
            printf("Error!operatorisnotcorrect"); break;
    }
    return0;
}
```

Output

```
Enteroperatoreither+or-or*ordivide:- Enter two
operands: 3.4
8.4
3.4-8.4=-5.0
```

This program takes an operator and two operands from user. The operator is stored in variable operator and two operands are stored in *num1* and *num2* respectively. Then, switch...case statement is used for checking the operator entered by user. If user enters + then, statements for `case: '+'`is executed and program is terminated. If user enters - then, statements for `case: '-'`is executed and program is terminated. This program workssimilarlyfor*and/operator.But,iftheoperatordoesn'tmatchesanyofthefour character[+,-,*and/],defaultstatementisexecutedwhichdisplayserrormessage.

CProgrammingFunctions

Inprogramming,afunction isasegment thatgroupscodeto performaspecifictask.

ACprogramhasatleastonefunction`main()`.Without`main()` function,thereis technically no C program.

TypesofCfunctions

Therearetwotypes of functionsin C programming:

- Libraryfunction
- Userdefinedfunction

Libraryfunction

Libraryfunctionsarethe in-builtfunctioninCprogrammingsystem.For example:

`main()`

- TheexecutionofeveryCprogramstartsfromthis`main()` function.

`printf()`

- `prinf()` isusedfordisplayingoutputinC.

`scanf()`

- `scanf()` isusedfortakinginputinC.

Visitthispageto learnmoreaboutlibraryfunctions inCprogramming language.

Userdefined function

C allows programmer to define their own function according to their requirement. These types of functions are known as user-defined functions. Suppose, a programmer wants to findfactorialofanumberandcheckwhetheritisprimeornotinsameprogram.Then,

he/she can create two separate user-defined functions in that program: one for finding factorial and other for checking whether it is prime or not.

Howuser-definedfunctionworksinCProgramming?

```
#include <stdio.h>
voidfunction_name(){
................
................
}
intmain(){
...........
...........
function_name();
...........
...........
}
```

Asmentionedearlier, everyC program begins from `main()`and program starts executing the codes inside `main()` function. When the control of program reaches to `function_name()` inside `main()` function. The control of program jumps to `void function_name()`andexecutes the codes inside it. When all the codes inside that user-defined function are executed, control of the program jumps to the statement just after `function_name()`from where it is called. Analyze the figure below for understanding the concept of function in C programming. Visit this page to learn in detail about user-defined functions.

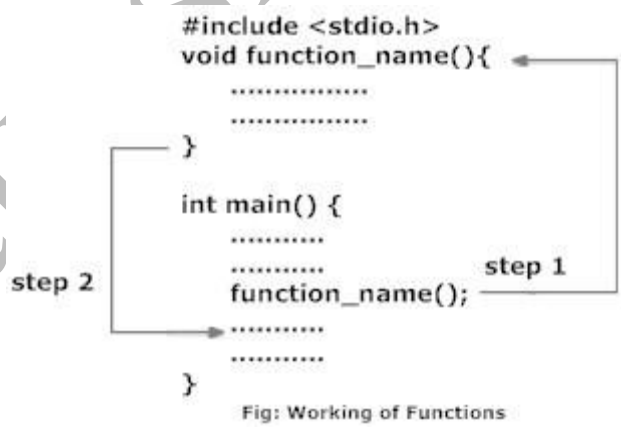


Fig: Working of Functions

Remember,thefunction nameis an identifier andshould be unique.

Advantagesofuserdefinedfunctions

1. User defined functions helps to decompose the large program into small segments which makes programmer easy to understand, maintain and debug.

2. Ifrepeatedcodeoccursinaprogram.Functioncanbeusedtoincludethosecodesand execute when needed by calling that function.
3. Programmerworkingonlargeprojectcandividetheworkloadbymakingdifferent functions.

CProgrammingUser-definedfunctions

Thischapteristhecontinuationtothe functionIntroductionchapter.

Exampleofuser-definedfunction

WriteaCprogramtoaddtwointegers.Makeafunction`add`toaddintegersand display sum in `main()` function.

```
/*Programtodemonstratetheworkingofuserdefinedfunction*/ #include
<stdio.h>
int add(int a, int b);           //functionprototype(declaration)
int main(){
     intnum1,num2,sum;
     printf("Enters two number to add\n");
     scanf("%d %d",&num1,&num2);
     sum=add(num1,num2);         //functioncall
     printf("sum=%d",sum);
     return0;
}
intadd(inta,int b)               //functiondeclarator
{
/*Startoffunctiondefinition.*/ int
     add;
     add=a+b;
     returnadd;                  //returnstatementoffunction
/*Endoffunctiondefinition.*/
}
```

Function prototype(declaration):

Every function in C programming should be declared before they are used. These type of declaration are also called function prototype. Function prototype gives compiler information about function name, type of arguments to be passed and return type.

Syntaxoffunctionprototype

```
return_type        function_name(type(1)        argument(1),. . . ,type(n)
argument(n));
```

Inthe above example,`int add(int a, int b);`is a function prototype which provides following information to the compiler:

1. nameofthefunctionis `add()`
2. returntypeofthefunctionis`int`.
3. twoargumentsoftype`int`arepassedto function.

Functionprototypearenotneededifuser-definitionfunctioniswrittenbefore`main()` function.

Functioncall

Control of the program cannot be transferred to user-defined function unless it is called invoked.

Syntaxoffunctioncall

```
function_name(argument(1),....argument(n));
```

In the above example, function call is made using statement `add(num1,num2);` from `main()`. This make the control of program jump from that statement to functiondefinition and executes the codes inside that function.

Functiondefinition

Functiondefinitioncontainsprogrammingcodestoperformspecific task.

Syntaxoffunctiondefinition

```
return_typefunction_name(type(1)argument(1),..,type(n)argument(n))
{
                //bodyoffunction
}
```

Functiondefinitionhastwomajor components:

1.Functiondeclarator

Functiondeclaratoristhefirstlineoffunctiondefinition.Whenafunctioniscalled, control of the program is transferred to function declarator.

Syntaxoffunctiondeclarator

```
return_typefunction_name(type(1)argument(1),....,type(n)argument(n))
```

Syntaxoffunctiondeclarationanddeclaratorarealmostsameexcept,thereisno semicolon at the end of declarator and function declarator is followed by function body.

Inaboveexample,`intadd(inta,intb)` inline12isafunction declarator.

2.Functionbody

Functiondeclaratoris followedby bodyoffunctioninsidebraces.

Passingargumentstofunctions

In programming, argument(parameter) refers to data this is passed to function(function definition) while calling function.

In above example two variable, num1 and num2 are passed to function during function call and these arguments are accepted by arguments a and b in function definition.

```
#include <stdio.h>
int add(int a, int b);
int main(){
    ............
  sum=add(num1,num2);
    ...........
 }
      int add(int a, int b) {
         ..................
         ................
    }
   Here,
         a=num1
         b=num2
```

Arguments that are passed in function call and arguments that are accepted in function definition should have same data type. For example:

If argument *num1* was of int type and *num2* was of float type then, argument variable *a* should be of type int and b should be of type float,i.e., type of argument during function call and function definition should be same.

Afunctioncanbecalledwithorwithoutan argument.

ReturnStatement

Returnstatementisusedforreturningavaluefrom functiondefinitiontocalling function.

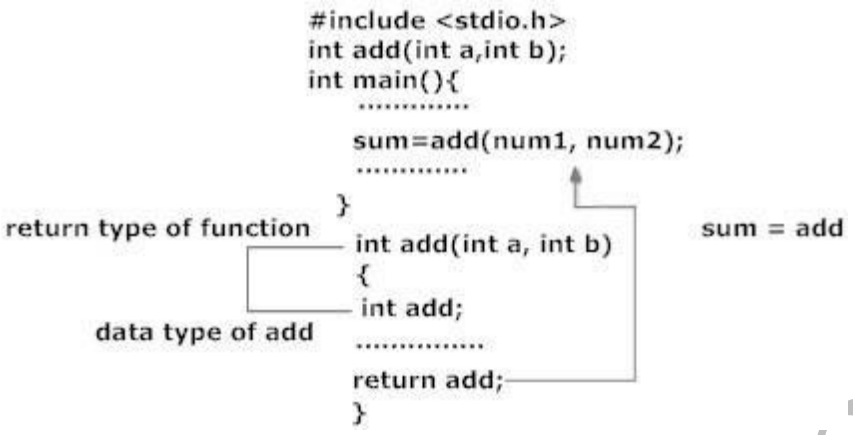
Syntaxofreturnstatement

```
return(expression);
```

Forexample:

```
return a;
return(a+b);
```

In above example, value of variable add in `add()` function is returned and that value is storedinvariable*sum*in`main()` function.Thedatatypeofexpressioninreturnstatement should also match the return type of function.



TypesofUser-definedFunctionsinCProgramming

Forbetterunderstandingofargumentsandreturntypeinfunctions,user-defined functions can be categorised as:

1. Functionwithnoargumentsandnoreturn value
2. Functionwithnoargumentsandreturnvalue
3. Functionwithargumentsbutnoreturnvalue
4. Functionwithargumentsandreturnvalue.

Let's take an example to find whether a number is prime or not using above 4 categories of user defined functions.

Functionwithnoargumentsandnoreturnvalue.

```
/*C program tocheck whether anumberentered by userisprime or not using
function with no arguments and no return value*/
#include<stdio.h>
void prime();
intmain(){
    prime();       //Noargumentispassedtoprime().
    return 0;
}
voidprime(){
/*Thereisnoreturnvaluetocalling functionmain().Hence,return type of
prime() is void */
    intnum,i,flag=0;
    printf("Enterpositiveintegerentertocheck:\n");
    scanf("%d",&num);
    for(i=2;i<=num/2;++i){
        if(num%i==0){
             flag=1;
         }
    }
    if(flag==1)
        printf("%disnotprime",num); else
       printf("%disprime",num);
```

```
    }
```

Function`prime()`is used forasking user ainput, check forwhetherit is primeofnot and display it accordingly. No argument is passed and returned form `prime()`function.

Functionwithnoargumentsbutreturnvalue

```
/*C program tocheck whether anumberentered by userisprime or not using
function with no arguments but having return value */
#include<stdio.h>
int input();
intmain(){
    intnum,i,flag= 0;
    num=input();     /*Noargumentispassedtoinput()*/
    for(i=2; i<=num/2; ++i){
    if(num%i==0){
        flag=1;
        break;
    }
    }
    if(flag== 1)
        printf("%disnotprime",num); else
        printf("%disprime",num);
    return 0;
}
int input(){     /*Integervalueisreturnedfrominput()tocalling function
*/
    intn;
    printf("Enterpositiveintegertocheck:\n");
    scanf("%d",&n);
    returnn;
}
```

Thereisnoargumentpassedto`input()`functionBut,thevalueof*n*isreturnedfrom `input()`to`main()`function.

Functionwithargumentsandnoreturnvalue

```
/*Programtocheckwhetheranumberenteredbyuserisprimeornot using function
with arguments and no return value */
#include<stdio.h>
voidcheck_display(intn);
int main(){
    int num;
    printf("Enterpositiveentertocheck:\n");
    scanf("%d",&num);
    check_display(num);/*Argumentnumispassedtofunction.*/ return
    0;
}
voidcheck_display(intn){
```

```
/* There is no return value to calling function. Hence, return type of
function is void. */
    int i, flag = 0;
for(i=2;i<=n/2;++i){
    if(n%i==0){
        flag=1;
        break;
    }
    }
    if(flag== 1)
        printf("%disnotprime",n);
    else
        printf("%disprime",n);
}
```

Here,`check_display()` functionis used for check whether it is prime or not and display it accordingly. Here, argument is passed to user-defined function but, value is notreturned from it to calling function.

Functionwithargumentandareturnvalue

```
/*Programtocheckwhetheranumberenteredbyuserisprimeornot using function
with argument and return value */
#include<stdio.h>
int check(int n);
int main(){
    intnum,num_check=0;
    printf("Enterpositiveentertocheck:\n");
    scanf("%d",&num);
    num_check=check(num);/*Argumentnumispassedtocheck() function. */
    if(num_check==1)
       printf("%disnotprime",num); else
       printf("%disprime",num);
    return 0;
}
intcheck(intn){
/*Integervalueisreturnedfromfunctioncheck()*/ int i;
    for(i=2;i<=n/2;++i){
    if(n%i==0)
        return1;
}
   return0;
}
```

CProgrammingRecursion

A function that calls itself is known as recursive function and this technique is known as recursion in C programming.

ExampleofrecursioninCprogramming

WriteaCprogramtofindsumoffirstnnaturalnumbersusingrecursion.Note: Positive integers are known as natural number i.e. 1, 2, 3.....n

```
#include<stdio.h>
int sum(int n);
int main(){
    intnum,add;
    printf("Enterapositiveinteger:\n");
    scanf("%d",&num);
    add=sum(num);
    printf("sum=%d",add);
}
intsum(intn){
    if(n==0)
       returnn;
    else
       returnn+sum(n-1);     /*selfcalltofunctionsum() */
}
```

Output

```
Enterapositiveinteger:
5
15
```

In,this simple C program, `sum()`function is invoked from the same function. If *n* is not equal to 0 then, the function calls itself passing argument 1 less than the previous argument it was called with. Suppose, *n* is 5 initially. Then, during next function calls, 4 is passed to function and the value of argument decreases by 1 in each recursive call. When, n becomes equal to 0, the value of n is returned which is the sum numbers from 5 to 1.

Forbettervisualizationofrecursioninthis example:

```
sum(5)
=5+sum(4)
=5+4+sum(3)
=5+4+3+sum(2)
=5+4+3+2+sum(1)
=5+4+3+2+1+sum(0)
=5+4+3+2+1+0
=5+4+3+2+1
=5+4+3+3
=5+4+6
=5+10
=15
```

Everyrecursivefunctionmustbeprovidedwithawaytoendtherecursion.Inthis example when, *n* is equal to 0, there is no recursive call and recursion ends.

AdvantagesandDisadvantagesofRecursion

Recursion is more elegant and requires few variables which make program clean. Recursion can be used to replace complex nesting code by dividing the problem intosame problem of its sub-type.

In other hand, it is hard to think the logic of a recursive function. It is also difficult to debug the code containing recursion.

UNIT-3

CProgrammingStorageClass

Every variable in C programming has two properties: type and storage class. Type refers to the data type of variable whether it is character or integer or floating-point value etc. And storage class determines how long it stays in existence.

Thereare4 typesof storageclass:

1. automatic
2. external
3. static
4. register

Automaticstorageclass

Keywordforautomaticvariable

`auto`

Variables declared inside the function body are automatic by default. These variable are also known as local variables as they are local to the function and doesn't have meaning outside that function

Since,variableinside afunctionisautomaticbydefault,keywordautoare rarely used.

Externalstorageclass

External variable can be accessed by any function. They are also known as global variables. Variables declared outside every function are external variables.

In case of large program, containing more than one file, if the global variable is declared in file 1 and that variable is used in file 2 then, compiler will show error. To solve this problem,keyword`extern`is usedinfile2toindicatethat,thevariablespecifiedisglobal variable and declared in another file.

Exampletodemonstrateworkingofexternalvariable

```
#includevoidC
heck(); int
a=5;
/*aisglobalvariablebecauseitisoutsideeveryfunction*/ int main(){
    a+=4;
    Check();
    return0;
```

```
}

voidCheck(){
   ++a;
/*----- Variable a is not declared in this function but, works in any
function as they are global variable--------- */
   printf("a=%d\n",a);
}
```

Output

```
a=10
```

RegisterStorageClass

Keywordtodeclareregistervariable

```
register
```

Exampleofregistervariable

```
registerinta;
```

Register variables are similar to automatic variable and exists inside that particular function only.

If the compiler encounters register variable, it tries to store variable in microprocessor's register rather than memory. Value stored in register are much faster than that ofmemory.

In case of larger program, variables that are used in loops and function parameters are declared register variables.

Since,therearelimitednumberofregisterinprocessorand ifitcouldn'tstorethevariable in register, it will automatically store it in memory.

StaticStorageClass

Thevalueofstaticvariablepersistsuntiltheendoftheprogram.Avariablecanbe declared static using keyword: `static`. For example:

```
staticint i;
```

Here,*i*isastatic variable.

Exampletodemonstratethestaticvariable

```
#include<stdio.h>
void Check();
intmain(){
   Check();
   Check();
```

```
   Check();
}
voidCheck(){
    static int c=0;
    printf("%d\t",c);
    c+=5;
}
```

Output

```
0       5      10
```

During first function call, it will display 0. Then, during second function call, variable *c* will not be initialized to 0 again, as it is static variable. So, 5 is displayed in second function call and 10 in third call.

Ifvariablechadbeenautomaticvariable, theoutput would have been:

```
0      0      0
```

CProgrammingFunctionExample

Thispagecontainsexamplesandsource codeon howtowork withuser-definedfunction. To understand all the program on this page, you should have knowledge of following function topics:

1. User-DefinedFunction
2. User-DefinedFunctionTypes
3. Storageclass(Specially,localvariables)
4. Recursion

CFunctionExamples

CProgrammingExamplesandSourceCode

CProgramtoDisplayPrimeNumbersBetweenIntervalsbyMakingFunction

CProgrammingExamplesandSourceCode

This program takes two positive integers from user and displays all prime numbers between these two intervals. To perform this task, user-defined function is created which will check whether a number is prime or not.

PrimeNumbersBetweentwoIntervalsbyMakingUser-definedFunction

```
#include<stdio.h>
intcheck_prime(intnum);
int main(){
   intn1,n2,i,flag;
   printf("Entertwonumbers(intervals):");
   scanf("%d %d",&n1, &n2);
   printf("Primenumbersbetween%dand%dare:",n1,n2); for(i=n1+1;i<n2;++i)
   {
      flag=check_prime(i);
      if(flag==0)
         printf("%d",i);
   }
   return0;
}
intcheck_prime(intnum)/*User-definedfunctiontocheckprimenumber*/
{
   int j,flag=0;
   for(j=2;j<=num/2;++j){
        if(num%j==0){
            flag=1;
            break;
        }
   }
   returnflag;
}
```

Output

```
Entertwonumbers(intervals):10 30
Primenumbersbetween10and30are:111317192329
```

In this program, all numbers between two intervals is passed to function **`int check_prime(intnum)`** using for loop. This function checks whether a number is prime or not. If the number is prime it returns 1 if not it return 0.

CProgramtoCheckPrimeandArmstrongNumberbyMaking Function

CProgrammingExamplesandSourceCode

In this program, user is asked to enter a positive integer and a character either 'p' or 'a'. If user enters *p* then, this program checks whether that number is prime or not and if user enters *a* then, this program checks whether that number is an Armstrong number or not. To perform this task, two user-defined functions are defined to check prime number and Armstrong number.

```
/* C program to check either prime number or Armstrong number depending
upon the data entered by user. */
#include<stdio.h>
int prime(int n);
intarmstrong(intn); int
main()
{
    char c;
    intn,temp=0;
    printf("Eneterapositiveinteger:");
    scanf("%d",&n);
    printf("EnterPtocheckprimeand         AtocheckArmstrongnumber:
");
    c=getche();
    if(c=='p'||c=='P')
    {
        temp=prime(n);
        if(temp==1)
           printf("\n%disaprimenumber.",n); else
           printf("\n%disnotaprimenumber.",n);
    }
    if(c=='a'||c=='A')
    {
        temp=armstrong(n);
        if(temp==1)
           printf("\n%disanArmstrongnumber.",n); else
           printf("\n%disnotanArmstrongnumber.",n);
    }
    return0;
}
intprime(intn)
{
    int i, flag=1;
    for(i=2;i<=n/2;++i)
    {
       if(n%i==0)
       {
          flag=0;
          break;
       }
    }
    returnflag;
}
```

```
intarmstrong(intn)
{
    intnum=0,temp,flag=0; temp=n;
    while(n!=0)
    {
        num+=(n%10)*(n%10)*(n%10);
        n/=10;
    }
    if(num==temp)
       flag=1;
    returnflag;
}
```

Output

```
Eneterapositiveinteger:371
EnterPtocheckprimeandAtocheckArmstrongnumber:p
371isnotaprimenumber.
```

CprogramtoCheckWhetheraNumbercanbeExpressasSumofTwoPrime Numbers

This program takes a positive integer from user and checks whether that number can be expressed as the sum of two prime numbers. If that number can be expressed as sum of two prime numbers then, that number is expressed as sum of two prime numbers in output. To perform this task, a user-defined function is created to check prime number.

```
#include<stdio.h>
int prime(int n);
int main()
{
    intn,i,flag=0;
    printf("Enterapositiveinteger:");
    scanf("%d",&n);
    for(i=2;i<=n/2;++i)
    {
        if(prime(i)!=0)
        {
            if(prime(n-i)!=0)
            {
                printf("%d=%d+%d\n",n,i,n-i);
                flag=1;
            }

        }
    }
    if(flag==0)
      printf("%dcan'tbeexpressedassumoftwoprimenumbers.",n); return 0;
}
```

```
intprime(intn)        /*Functiontocheckprimenumber*/
{
    int i, flag=1;
    for(i=2;i<=n/2;++i)
       if(n%i==0)
          flag=0;
    return flag;
}
```

Output

```
Enterapositiveinteger:34
34=3+31
34=5+29
34=11+23
34=17+17
```

CprogramtoFindSumofNaturalNumbersusing Recursion.

Inthisprogram,userisaskedtoenterapositiveintegerandsumofnaturalnumbersupto that integer is displayed by this program. Suppose, user enters 5 then,

```
Sumwillbeequalto1+2+3+4+5= 15
```

Insteadofusingloopstofindsumofnaturalnumbers,`recursion` isusedinthis program.

SourceCodetoCalculatedSumusingRecursion

```
#include<stdio.h>
int add(int n);
int main()
{
    intn;
    printf("Enteranpositiveinteger:");
    scanf("%d",&n);
    printf("Sum=%d",add(n)); return
    0;
}
intadd(intn)
{
    if(n!=0)
     returnn+add(n-1);/*recursivecall*/
}
```

Output

```
Enteranpositiveinteger:20 Sum =
210
```

CProgrammingExamplesandSourceCode

CprogramtoCalculateFactorialofaNumberUsing Recursion

This program takes a positive integer from user and calculates the factorial of that number. Instead of loops to calculate factorial, this program uses recursive function to calculate the factorial of a number.

SourceCodetoCalculateFactorialUsingRecursion

```
/*Sourcecodetofindfactorialofanumber. */

#include<stdio.h>
intfactorial(intn); int
main()
{
    intn;
    printf("Enteranpositiveinteger:"); scanf("%d",&n);
    printf("Factorialof%d=%ld",n,factorial(n)); return
    0;
}
intfactorial(intn)
{
    if(n!=1)
     returnn*factorial(n-1);
}
```

Output

```
Enteranpositiveinteger:6 Factorial
of 6 = 720
```

CProgramtoFindH.C.FUsingRecursion

CProgramtoFindG.C.DUsingRecursion

This program takes two positive integers from user and calculates GCD using recursion. Visit this page to learn, how you can calculate GCD using loops.

SourcecodetoCalculateH.C.Fusingrecursion

```
/*ExampletocalculateGCDorHCFusingrecursivefunction.*/

#include<stdio.h>
inthcf(intn1,intn2); int
main()
{
   intn1,n2;
   printf("Entertwopositiveintegers:");
   scanf("%d%d",&n1,&n2);
```

```
   printf("H.C.Fof%dand%d=%d",n1,n2,hcf(n1,n2)); return 0;
}
inthcf(intn1,intn2)
{
    if(n2!=0)
       returnhcf(n2,n1%n2); else
       returnn1;
}
```

Output

```
Entertwopositiveintegers:366 60
H.C.Fof366and60=6
```

CprogramtoReverseaSentenceUsing Recursion

This program takes a sentence from user and reverses that sentence using recursion. This program does not use string to reverse the sentence or store the sentence.

Sourcecodetoreverseasentenceusingrecursion.

```
/*Exampletoreverseasentenceenteredbyuserwithoutusingstrings.
*/

#include<stdio.h>
void Reverse();
int main()
{
    printf("Enterasentence:");
    Reverse();
    return0;
}
voidReverse()
{
    char c;
    scanf("%c",&c);
    if(c!='\n')
    {
       Reverse();
       printf("%c",c);
    }
}
```

Output

```
Enterasentence:margorpemosewa
awesome program
```

Thisprogramprints"Enterasentence:"then,`Reverse()` functioniscalled.This

function stores the first letter entered by user and stores in variable c. If that variable is other than '\n' [enter character] then, again `Reverse()` function is called. Don't assume this`Reverse()` function and the `Reverse()` function before is same although they both have same name. Also, the variables are also different, i.e., c variable in both functions are also different. Then, the second character is stored in variable c of second Reverse function. This process goes on until user enters '\n'. When, user enters '\n', the last function `Reverse()` function returns to second last `Reverse()` function and prints the last character. Second last `Reverse()` function returns to the third last `Reverse()` function and prints second last character. This process goes on and the final output willbe the reversed sentence.

CprogramtoCalculatethePowerofaNumberUsingRecursion

This program takes two integers from user (base number and a exponent) and calculates the power. Instead of using loops to calculate power, this program uses recursion to calculate the power of a number.

Sourcecodetocalculatepowerusingrecursion

```
/*SourceCodetocalculatepowerusingrecursivefunction*/

#include<stdio.h>
intpower(intn1,intn2); int
main()
{
    intbase,exp;
    printf("Enterbasenumber:");
    scanf("%d",&base);
    printf("Enterpowernumber(positiveinteger):");
    scanf("%d",&exp);
    printf("%d^%d=%d",base,exp,power(base,exp)); return 0;
}
intpower(intbase,intexp)
{
    if(exp!=1)
        return(base*power(base,exp-1));
}
```

Output

```
Enterbasenumber:3
```

```
Enterpowernumber(positiveinteger):3 3^3 =
27
```

This program can only calculate the power if base power and exponent are integers only. If you need to the calculate power of a floating point number then, you can use pow() library function.

CProgramtoConvertBinaryNumbertoDecimalandDecimalto Binary

This program converts either binary number entered by user to decimal number or decimal number entered by user to binary number in accordance with the character entered by user.

SourceCodetoConvertEitherBinaryNumbertoDecimalorDecimalNumbertoBinary

```
/*Cprogrammingsourcecodetoconverteitherbinarytodecimalor decimal to
binary according to data entered by user. */

#include<stdio.h>
#include <math.h>
intbinary_decimal(intn);
intdecimal_binary(intn);
int main()
{
   int n;
   charc;
   printf("Instructions:\n");
   printf("1.Enteralphabet'd'toconvertbinarytodecimal.\n");
   printf("2.Enteralphabet'b'toconvertdecimaltobinary.\n");
   scanf("%c",&c);
   if(c=='d'||c=='D')
   {
       printf("Enterabinarynumber:");
       scanf("%d", &n);
       printf("%dinbinary=%dindecimal",n,binary_decimal(n));
   }
   if(c=='b'||c=='B')
   {
       printf("Enteradecimalnumber:");
       scanf("%d", &n);
       printf("%dindecimal=%dinbinary",n,decimal_binary(n));
   }
   return0;
}

intdecimal_binary(intn)/*Functiontoconvertdecimaltobinary.*/
{
    intrem,i=1,binary=0;
    while(n!=0)
```

```
    {
        rem=n%2;
        n/=2;
        binary+=rem*i;
        i*=10;
    }
    returnbinary;
}

intbinary_decimal(intn)/*Functiontoconvertbinarytodecimal.*/

{
    intdecimal=0,i=0,rem;
    while (n!=0)
    {
        rem=n%10;
        n/=10;
        decimal+=rem*pow(2,i);
        ++i;
    }
    returndecimal;
}
```

Output

```
Instructions:
1. Enteralphabet'd'toconvertbinarytodecimal.
2. Enteralphabet'b'toconvertdecimaltobinary. d
Enter a binary number: 110111
110111inbinary=55indecimal
```

This program asks user to enter alphabet 'b' to convert decimal number to binary and alphabet 'd' to convert binary number to decimal. In accordance with the character entered, user is asked to enter either binary value to convert to decimal or decimal value to convert to binary.

To perform conversion, two functions are made `decimal_binary()`; to convert decimal to binary and `binary_decimal()`; to convert binary to decimal. Decimal number entered by user is passed to `decimal_binary()` and this function computes the binary value of that number and returns it `main()` function. Similarly, binary number is passed to function `binary_decimal()` and this function computes decimal value of that number and returns it to `main()` function.

<u>CProgramtoConvertOctalNumbertoDecimalandDecimaltoOctal</u>

Thisprogramconvertseitheroctalnumberenteredbyusertodecimalnumberordecimal

numberenteredby userto octalin accordancewith thecharacter enteredby user.

SourceCodetoConvertOctalNumbertoDecimalandViceVersa

```
/*Cprogrammingsourcecodetoconverteitheroctaltodecimalor decimal to octal
according to data entered by user. */

#include<stdio.h>
#include <math.h>
intdecimal_octal(intn);
intoctal_deciaml(intn);
int main()
{
   int n;
   charc;
   printf("Instructions:\n");
   printf("1.Enteralphabet'o'toconvertdecimaltooctal.\n");
   printf("2.Enteralphabet'd'toconvertoctaltodecimal.\n");
   scanf("%c",&c);
   if(c=='d'||c=='D')
   {
       printf("Enteranoctalnumber:"); scanf("%d",
       &n);
       printf("%dinoctal=%dindecimal",n,octal_decimal(n));
   }
   if(c=='o'||c=='O')
   {
       printf("Enteradecimalnumber:"); scanf("%d",
       &n);
       printf("%dindecimal=%dinoctal",n,decimal_octal(n));
   }
   return0;
}

intdecimal_octal(intn)/*Functiontoconvertdecimaltooctal*/
{
    intrem,i=1,octal=0; while
    (n!=0)
    {
        rem=n%8;
        n/=8;
        octal+=rem*i;
        i*=10;
    }
   returnoctal;
}

intoctal_decimal(intn)/*Functiontoconvertoctaltodecimal*/
{
    intdecimal=0,i=0,rem;
    while (n!=0)
    {
        rem=n%10; n/=10;
        decimal+=rem*pow(8,i);
        ++i;
```

```
    }
    returndecimal;
}
```

Output

```
Instructions:
1. Enteralphabet'o'toconvertdecimaltooctal.
2.  Enteralphabet'd'toconvertoctaltodecimal. d
Enter an octal number: 2341
2341inoctal=1249indecimal
```

This program asks user to enter a character and in accordance with that character user is asked to enter either octal number or decimal number. If user chooses to convert octal number to decimal then, that number is passed to function `octal_decimal()`. This function will convert the octal number passed by user to decimal number and returns it to main function. Similarly, if user chooses to convert decimal number to octal then, that number is passed to function `decimal_octal()`. This function will convert decimal number to octal number and returns it to main function.

DisplayingandReadingOctalNumberusing%o

Inthe aboveprogram,itisshownhowyoucanconvertdecimalnumberto octalandoctal to decimal number manually. But, in C programming, there is easy solution for it. You can display the corresponding octal number of a decimal numberdirectly using %o format string. You also can take number from user in octal form using %o. This program will demonstrate the working of %o.

```
/*Cprogramtotakeanddisplaynumberinoctalform*/ #include
<stdio.h>
intmain()
{
  intn;
  printf("Enteradecimalnumber:");
  scanf("%d",&n);
/*%owilldisplaytheintegerincorrespondingoctalform*/
  printf("%d in decimal = %o in octal", n, n);
  printf("\nEnter an octal number: ");
  scanf("%o",&n);       /*Takesnumberinoctalform.*/
  printf("%o in octol = %d in decimal", n, n);
  return0;
}
```

CProgrammingExamplesandSourceCode

CProgramtoConvertBinaryNumbertoOctalandOctalto Binary

This program converts either binary number entered by user to octal number or octal number entered by user to binary number in accordance with the character entered by user.

SourceCodetoConvert BinarytoOctalandViceVersa

```
/* C programming source code to convert either binary to octal or octal
to binary according to data entered by user. */

#include<stdio.h>
#include <math.h>
intbinary_octal(intn);
intoctal_binary(intn);
int main()
{
    int n;
    charc;
    printf("Instructions:\n");
    printf("1.Enteralphabet'o'toconvertbinarytooctal.\n");
    printf("2.Enteralphabet'b'toconvertoctaltobinary.\n");
    scanf("%c",&c);
    if(c=='o'||c=='O')
    {
        printf("Enterabinarynumber:");
        scanf("%d",&n);
        printf("%dinbinary=%dinoctal",n,binary_octal(n));
    }
    if(c=='b'||c=='B')
    {
        printf("Enteraoctalnumber:");
        scanf("%d",&n);
        printf("%dinoctal=%dinbinary",n,octal_binary(n));
    }
    return0;
}
intbinary_octal(intn)/*Functiontoconvertbinarytooctal.*/
{
    intoctal=0,decimal=0,i=0;
    while(n!=0)
    {
        decimal+=(n%10)*pow(2,i);
        ++i;
        n/=10;
    }
```

```
/*Atthispoint,thedecimalvariablecontainscorrespondingdecimal value of
binary number. */

    i=1;
    while(decimal!=0)
    {
        octal+=(decimal%8)*i;
        decimal/=8;
        i*=10;
    }
    returnoctal;
}
intoctal_binary(intn)/*Functiontoconvertoctaltobinary.*/
{
    intdecimal=0,binary=0,i=0; while
    (n!=0)
    {
        decimal+=(n%10)*pow(8,i);
        ++i;
        n/=10;
    }
/*Atthispoint,thedecimalvariable containscorrespondingdecimal value of
that octal number. */
    i=1;
    while(decimal!=0)
    {
        binary+=(decimal%2)*i;
        decimal/=2;
        i*=10;
    }
    returnbinary;
}
```

Output

```
Instructions:
1. Enteralphabet'o'toconvertbinarytooctal.
2. Enteralphabet'b'convertoctaltobinary. o
Enter a binary number: 11011
11011inbinary=33inoctal
```

This program asks user to enter alphabet 'b' to convert octal number to binary or alphabet 'o' to convert binary number to octal. In accordance with the character entered, user is asked to enter either binary value to convert to octal or octal value to convert to binary.

To perform conversion, two functions are made `octal_binary();`to convert octal to binary and binary_octal(); to convert binary to octal. Octal number entered by user is passedtooctal_binary()andthisfunctioncomputesthebinaryvalueofthatoctalnumber number and returns it main() function. Similarly, binary number is passed to function binary_octal()andthisfunctioncomputesoctalvalueofthatnumberandreturnitmain()

function.

CProgrammingArrays

In C programming, one of the frequently arising problem is to handle similar types of data. For example: If the user want to store marks of 100 students. This can be done by creating 100 variable individually but, this process is rather tedious and impracticable. These type of problem can be handled in C programming using arrays.

Anarrayisasequenceofdataitemofhomogeneousvalue(sametype). Arrays

are of two types:

1. One-dimensionalarrays
2. Multidimensionalarrays(willbediscussedinnextchapter)

Declarationofone-dimensional array

```
data_typearray_name[array_size];
```

Forexample:

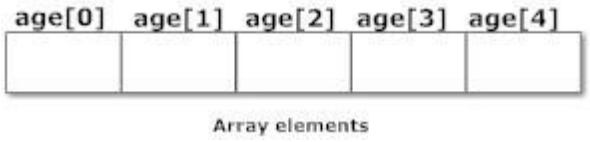
```
intage[5];
```

Here, the name of array is *age*. The size of array is 5,i.e., there are 5 items(elements) of array *age*. All element in an array are of the same type (int, in this case).

Arrayelements

Size of arraydefinesthenumber of elementsinanarray.Eachelementof arraycanbe accessed and used by user according to the need of program. For example:

```
intage[5];
```



Notethat,the firstelement isnumbered 0and soon.

Here,the sizeofarray *age*is 5times thesizeof intbecausethereare5elements.

Suppose, the starting address of `age[0]`is 2120d and the size of int be 4 bytes. Then, the next address (address of `a[1]`) will be 2124d, address of `a[2]`will be 2128d and so on.

Initializationofone-dimensionalarray:

Arrays canbeinitializedatdeclarationtimeinthissourcecodeas:

```
intage[5]={2,4,34,3,4};
```

Itisnot necessaryto definethe sizeofarraysduring initialization.

```
intage[]={2,4,34,3,4};
```

Inthiscase,thecompilerdeterminesthesizeofarraybycalculatingthenumberof elements of an array.

Initialization of one-dimensional array

Accessingarrayelements

InCprogramming,arrayscanbeaccessedandtreatedlikevariablesinC. For

example:

```
scanf("%d",&age[2]);
/*statementtoinsertvalueinthethirdelementofarrayage[]. */

scanf("%d",&age[i]);
/*Statementtoinsertvaluein(i+1)thelementofarrayage[].*/
/* Because, the first element of array is age[0], second is age[1], ith
is age[i-1] and (i+1)th is age[i]. */

printf("%d",age[0]);
/*statementtoprintfirstelementofanarray. */

printf("%d",age[i]);
/*statementtoprint(i+1)thelementofanarray.*/
```

ExampleofarrayinCprogramming

```
/*Cprogramtofindthesummarksofnstudentsusingarrays*/ #include

<stdio.h>
```

```
intmain(){
     int marks[10],i,n,sum=0;
     printf("Enternumberofstudents:");
     scanf("%d",&n);
     for(i=0;i<n;++i){
          printf("Entermarksofstudent%d:",i+1);
          scanf("%d",&marks[i]);
          sum+=marks[i];
     }
     printf("Sum=%d",sum);
return 0;
}
```

Output

```
Enternumberofstudents:3
Entermarksofstudent1:12
Entermarksofstudent2:31 Enter
marks of student3: 2 sum=45
```

ImportantthingtorememberinC arrays

Suppose,youdeclaredthearrayof10students.Forexample: `arr[10]`.Youcanusearray members from `arr[0]` to `arr[9]`. But, what if you want to use element `arr[10]`, `arr[13]`etc. Compiler may not show error using these elements but, may cause fatal error during program execution.

CProgrammingMultidimensionalArrays

Cprogramminglanguageallowsprogrammertocreatearraysof arraysknownas multidimensional arrays. For example:

```
floata[2][6];
```

Here,*a*isan arrayoftwo dimension,whichisanexampleofmultidimensionalarray.

For better understandingof multidimensionalarrays,arrayelementsofabove example can be thinked of as below:

	col 1	col 2	col 3	col 4	col 5	col 6
row 1	a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]
row 2	a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]

Figure: Multidimensional Arrays

InitializationofMultidimensionalArrays

InC,multidimensionalarrayscanbeinitializedindifferentnumberofways.

```
intc[2][3]={{1,3,0},{-1,5,9}};
                 OR
intc[][3]={{1,3,0},{-1,5,9}};
                 OR
intc[2][3]={1,3,0,-1,5,9};
```

InitializationOfthree-dimensionalArray

```
doublecprogram[3][2][4]={
{{-0.1,0.22,0.3,4.3},{2.3,4.7,-0.9,2}},
 {{0.9,3.6,4.5,4},{1.2,2.4,0.22,-1}},
 {{8.2,3.12,34.2,0.1},{2.1,3.2,4.3,-2.0}}
};
```

Suppose there is a multidimensional array `arr[i][j][k][m]`. Then this array can holdi*j*k*m numbers of data.

Similarly,thearrayof anydimensioncanbeinitializedinC programming.

ExampleofMultidimensionalArrayInC

Write a C program to find sum of two matrix of order 2*2 using multidimensional arrays where, elements of matrix are entered by user.

```
#include<stdio.h>
int main(){
   floata[2][2],b[2][2],c[2][2];
   int i,j;
   printf("Entertheelementsof1stmatrix\n");
/*ReadingtwodimensionalArraywiththehelpoftwoforloop.If   therewasanarray
of'n'dimension,'n'numbersofloops   areneeded   for   inserting   data   to
array.*/
   for(i=0;i<2;++i)
       for(j=0;j<2;++j){
       printf("Entera%d%d:",i+1,j+1);
       scanf("%f",&a[i][j]);
       }
   printf("Entertheelementsof2ndmatrix\n");
   for(i=0;i<2;++i)
       for(j=0;j<2;++j){
       printf("Enterb%d%d:",i+1,j+1);
       scanf("%f",&b[i][j]);
       }
   for(i=0;i<2;++i)
       for(j=0;j<2;++j){
/*Writingtheelementsofmultidimensionalarrayusingloop.*/
```

```
       c[i][j]=a[i][j]+b[i][j];    /*Sumofcorrespondingelementsof two
arrays. */
       }
   printf("\nSumOfMatrix:");
   for(i=0;i<2;++i)
       for(j=0;j<2;++j){
       printf("%.1f\t",c[i][j]);
           if(j==1)             /*Todisplaymatrixsuminorder.*/
              printf("\n");
      }
return0;
}
```

Ouput

```
Entertheelementsof1stmatrix Enter
a11: 2;
Entera12:0.5;
Entera21:-1.1;
Entera22: 2;
Entertheelementsof2ndmatrix Enter
b11: 0.2;
Enterb12: 0;
Enterb21:0.23;
Enterb22: 23;

SumOfMatrix:
2.2     0.5
-0.9    25.0
```

CProgrammingArraysandFunctions

In C programming, a single arrayelementor an entire array can be passed to a function. Also,both one-dimensional and multi-dimensional array can be passed to function as argument.

PassingOne-dimensionalArrayInFunction

Cprogram topassasingle elementofanarrayto function

```
#include <stdio.h>
voiddisplay(inta)
   {
   printf("%d",a);
   }
intmain(){
   intc[]={2,3,4};
   display(c[2]);//Passingarrayelementc[2]only. return
   0;
}
```

Output

4

Singleelementofanarraycanbepassedinsimilarmanneraspassingvariabletoa function.

Passingentireone-dimensionalarraytoafunction

Whilepassingarraystotheargument,thenameof thearrayispassedas an argument(,i.e, starting address of memory area is passed as argument).

WriteaCprogramtopassanarraycontainingageofpersontoafunction.This function should find average age and display the average age in main function.

```
#include<stdio.h>
floataverage(floata[]);
int main(){
     floatavg,c[]={23.4,55,22.6,3,40.5,18};
     avg=average(c);    /*Onlynameofarrayispassedasargument.*/
     printf("Average age=%.2f",avg);
     return0;
   }
floataverage(floata[]){
     int i;
     floatavg,sum=0.0;
     for(i=0;i<6;++i){
       sum+=a[i];
     }
     avg=(sum/6);
     return avg;
}
```

Output

```
Averageage=27.08
```

PassingMulti-dimensionalArraystoFunction

To pass two-dimensional array to a function as an argument, starting address of memory area reserved is passed as in one dimensional array

Exampletopasstwo-dimensionalarraystofunction

```
#include
voidFunction(intc[2][2]);
int main(){
   int c[2][2],i,j;
   printf("Enter4numbers:\n");
   for(i=0;i<2;++i)
      for(j=0;j<2;++j){
           scanf("%d",&c[i][j]);
      }
   Function(c);    /*passingmulti-dimensionalarraytofunction*/
   return 0;
```

```
}
voidFunction(intc[2][2]){
/*Insteadtoaboveline,voidFunction(intc[][2]){isalsovalid*/ int i,j;
   printf("Displaying:\n");
   for(i=0;i<2;++i)
      for(j=0;j<2;++j)
          printf("%d\n",c[i][j]);
}
```

Output

```
Enter4numbers:
2
3
4
5
Displaying:
2
3
4
5
```

CProgrammingPointers

PointersarethepowerfulfeatureofCand(C++)programming,whichdiffersitfrom other popular programming languages like: java and Visual Basic.

Pointersareused inC programto accessthememoryand manipulatetheaddress.

Referenceoperator(&)

If*var*isavariablethen,`&var`istheaddressinmemory.

```
/*ExampletodemonstrateuseofreferenceoperatorinCprogramming.
*/
#include<stdio.h>
int main(){
  intvar=5;
  printf("Value:%d\n",var);
  printf("Address:%d",&var);//Notice,theampersand(&)beforevar. return
  0;
}
```

Output

```
Value:5
Address:2686778
```

Note: Youmay obtaindifferentvalueof addresswhile usingthis code.

In above source code, value 5 is stored in the memory location 2686778.*var* is just thename given to that location.

You,havealreadyusedreferenceoperatorinCprogramwhileusing`scanf()` function.

```
scanf("%d",&var);
```

Referenceoperator(*)andPointervariables

Pointervariable or simply pointerare the special types of variables that holds memory address rather than data, that is, a variable that holds address value is called a pointer variable or simply a pointer.

DeclarationofPointer

Dereferenceoperator(*) areusedfordefiningpointer variable

```
data_type*pointer_variable_name;
int* p;
```

Abovestatement defines,*p*as pointervariableof type `int`.

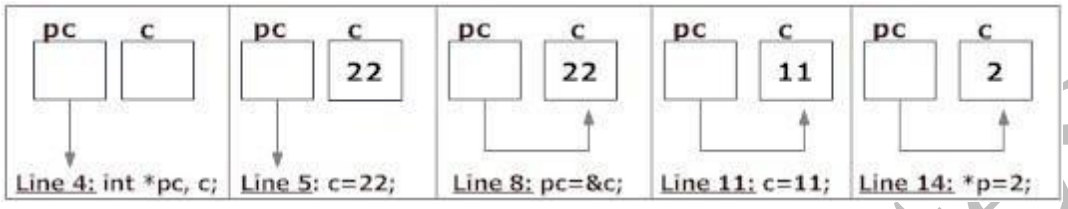
ExampleToDemonstrateWorkingofPointers

```
/*Sourcecodetodemonstrate,handlingofpointersinCprogram*/ #include
<stdio.h>
intmain(){
   int*pc;
   int c;
   c=22;
   printf("Addressofc:%d\n",&c);
   printf("Value of c:%d\n\n",c);
   pc=&c;
   printf("Address of pointer pc:%d\n",pc);
   printf("Contentofpointerpc:%d\n\n",*pc);
   c=11;
   printf("Address of pointer pc:%d\n",pc);
   printf("Contentofpointerpc:%d\n\n",*pc);
   *pc=2;
   printf("Addressofc:%d\n",&c);
   printf("Value of c:%d\n\n",c);
   return 0;
}
```

Output

```
Addressofc:2686784 Value
of c: 22

Addressofpointerpc:2686784
```

```
Contentofpointerpc: 22

Addressofpointerpc:2686784
Content of pointer pc: 11

Addressofc:2686784 Value
of c: 2
```



Explanationofprogramandfigure

1. Code `int* pc;` creates a pointer *pc* and a code `int c;` creates normal variable *c*. Pointer *pc* points to some address and that address has garbage value. Similarly, variable*c* also has garbage value at this point.
2. Code`c=22;` makes the value of c equal to 22, i.e.,22is stored in the memory location of variable *c*.
3. Codepc=&c;makespointer, pointto address of c. Note that, `&c`istheaddressofvariable *c*(because *c* is normal variable) and *pc* is the address of *pc* (because pc is the pointer variable).Sincetheaddress of pc and address of c is same, `*pc`will be equal to the value of *c*.
4. Code`c=11;` makesthevalue of *c*, 11. Since, pointer *pc*is pointing to address of *c*. Value inside address *pc* will also be 11.
5. Code `*pc=2;` change the contents of the memory location pointed by pointer *pc* to change to 2. Since address of pointer *pc* is same as address of *c*, value of *c* also changesto 2.

Commonlydonemistakesinpointers

Suppose,theprogrammerwantpointerpcto pointtotheaddressof c. Then,

```
intc,*pc;
pc=c;/*pcisaddresswhereas,cisnotanaddress.*/
*pc=&c;/*&cisaddresswhereas,*pcisnotanaddress. */
```

Inbothcases, pointerpcisnotpointing tothe addressof c.

CProgrammingPointersandArrays

Arrays are closely related to pointers in C programming but the important difference between them is that, a pointer variable can take different addresses as value whereas, in case of array it is fixed. This can be demonstrated by an example:

```
#include<stdio.h>
int main(){
   charc[4];
   int i;
   for(i=0;i<4;++i){
      printf("Addressofc[%d]=%x\n",i,&c[i]);
   }
   return0;
}
Addressofc[0]=28ff44
Addressofc[1]=28ff45
Addressofc[2]=28ff46
Addressofc[3]=28ff47
```

Notice, that there is equal difference (difference of 1 byte) between any two consecutive elements of array.

Note: Youmaygetdifferentaddressof anarray.

RelationbetweenArraysandPointers

Considerandarray:

```
intarr[4];
```

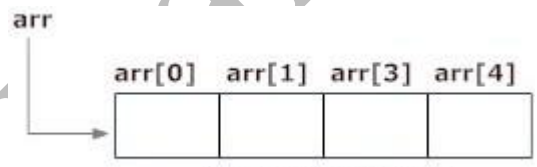


Figure: Array as Pointer

In arrays of C programming, name of the array always points to the first element of an array. Here, address of first element of an array is `&arr[0]`. Also, *arr* represents the address of the pointer where it is pointing. Hence, `&arr[0] is equivalent to` *arr*.

Also,valueinsidetheaddress`&arr[0]`andaddress*arr*areequal.Valueinaddress `&arr[0]`is`arr[0]`and valueinaddress*arr*is `*arr`.Hence,`arr[0]`isequivalentto `*arr`.

Similarly,

```
&a[1]isequivalentto(a+1)AND,a[1]isequivalentto*(a+1).
&a[2]isequivalentto(a+2)AND,a[2]isequivalentto*(a+2).
&a[3]isequivalentto(a+1)AND,a[3]isequivalentto*(a+3).
.
.
&a[i]isequivalentto(a+i)AND,a[i]isequivalentto*(a+i).
```

InC,youcandeclare anarrayandcanusepointer to alterthedataofan array.

```
//Programtofindthesumofsixnumberswitharraysandpointers. #include
<stdio.h>
intmain(){
  int i,class[6],sum=0;
  printf("Enter6numbers:\n");
  for(i=0;i<6;++i){
      scanf("%d",(class+i));//(class+i)isequivalentto&class[i] sum +=
      *(class+i); // *(class+i) is equivalent to class[i]
  }
  printf("Sum=%d",sum);
  return 0;
}
```

Output

```
Enter6numbers:
2
3
4
5
3
4
Sum=21
```

CProgrammingPointersandFunctions-CallbyReference

When,argumentispassedusingpointer,addressofthememorylocationispassedinstead of value.

ExampleofPointerAndFunctions

Programtoswaptwonumberusingcallbyreference.

```
 /*CProgramtoswaptwonumbersusingpointersandfunction.*/ #include
<stdio.h>
voidswap(int*a,int*b);
int main(){
```

```
  intnum1=5,num2=10;
  swap(&num1,&num2);   /*addressofnum1andnum2ispassedtoswap function */
  printf("Number1=%d\n",num1);
  printf("Number2 = %d",num2);
  return 0;
}
void swap(int *a,int *b){ /* pointer a and b points to address of num1
and num2 respectively */
  inttemp;
  temp=*a;
  *a=*b;
  *b=temp;
}
```

Output

```
Number1= 10
Number2=5
```

Explanation

Theaddressofmemorylocation*num1*and*num2*arepassedtofunctionandthepointers `*a`and `*b`accept those values. So, the pointer *a* and *b* points to address of *num1* and *num2* respectively. When, the value of pointer are changed, the value in memory location alsochanged correspondingly. Hence, change made to `*a`and `*b`was reflected in *num1* and *num2* in main function.

ThistechniqueisknownascallbyreferenceinC programming.

CProgrammingDynamicMemoryAllocation

The exact size of array is unknown untill the compile time,i.e., time when a compier compiles code written in a programming language into a executable form. The size of array you have declared initially can be sometimes insufficient and sometimes more than required. Dynamic memory allocation allows a program to obtain more memory space, while running or to release space when no space is required.

Although, C language inherently does not has any technique to allocated memory dynamically, there are 4 library functions under `"stdlib.h"`for dynamic memory allocation.

Function	UseofFunction
malloc()	Allocatesrequestedsizeofbytesandreturnsapointerfirstbyteofallocated space
calloc()	Allocates space for an array elements, initializes to zero and then returns a pointer tomemory

Function	UseofFunction
free()	dellocatethepreviouslyallocatedspace
realloc()	Changethesizeofpreviouslyallocatedspace

malloc()

The name malloc stands for "memory allocation". The function `malloc()` reserves a blockof memory of specified size and return a pointer of type `void` which can be casted into pointer of any form.

Syntaxofmalloc()

```
ptr=(cast-type*)malloc(byte-size)
```

Here, *ptr* is pointer of cast-type. The `malloc()` function returns a pointer to an area of memory with size of byte size. If the space is insufficient, allocation fails and returns NULL pointer.

```
ptr=(int*)malloc(100*sizeof(int));
```

This statement will allocate either 200 or 400 according to size of `int` 2 or 4 bytes respectively and the pointer points to the address of first byte of memory.

calloc()

The name calloc stands for "contiguous allocation". The only difference betweenmalloc() and calloc() is that, malloc() allocates single block of memory whereas calloc() allocates multiple blocks of memory each of same size and sets all bytes to zero.

Syntaxofcalloc()

```
ptr=(cast-type*)calloc(n,element-size);
```

This statement will allocate contiguous space in memory for an array of *n* elements. For example:

```
ptr=(float*)calloc(25,sizeof(float));
```

This statement allocates contiguous space in memory for an array of 25 elements each of size of float, i.e, 4 bytes.

free()

Dynamically allocated memory with either calloc() or malloc() does not get return on its own. The programmer must use free() explicitly to release space.

syntaxoffree()

```
free(ptr);
```

Thisstatementcausethe spaceinmemorypointerbyptr tobedeallocated.

Examplesofcalloc()andmalloc()

WriteaCprogramtofindsumofnelementsenteredbyuser.Toperformthis program, allocate memory dynamically using malloc() function.

```
#include <stdio.h>
#include<stdlib.h>
int main(){
    intn,i,*ptr,sum=0;
    printf("Enternumberofelements:");
    scanf("%d",&n);
    ptr=(int*)malloc(n*sizeof(int));//memoryallocatedusingmalloc
    if(ptr==NULL)
    {
        printf("Error!memorynotallocated.");
        exit(0);
    }
    printf("Enterelementsofarray:");
    for(i=0;i<n;++i)
    {
        scanf("%d",ptr+i);
        sum+=*(ptr+i);
    }
    printf("Sum=%d",sum);
    free(ptr);
    return0;
}
```

WriteaCprogramtofindsumofnelementsenteredbyuser.Toperformthis program, allocate memory dynamically using calloc() function.

```
#include <stdio.h>
#include<stdlib.h>
int main(){
    intn,i,*ptr,sum=0;
    printf("Enternumberofelements:");
    scanf("%d",&n);
    ptr=(int*)calloc(n,sizeof(int));
    if(ptr==NULL)
    {
        printf("Error!memorynotallocated.");
        exit(0);
    }
    printf("Enterelementsofarray:");
    for(i=0;i<n;++i)
    {
        scanf("%d",ptr+i);
        sum+=*(ptr+i);
    }
    printf("Sum=%d",sum);
    free(ptr);
```

```
    return0;
}
```

realloc()

If the previously allocated memory is insufficient or more than sufficient. Then, you can change memory size previously allocated using realloc().

Syntaxofrealloc()

```
ptr=realloc(ptr,newsize);
```

Here,*ptr*isreallocated withsizeof newsize.

```
#include <stdio.h>
#include<stdlib.h>
int main(){
    int*ptr,i,n1,n2;
    printf("Enter size of array: ");
    scanf("%d",&n1);
    ptr=(int*)malloc(n1*sizeof(int));
    printf("Addressofpreviouslyallocatedmemory:");
    for(i=0;i<n1;++i)
         printf("%u\t",ptr+i);
    printf("\nEnternewsizeofarray:");
    scanf("%d",&n2);
    ptr=realloc(ptr,n2);
    for(i=0;i<n2;++i)
         printf("%u\t",ptr+i);
    return 0;
}
```

CProgrammingArrayandPointerExamples

This page contains examples and source code on arrays and pointers. To understand allprogram on this page, you should have knowledge of followingarray and pointer topics:

1. Arrays
2. Multi-dimensionalArrays
3. Pointers
4. ArrayandPointerRelation
5. CallbyReference
6. DynamicMemoryAllocation

ArrayandPointerExamples

CProgrammingExamplesandSourceCode

CProgramtoCalculateAverageUsing Arrays

This program takes *n* number of element from user(where, n is specified by user), stores data in an array and calculates the average of those numbers.

SourceCodetoCalculateAverageUsingArrays

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

CProgrammingExamplesandSourceCode

```
intmain(){
    intn,i;
    float num[100], sum=0.0, average;
    printf("Enterthenumbersofdata:");
    scanf("%d",&n);
    while(n>100||n<=0)
    {
        printf("Error!numbershouldinrangeof(1to100).\n"); printf("Enter
        the number again: ");
        scanf("%d",&n);
    }
   for(i=0;i<n;++i)
   {
      printf("%d.Enternumber:",i+1); scanf("%f",&num[i]);
      sum+=num[i];
   }
   average=sum/n;
   printf("Average=%.2f",average); return
   0;
}
```

Output

```
Enterthenumbersofdata:6
1. Enternumber:45.3
2. Enternumber:67.5
3. Enternumber:-45.6
4. Enternumber:20.34
5. Enternumber:33
6. Enternumber:45.6
Average=27.69
```

Thisprogramcalculatestheaverageifthenumberofdataarefrom1to100.Ifuserenters value of *n* above 100 or below 100 then, while loop is executed which asks user to enter value of *n* until it is between 1 and 100.

CProgramtoFindLargestElementofanArray

This program takes *n* number of element from user(where, *n* is specified by user) and stores data in an array. Then, this program displays thelargest element ofthat array using loops.

SourceCodetoDisplayLargestElementofanarray

```
#include<stdio.h>
int main(){
    int i,n;
    floatarr[100];
    printf("Entertotalnumberofelements(1to100):");
    scanf("%d",&n);
```

```
    printf("\n");
    for(i=0;i<n;++i)/*Storesnumberenteredbyuser.*/
    {
       printf("EnterNumber%d:",i+1);
       scanf("%f",&arr[i]);
    }
    for(i=1;i<n;++i)/*Looptostorelargestnumbertoarr[0] */
    {
       if(arr[0]<arr[i]) /* Change < to > if you want to find smallest
element*/
           arr[0]=arr[i];
    }
    printf("Largestelement=%.2f",arr[0]);
    return 0;
}
```

Output

```
Entertotalnumberofelements(1to100):8

EnterNumber1:23.4
EnterNumber2:-34.5
EnterNumber3: 50
EnterNumber4:33.5
EnterNumber5:55.5
EnterNumber6:43.7
EnterNumber7: 5.7
EnterNumber8:-66.5
```

This program takes *n* number of elements from user and stores it in array *arr[]*. To find the largest element, the first two elements of array are checked and largest of these two element is placed in *arr[0]*. Then, the first and third elements are checked and largest of these two element is placed in *arr[0]*. This process continues until and first and last elements are checked. After this process, the largest element of an array will be in *arr[0]* position.

CProgramtoCalculateStandard Deviation

Thisprogramcalculates thestandarddeviationofindividualseriesusingarrays.Visitthis page to learn about Standard Deviation.

In thisprogram,elementsofarrays areusedforstoringthedataand thisarrayispassedto function which calculates standard deviation and finally the result(standard deviation) is displayed in `main()` function.

SourceCodetoCalculateStandardDeviationbyPassingittoFunction

```
/*Sourcecodetocalculatestandarddeviation.*/
```

CProgrammingExamplesandSourceCode

```
#include<stdio.h>
#include <math.h>
floatstandard_deviation(floatdata[],intn); int
main()
{
    intn,i;
    floatdata[100];
    printf("Enternumberofdatas(shouldbelessthan100):"); scanf("%d",&n);
    printf("Enterelements:");
    for(i=0; i<n; ++i)
        scanf("%f",&data[i]);
    printf("\n");
    printf("StandardDeviation=%.2f",standard_deviation(data,n)); return
    0;
}
floatstandard_deviation(floatdata[],intn)
{
    floatmean=0.0,sum_deviation=0.0; int
    i;
    for(i=0;i<n;++i)
    {
        mean+=data[i];
    }
    mean=mean/n;
    for(i=0;i<n;++i)
    sum_deviation+=(data[i]-mean)*(data[i]-mean);
    return sqrt(sum_deviation/n);
}
```

Output

```
Enternumberofdatas(shouldbelessthan100):6 Enter
elements: 12
24.5
65.4
10.3
29.9
34.3
```

CProgramtoAddTwoMatrixUsingMulti-dimensional Arryas

This program asks user to enter the size of the matrix (rows and column) then, it asks the user to enter the elements of two matrices and finally it adds two matrix and displays the result.

SourceCodetoAddTwoMatrixinCprogramming

```
#include<stdio.h>
int main(){
    intr,c,a[100][100],b[100][100],sum[100][100],i,j;
    printf("Enternumberofrows(between1and100):");
```

```
    scanf("%d",&r);
    printf("Enternumberofcolumns(between1and100):");
    scanf("%d",&c);
    printf("\nEnterelementsof1stmatrix:\n");

/*Storingelementsoffirstmatrixenteredbyuser.*/

    for(i=0;i<r;++i)
       for(j=0;j<c;++j)
       {
           printf("Enterelementa%d%d:",i+1,j+1);
           scanf("%d",&a[i][j]);
       }

/*Storingelementsofsecondmatrixenteredbyuser.*/

    printf("Enterelementsof2ndmatrix:\n");
    for(i=0;i<r;++i)
       for(j=0;j<c;++j)
       {
           printf("Enterelementa%d%d:",i+1,j+1);
           scanf("%d",&b[i][j]);
       }

/*AddingTwomatrices*/

   for(i=0;i<r;++i)
       for(j=0;j<c;++j)
           sum[i][j]=a[i][j]+b[i][j];

/*Displayingtheresultantsummatrix.*/

    printf("\nSumoftwomatrixis:\n\n"); for(i=0;i<r;++i)
       for(j=0;j<c;++j)
       {
           printf("%d   ",sum[i][j]);
           if(j==c-1)
               printf("\n\n");
       }

    return0;
}
```

Output

```
Enterelementa12:-4 Enter
element a21: 8 Enter
element a22: 5 Enter
element a31: 1 Enter
element a32: 0
Enterelementsof2ndmatrix: Enter
element a11: 4
Enterelementa12:-7
```

```
Enterelementa21:9
Enterelementa22:1
Enterelementa31:4
Enterelementa32:5

Sumoftwomatrixis:

8   -11

17   6

5   5
```

CProgramtoMultiplytoMatrixUsingMulti-dimensionalArrays

This program asks user to enter two matrices and this program multiplies these twomatrix and displays it. If you don't know matrix multiplication, visit this page to learn, how two matrix can be multiplied.

SourcecodetomultiplytomatrixinCprogramming

```
#include<stdio.h>
int main()
{
    inta[10][10],b[10][10],mult[10][10],r1,c1,r2,c2,i,j,k;
    printf("Enterrowsandcolumnforfirstmatrix:"); scanf("%d%d", &r1,
    &c1);
    printf("Enterrowsandcolumnforsecondmatrix:"); scanf("%d%d",&r2,
    &c2);

/*Ifcolumoffirstmatrixinnotequaltorowofsecondmatrix, asking user to
enter the size of matrix again. */
    while(c1!=r2)
    {
        printf("Error!columnoffirstmatrixnotequaltorowof second.\n\n");
        printf("Enterrowsandcolumnforfirstmatrix:"); scanf("%d%d",
        &r1, &c1);
        printf("Enterrowsandcolumnforsecondmatrix:"); scanf("%d%d",&r2,
        &c2);
    }

/* Storing elements of first matrix. */
    printf("\nEnterelementsofmatrix1:\n"); for(i=0;
    i<r1; ++i)
    for(j=0;j<c1;++j)
    {
        printf("Enterelementsa%d%d:",i+1,j+1);
        scanf("%d",&a[i][j]);
    }

/* Storing elements of second matrix. */
    printf("\nEnterelementsofmatrix2:\n");
    for(i=0;i<r2;++i)
```

```
    for(j=0;j<c2;++j)
    {
        printf("Enterelementsb%d%d:",i+1,j+1);
        scanf("%d",&b[i][j]);
    }

/*Initializingelementsofmatrixmultto0.*/ for(i=0;
    i<r1; ++i)
    for(j=0;j<c2;++j)
    {
       mult[i][j]=0;
    }

/*Multiplyingmatrixaandbandstoringinarraymult.*/ for(i=0;
    i<r1; ++i)
    for(j=0;j<c2;++j)
    for(k=0;k<c1;++k)
    {
        mult[i][j]+=a[i][k]*b[k][j];
    }

/*Displayingthemultiplicationoftwomatrix.*/
    printf("\nOutput Matrix:\n");
    for(i=0;i<r1;++i)
    for(j=0;j<c2;++j)
    {
        printf("%d",mult[i][j]);
        if(j==c2-1)
            printf("\n\n");
    }
    return0;
}
```

Output

```
Enterrowsandcolumnforfirstmatrix:3 2
Enterrowsandcolumnforsecondmatrix:3 2
Error!columnoffirstmatrixnotequaltorowofsecond.

Enterrowsandcolumnforfirstmatrix:2 3
Enterrowsandcolumnforsecondmatrix:3 2

Enterelementsofmatrix1: Enter
elements a11: 3
Enterelementsa12:-2 Enter
elements a13: 5 Enter
elements a21: 3 Enter
elements a22: 0
Enterelementsa23:4
```

```
Enterelementsofmatrix2:
Enter elements b11: 2
Enter elements b12: 3
Enterelementsb21:-9 Enter
elements b22: 0 Enter
elements b31: 0 Enter
elements b32: 4

OutputMatrix:
2429

625
```

In this program, user is asked to enter the size of two matrix at first. The column of first matrix should be equal to row of second matrix for multiplication. If this condition is not satisfied then, the size of matrix is again asked using while loop. Then, user is asked to enter two matrix and finally the output of two matrix is calculated and displayed.

This program is little bit larger and it is better to solve this program by passing it to a function. Visit this page to learn about multiplying matrices by passing arrays to afunction.

CProgramtoFindTransposeofa Matrix

Thisprogramasksusertoenteramatrix(sizeofmatrixisspecifiedbyuser)andthis program finds the transpose of that matrix and displays it.

SourceCodetoFindTransposeofaMatrix

```
#include<stdio.h>
int main()
{
    int a[10][10], trans[10][10], r, c, i, j;
    printf("Enterrowsandcolumnofmatrix:"); scanf("%d
    %d", &r, &c);

/*Storingelementofmatrixenteredbyuserinarraya[][].*/ printf("\nEnter
    elements of matrix:\n");
    for(i=0;i<r;++i)
    for(j=0;j<c;++j)
    {
        printf("Enterelementsa%d%d:",i+1,j+1);
        scanf("%d",&a[i][j]);
    }
/* Displaying the matrix a[][] */
    printf("\nEnteredMatrix:\n");
    for(i=0; i<r; ++i)
    for(j=0;j<c;++j)
    {
        printf("%d",a[i][j]);
        if(j==c-1)
```

```
            printf("\n\n");
    }

/* Findingtransposeofmatrixa[][]andstoringitinarraytrans[][].
*/
    for(i=0;i<r;++i)
    for(j=0;j<c;++j)
    {
       trans[j][i]=a[i][j];
    }

/*Displayingthetranspose,i.e,Displayingarraytrans[][].*/
    printf("\nTranspose of Matrix:\n");
    for(i=0;i<c;++i)
    for(j=0;j<r;++j)
    {
        printf("%d",trans[i][j]);
        if(j==r-1)
            printf("\n\n");
    }
    return0;
}
```

Output

```
Enterrowsandcolumnofmatrix:2 3

Enterelementsofmatrix:
Enter elements a11: 1
Enter elements a12: 2
Enter elements a13: 9
Enter elements a21: 0
Enter elements a22: 4
Enter elements a23: 7

EnteredMatrix:
129

047


TransposeofMatrix:
10

24

97
```

<u>CProgramtoMultiplytwoMatricesbyPassingMatrixto Function</u>

This program asks user to enter the size of the matrix (rows and column) then, it asks the usertoentertheelementsoftwomatricesandfinallyitaddstwomatrixanddisplaysthe

result.Toperformthistaskthreefunctionsare made:

1. Totakesmatrixelementsfromuser
2. Tomultiplytwomatrix
3. Todisplaytheresultantmatrixaftermultiplication

SourceCodetoMultiplyMatrixbyPassingittoaFunction

```
#include<stdio.h>
voidtake_data(inta[][10],intb[][10],intr1,intc1,intr2,int c2);
voidmultiplication(inta[][10],intb[][10],intmult[][10],intr1,int c1,int
r2,int c2);
voiddisplay(intmult[][10],intr1,intc2); int
main()
{
    inta[10][10],b[10][10],mult[10][10],r1,c1,r2,c2,i,j,k;
    printf("Enterrowsandcolumnforfirstmatrix:");
    scanf("%d%d", &r1, &c1);
    printf("Enterrowsandcolumnforsecondmatrix:"); scanf("%d%d",&r2,
    &c2);

/*Ifcolumoffirstmatrixinnotequaltorowofsecondmatrix, asking user to
enter the size of matrix again. */

    while(c1!=r2)
    {
        printf("Error!columnoffirstmatrixnotequaltorowof second.\n");
        printf("Enterrowsandcolumnforfirstmatrix:"); scanf("%d%d", &r1,
        &c1);
        printf("Enterrowsandcolumnforsecondmatrix:"); scanf("%d%d",&r2,
        &c2);
    }
    take_data(a,b,r1,c1,r2,c2);/* Function to take matices data */
    multiplication(a,b,mult,r1,c1,r2,c2);/*Functiontomultiplytwo
matrices.*/
    display(mult,r1,c2); /* Function to display resultant matrix after
multiplication. */
    return0;
}

voidtake_data(inta[][10],intb[][10],intr1,intc1,intr2,intc2)
{
    int i,j;
    printf("\nEnterelementsofmatrix1:\n");
    for(i=0; i<r1; ++i)
    for(j=0;j<c1;++j)
    {
        printf("Enterelementsa%d%d:",i+1,j+1);
        scanf("%d",&a[i][j]);
    }

    printf("\nEnterelementsofmatrix2:\n");
    for(i=0; i<r2; ++i)
```

```
    for(j=0;j<c2;++j)
    {
        printf("Enterelementsb%d%d:",i+1,j+1);
        scanf("%d",&b[i][j]);
    }
}

voidmultiplication(inta[][10],intb[][10],intmult[][10],intr1,int c1,int
r2,int c2)
{
    inti,j,k;
/*Initializingelementsofmatrixmultto0.*/
    for(i=0; i<r1; ++i)
    for(j=0;j<c2;++j)
    {
       mult[i][j]=0;
    }
/*Multiplyingmatrixaandbandstoringinarraymult.*/ for(i=0; i<r1;
    ++i)
    for(j=0;j<c2;++j)
    for(k=0;k<c1;++k)
    {
        mult[i][j]+=a[i][k]*b[k][j];
    }
}

voiddisplay(intmult[][10],intr1,intc2)
{
    inti,j;
    printf("\nOutputMatrix:\n");
    for(i=0; i<r1; ++i)
    for(j=0;j<c2;++j)
    {
        printf("%d",mult[i][j]);
        if(j==c2-1)
            printf("\n\n");
    }
}
```

Output

```
Enterrowsandcolumnforfirstmatrix:3 2
Enterrowsandcolumnforsecondmatrix:3 2
Error!columnoffirstmatrixnotequaltorowofsecond.

Enterrowsandcolumnforfirstmatrix:2 3
Enterrowsandcolumnforsecondmatrix:3 2

Enterelementsofmatrix1:
Enter elements a11: 3
```

```
Enterelementsa12:-2 Enter
elements a13: 5 Enter
elements a21: 3 Enter
elements a22: 0 Enter
elements a23: 4

Enterelementsofmatrix2: Enter
elements b11: 2
Enter elements b12: 3
Enterelementsb21:-9 Enter
elements b22: 0 Enter
elements b31: 0 Enter
elements b32: 4

OutputMatrix:
2429

625
```

CProgramtoSortElementsofanArray

CProgramtoAccessElementsofanArrayUsing Pointer

Thisprogramdeclaresthearrayoffiveelementandtheelementsofthatarrayareaccessed using pointer.

SourceCodetoAccessArrayElementsUsingPointer

```
#include<stdio.h>
int main(){
   int data[5], i;
   printf("Enterelements:");
   for(i=0;i<5;++i)
     scanf("%d",data+i);
   printf("Youentered:");
   for(i=0;i<5;++i)
      printf("%d\n",*(data+i));
   return 0;
}
```

Output

```
Enterelements:1
2
3
5
4
Youentered:1
2
3
5
4
```

CProgrammingExamplesandSourceCode

Visitthispagetolearn aboutrelationshipbetweenpointerand arrays.

CProgramSwapNumbersinCyclicOrderUsingCallby Reference

This program takes three enters from user which is stored in variable a, b and c respectively. Then, these variables are passed to function using call by reference. This function swaps the value of these elements in cyclic order.

CProgramtoSwapElementsUsingCallbyReference

```
#include<stdio.h>
voidCycle(int*a,int*b,int*c); int
main(){
    inta,b,c;
    printf("Entervalueofa,bandcrespectively:");
    scanf("%d%d%d",&a,&b,&c);
    printf("Valuebeforeswapping:\n");
    printf("a=%d\nb=%d\nc=%d\n",a,b,c);
    Cycle(&a,&b,&c);
    printf("Valueafterswappingnumbersincycle:\n");
    printf("a=%d\nb=%d\nc=%d\n",a,b,c);
    return0;
}
voidCycle(int*a,int*b,int*c){ int
    temp;
    temp=*b;
    *b=*a;
    *a=*c;
    *c=temp;
}
```

Output

```
Entervalueofa,bandcrespectively:1 2
3
Valuebeforeswapping:
a=1
b=2
c=3
Valueafterswappingnumbersincycle: a=3
b=1
c=2
```

CProgramtoFindLargestNumberUsingDynamicMemoryAllocation

In thisprogram,calloc() functionisused toallocatethememorydynamically.Depending uponthe numberof elements,the requiredsizeisallocated whichpreventsthe wastage of

memory.Ifnomemoryisallocated,errorisdisplayedandtheprogramisterminated.

SourceCodetoFindLargestElementUsingDynamicMemoryAllocation

```
#include <stdio.h>
#include<stdlib.h>
int main(){
    int i,n;
    float*data;
    printf("Entertotalnumberofelements(1to100):"); scanf("%d",&n);
    data=(float*)calloc(n,sizeof(float));    /* Allocates the memory for
'n' elements */
    if(data==NULL)
    {
        printf("Error!!!memorynotallocated.");
        exit(0);
    }
    printf("\n");
    for(i=0;i<n;++i)/*Storesnumberenteredbyuser.*/
    {
       printf("EnterNumber%d:",i+1);
       scanf("%f",data+i);
    }
    for(i=1;i<n;++i)   /*Looptostorelargestnumberataddressdata
*/
    {
       if(*data<*(data+i))/*Change<to>ifyouwanttofind smallest number */
           *data=*(data+i);
    }
    printf("Largestelement=%.2f",*data); return
    0;
}
```

Output

```
Entertotalnumberofelements(1to100):10 Enter

Number 1: 2.34
EnterNumber2:3.43
EnterNumber3:6.78
EnterNumber4:2.45
EnterNumber5:7.64
EnterNumber6:9.05
EnterNumber7:-3.45
EnterNumber8:-9.99
EnterNumber9:5.67
```

```
EnterNumber10:34.95
Largestelement:34.95
```

CProgrammingString

InCprogramming,arrayofcharacterarecalledstrings.Astringisterminatedbynull character `/0`. For example:

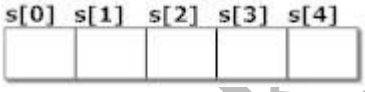
```
"cstringtutorial"
```

Here,"cstringtutorial"isastring.When,compilerencountersstrings,itappendsnull character at the end of string.

Declarationofstrings

Strings are declared in C in similar manner as arrays. Only difference is that, strings are of `char`type.

```
chars[5];
```



Stringscanalso bedeclaredusing pointer.

```
char *p
```

Initializationofstrings

InC,stringcanbeinitializedindifferentnumberofways.

```
charc[]="abcd";
     OR,
charc[5]="abcd";
     OR,
charc[]={'a','b','c','d','\0'};
     OR;
charc[5]={'a','b','c','d','\0'};
```

c[0]	c[1]	c[2]	c[3]	c[4]
a	b	c	d	\0

Stringcanalso beinitializedusing pointers

```
char*c="abcd";
```

ReadingStringsfrom user.

Readingwordsfromuser.

```
char c[20];
scanf("%s",c);
```

String variable*c* can only take a word.It is beacause whenwhite space is encountered,the `scanf()` function terminates.

WriteaC program to illustratehow toread stringfromterminal.

```
#include<stdio.h>
int main(){
    char name[20];
    printf("Entername:");
    scanf("%s",name);
    printf("Yournameis%s.",name);
    return 0;
}
```

Output

```
Entername:DennisRitchie
Your name is Dennis.
```

Here, program will ignore Ritchie because, `scanf()` function takes only string before the white space.

Readingalineoftext

Cprogram toreadlineoftextmanually.

```
#include<stdio.h>
int main(){
    charname[30],ch;
    int i=0;
    printf("Entername:");
    while(ch!='\n')    //terminatesifuserhitenter
    {
        ch=getchar();
        name[i]=ch;
        i++;
    }
    name[i]='\0';       //insertingnullcharacteratend
    printf("Name: %s",name);
    return0;
```

```
}
```

This process to take string is tedious. There are predefined functions `gets()` and `puts` in C language to read and display string respectively.

```
intmain(){
    char name[30];
    printf("Entername:");
    gets(name);     //Functiontoreadstringfromuser.
    printf("Name: ");
    puts(name);    //Functiontodisplaystring.
    return 0;
}
```

Both,theaboveprogramhassameoutput below:

Output

```
Entername:TomHanks Name:
Tom Hanks
```

PassingStringstoFunctions

Stringcanbepassedtofunctioninsimilarmannerasarraysas,stringisalsoanarray. Learn more about passing array to a function.

```
#include<stdio.h>
voidDisplay(charch[]);
int main(){
    charc[50];
    printf("Enterstring:");
    gets(c);
    Display(c);     //Passingstringctofunction.
    return 0;
}
void Display(char ch[]){
    printf("StringOutput:");
    puts(ch);
}
```

Here,string`c`ispassedfrom`main()` functiontouser-definedfunction`Display()`.In function declaration, `ch[]` is the formal argument.

Stringhandlingfunctions

You can perform different type of string operations manually like: finding length of string, concatenating(joining) two strings etc. But, for programmers ease, many library function are defined under header file `<string.h>`to handle these commonly used talk in C programming. You will learn more about string hadling function in next chapter.

StringManipulationsInCProgrammingUsingLibraryFunctions

Strings are often needed to be manipulated by programmer according to the need of a problem. All string manipulation can be done manually by the programmer but, this makes programming complex and large. To solve this, the C supports a large number of string handling functions.

There are numerous functions defined in `"string.h"`header file. Few commonly used string handling functions are discussed below:

Function	**WorkofFunction**
strlen()	Calculatesthelengthof string
strcpy()	Copiesastringtoanotherstring
strcat()	Concatenates(joins)twostrings
strcmp()	Comparestwo string
strlwr()	Convertsstringtolowercase
strupr()	Convertsstringto uppercase

Strings handling functions are defined under `"string.h"`header file, i.e, you have to include the code below to run string handling functions.

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

gets()andputs()

Functions gets() and puts() are two string functions to take string input from user and display string respectively as mentioned in previous chapter.

```
#include<stdio.h>in
t main(){
    char name[30];
    printf("Entername:");
    gets(name);      //Functiontoreadstringfromuser.
    printf("Name: ");
    puts(name);     //Functiontodisplaystring.
    return 0;
}
```

Though,gets()and`puts()`functionhandlestring,boththesefunctionsaredefinedin `"stdio.h"`headerfile.

StringExamplesinCProgramming

This page contains examples and source code on strings in C programming. To understand all the example on this page, you should have basic knowledge of string, passing string to function and few commonly used standard library functions tomanipulate strings.

ExamplesofStringsinCProgramming

CProgrammingExamplesandSourceCode

1.CProgramtoFindtheFrequencyofCharactersinaString

Thisprogramasksusertoenterastringandacharacterandthisprogramchecks how many times that character is repeated in the string entered by user.

SourceCodetoFindtheFrequencyofCharacters

```
#include<stdio.h>
int main(){
   charc[1000],ch;
   int i,count=0;
   printf("Enterastring:");
   gets(c);
   printf("Enteracharactertofindfrequency:");
   scanf("%c",&ch);
   for(i=0;c[i]!='\0';++i)
   {
       if(ch==c[i])
           ++count;
   }
   printf("Frequencyof%c=%d",ch,count); return 0;
}
```

Output

```
Enterastring:Thiswebsiteisawesome. Enter
a character to find frequency: e
Frequencyofe=4
```

2.CProgramtoFindtheNumberofVowels,Consonants,DigitsandWhitespaceinaString

This program takes a string from user and finds the total number of vowels, consonants, digits and white space present in that string.

SourceCodetoFindNumberofVowels,Consonants,DigitsandWhiteSpaceCharacter

```
#include<stdio.h>in
t main(){
    charline[150];inti,
    v,c,ch,d,s,o;
    o=v=c=ch=d=s=0;
    printf("Enteralineofstring:\n");
    gets(line);
    for(i=0;line[i]!='\0';++i)
    {
        if(line[i]=='a'||line[i]=='e'||line[i]=='i'||line[i]=='o'
||line[i]=='u'||line[i]=='A'||line[i]=='E'||line[i]=='I'|| line[i]=='O'
|| line[i]=='U')
            ++v;
        else   if((line[i]>='a'&&   line[i]<='z')   ||   (line[i]>='A'&&
line[i]<='Z'))
            ++c;
        elseif(line[i]>='0'&&c<='9')
            ++d;
        elseif(line[i]==' ')
            ++s;
    }
    printf("Vowels: %d",v);
    printf("\nConsonants: %d",c);
    printf("\nDigits: %d",d);
    printf("\nWhitespaces:%d",s);
    return 0;
}
```

Output

```
Enteralineofstring:
Thisprogramiseasy2understand Vowels:
9
Consonants:18
Digits:1
Whitespaces:5
```

CProgramtoReverseaStringbyPassingittoFunction

HereNo Program.

4.CProgramtoFindtheLengthofaString

Youcanusestandardlibraryfunctionstrlen()tofindthelengthofastringbut,this programcomputesthelengthofastringmanuallywithoutusingstrlen()funtion.

SourceCodetoCalculatedLengthwithoutUsingstrlen() Function

```
#include<stdio.h>
int main()
{
    char s[1000],i;
    printf("Enterastring:");
    scanf("%s",s);
    for(i=0; s[i]!='\0'; ++i);
    printf("Lengthofstring:%d",i); return
    0;
}
```

Output

```
Enterastring:Programiz Length
of string: 9
```

This program asks user to enter astring and computes thelength ofstring manually using for loop.

5.CprogramtoConcatenateTwoStrings

You can concatenate two strings easily using standard library functionstrcat() but, thisprogram concatenates two strings manually without using `strcat()` function.

SourceCodetoConcatenateTwoStringsManually

```
#include<stdio.h>
int main()
{
    char s1[100], s2[100], i, j;
    printf("Enterfirststring:");
    scanf("%s",s1);
    printf("Entersecondstring:");
    scanf("%s",s2);
    for(i=0;s1[i]!='\0';++i);/*icontainslengthofstrings1.*/ for(j=0;
    s2[j]!='\0'; ++j, ++i)
    {
        s1[i]=s2[j];
    }
    s1[i]='\0';
    printf("Afterconcatenation:%s",s1);
    return 0;
}
```

Output

```
Enter first string: lol
Enter second string: :)
Afterconcatenation:lol:)
```

6.**CProgramtoCopyaStringwithoutusing`strcpy()`function.**

You can use the strcpy() function to copy the content of one string to another but, this program copies the content of one string to another manually without using `strcpy()` function.

SourceCodetoCopyStringManually

```
#include<stdio.h>
int main()
{
    char s1[100], s2[100], i;
    printf("Enterstrings1:");
    scanf("%s",s1);
    for(i=0;s1[i]!='\0';++i)
    {
        s2[i]=s1[i];
    }
    s2[i]='\0';
    printf("Strings2:%s",s2);
    return 0;
}
```

Output

```
EnterStrings1:programiz
String s2: programiz
```

This aboveprogram copies the content ofstrings*1*to string *s2*manually.

7.CProgramtoRemoveallCharactersinaStringexceptalphabet

Thisprogramtakesastringsfromuserandremovesallcharactersinthatstringexcept alphabets.

SourceCodetoRemoveCharactersinStringExceptAlphabets

```
#include<stdio.h>in
t main(){
    charline[150];
```

```
    int i,j;
    printf("Enterastring:");
    gets(line);
    for(i=0;line[i]!='\0';++i)
    {
        while             (!((line[i]>='a'&&line[i]<='z')              ||
(line[i]>='A'&&line[i]<='Z'||line[i]=='\0')))
        {
            for(j=i;line[j]!='\0';++j)
            {
                line[j]=line[j+1];
            }
            line[j]='\0';
        }
    }
    printf("OutputString:");
    puts(line);
    return0;
}
```

Output

```
Enterastring:p2'r"o@gram84iz./ Output
String: programiz
```

This program takes a string from user and for loop executed until all characters of string is checked. If any character inside a string is not a alphabet, all characters after it including null character is shifted by 1 position backwards.

8.CProgramtoSortElementsinLexicographicalOrder(DictionaryOrder)

This program takes 10 words from user and sorts elements in lexicographical order. Toperform this task, two dimensional string is used.

SourceCodetoSortWordsinDictionaryOrder

```
#include<stdio.h>#i
nclude<string.h>
int main(){
    int i,j;
    char str[10][50],temp[50];
    printf("Enter10words:\n");
    for(i=0;i<10;++i)
        gets(str[i]);
    for(i=0;i<9;++i)
       for(j=i+1;j<10 ;++j){
          if(strcmp(str[i],str[j])>0)
          {
            strcpy(temp,str[i]);
```

```
            strcpy(str[i],str[j]);
            strcpy(str[j],temp);
          }
    }
    printf("Inlexicographicalorder:\n");
    for(i=0;i<10;++i){
       puts(str[i]);
    }
 return0;
}
```

Output

```
Enter10words: fortran
java
perl
python
php
javascript
c
cpp
ruby
csharp

Inlexicographicalorder: c
cpp
csharp
fortran
java
javascript
perl
php
python
ruby
```

9.CProgramtoChangeDecimaltoHexadecimalNumberandViceVersa

HereNo Program

10.CProgramtoConvertHexadecimaltoOctalandViceVersa Here No Program

CProgramtoConvertBinaryNumbertoHexadecimalViceVersa

HereNo Program

UNIT-4

CProgrammingStructure

Structure is the collection of variables of different types under a single name for better handling. For example: You want to store the information about person about his/her name, citizenship number and salary. You can create these information separately but, better approach will be collection of these information under single name because all these information are related to person.

StructureDefinitioninC

Keyword`struct`isused forcreatingastructure.

Syntaxofstructure

```
structstructure_name
{
    data_typemember1;
    data_typemember2;
    .
    .
    data_typememeber;
};
```

Wecan createthestructureforapersonasmentioned aboveas:

```
structperson
{
    charname[50];
    int cit_no;
    float salary;
};
```

Thisdeclarationabovecreatesthederiveddatatype**`structperson`**.

Structurevariabledeclaration

Whena structure isdefined,itcreates auser-definedtypebut,nostorageis allocated. For the above structure of person, variable can be declared as:

```
structperson
{
    charname[50];
    int cit_no;
    float salary;
};
```

```
Insidemainfunction:
structpersonp1,p2,p[20];
```

Anotherwayofcreatingsturcturevariableis:

```
structperson
{
    charname[50];
    int cit_no;
    float salary;
}p1,p2,p[20];
```

Inbothcases,2variables*p1*,*p2*andarray*p*having20elementsoftype**structperson** arecreated.

Accessingmembersofa structure

Therearetwotypes ofoperators usedforaccessing membersofastructure.

1. Memberoperator(.)
2. Structurepointeroperator(->)(willbediscussedinstructureandpointerschapter)

Any member of a structure can be accessed as: `structure_variable_name.member_name`

Suppose,wewantto accesssalaryfor variable *p2*.Then,it canbe accessedas:

```
p2.salary
```

Exampleofstructure

Write a C program to add two distances entered by user. Measurement of distance should be in inch and feet.(Note: 12 inches = 1 foot)

```
#include<stdio.h>
struct Distance{
    int feet;
    floatinch;
}d1,d2,sum;
intmain(){
    printf("1stdistance\n");
    printf("Enter feet: ");
    scanf("%d",&d1.feet);   /*inputoffeetforstructurevariabled1
*/
    printf("Enterinch:");
```

```
    scanf("%f",&d1.inch);   /*inputofinchforstructurevariabled1
*/
    printf("2nddistance\n");
    printf("Enter feet: ");
    scanf("%d",&d2.feet);   /*inputoffeetforstructurevariabled2
*/
    printf("Enterinch:");
    scanf("%f",&d2.inch);   /*inputofinchforstructurevariabled2
*/
    sum.feet=d1.feet+d2.feet;
    sum.inch=d1.inch+d2.inch;
    if(sum.inch>12){     //Ifinchisgreaterthan12,changingitto
feet.
        ++sum.feet;
        sum.inch=sum.inch-12;
    }
    printf("Sumofdistances=%d\'-%.1f\"",sum.feet,sum.inch);
/*printingsumofdistanced1andd2*/ return 0;
}
```

Output

```
1st distance
Enterfeet:12
Enterinch:7.9
2nd distance
Enter feet: 2
Enterinch:9.8
Sumofdistances=15'-5.7"
```

<u>Keywordtypedefwhileusingstructure</u>

Programmergenerally`usetypedef`whileusingstructureinClanguage. For example:

```
typedefstructcomplex{ int
  imag;
  floatreal;
}comp;

Insidemain:
compc1,c2;
```

Here,`typedef`keywordisusedincreatingatype*comp*(whichisoftypeas**`struct complex`**). Then, two structure variables *c1* and *c2* are created by this *comp* type.

<u>Structureswithinstructures</u>

Structurescanbenested withinotherstructuresinC programming.

```
structcomplex
{
 int imag_value;
 floatreal_value;
};
structnumber{structco
   mplexc1; int real;
}n1,n2;
```

Supposeyouwanttoaccess*imag_value*for*n2*structurevariablethen,structuremember *n1.c1.imag_value*isused.

CProgrammingStructureandPointer

Pointerscanbeaccessedalongwithstructures.Apointervariableofstructurecanbe created as below:

```
structname{
    member1;
    member2;
    .
    .
};
--------Insidefunction------- struct
name *ptr;
```

Here, the pointer variable of type **`struct name`**is created.

Structure'smemberthroughpointercanbeusedintwoways:

1. Referencingpointertoanother addresstoaccessmemory
2. Usingdynamicmemoryallocation

Consideranexampletoaccessstructure'smemberthroughpointer.

```
#include<stdio.h>
struct name{
   int a;
   floatb;
};
intmain(){
    structname*ptr,p;
    ptr=&p;            /*Referencingpointertomemoryaddressofp*/
    printf("Enter integer: ");
    scanf("%d",&(*ptr).a);
    printf("Enternumber:");
```

```
    scanf("%f",&(*ptr).b);
    printf("Displaying: ");
    printf("%d%f",(*ptr).a,(*ptr).b);
    return 0;
}
```

Inthisexample,thepointervariableoftype**structname**isreferencedtotheaddressof *p*.Then,only thestructurememberthroughpointer cancan accessed.

Structurepointermembercanalsobeaccessedusing->operator.

```
(*ptr).aissameasptr->a
(*ptr).bissameasptr->b
```

AccessingstructurememberthroughpointerusingdynamicmemoryallocationTo access structure member using pointers, memory can be allocated dynamically using malloc() function defined under "stdlib.h"header file.

Syntaxtouse malloc()

```
ptr=(cast-type*)malloc(byte-size)
```

Exampletousestructure'smemberthroughpointerusingmalloc()function.

```
#include<stdio.h>#
include<stdlib.h>s
truct name {
   int a;
   float b;
   charc[30];
};
intmain(){
   struct name *ptr;
   int i,n;
   printf("Entern:");
   scanf("%d",&n);
   ptr=(structname*)malloc(n*sizeof(structname));
/*Abovestatementallocatesthememoryfornstructureswithpointer ptr
pointing to base address */
   for(i=0;i<n;++i){
       printf("Enter    string,    integer    and    floating    number
respectively:\n");
       scanf("%s%d%f",&(ptr+i)->c,&(ptr+i)->a,&(ptr+i)->b);
   }
   printf("DisplayingInfromation:\n");
   for(i=0;i<n;++i)
       printf("%s\t%d\t%.2f\n",(ptr+i)->c,(ptr+i)->a,(ptr+i)->b);
   return 0;
}
```

Output

```
Entern:2
```

```
Enterstring,integerandfloatingnumberrespectively: Programming
2
3.2
Enterstring,integerandfloatingnumberrespectively: Structure
6
2.3
Displaying Information
Programming      2      3.20
Structure      6      2.30
```

CProgrammingStructureandFunction

InC, structurecan bepassed tofunctions bytwo methods:

1. Passingbyvalue(passingactualvalueasargument)
2. Passingbyreference(passingaddressofanargument)

Passingstructureby value

A structure variable can be passed to the function as an argument as normal variable. If structureis passedby value, changemadein structurevariablein function definition does not reflect in original structure variable in calling function.

Write a C program to create a structure student, containing name and roll. Ask user the name and roll of a student in main function. Pass this structure to a function and display the information in that function.

```
#include<stdio.h>
struct student{
    charname[50];
    int roll;
};
voidDisplay(structstudentstu);
/*functionprototypeshouldbebelowtothestructuredeclaration otherwise
compiler shows error */
intmain(){
    structstudents1;
    printf("Enterstudent'sname:");
    scanf("%s",&s1.name);
    printf("Enter roll number:");
    scanf("%d",&s1.roll);
    Display(s1);   //passingstructurevariables1asargument
    return 0;
}
void Display(struct student stu){
  printf("Output\nName:%s",stu.name);
  printf("\nRoll: %d",stu.roll);
}
```

Output

```
Enterstudent'sname:KevinAmla Enter
roll number: 149
Output
Name:KevinAmla
Roll: 149
```

Passingstructurebyreference

The address location of structure variable is passed to function while passing it by reference. If structure is passed by reference, change made in structure variable in function definition reflects in original structure variable in the calling function.

Write a C program to add two distances(feet-inch system) entered by user. To solve this program, make a structure. Pass two structure variable (containing distance in feet and inch) to add function by reference and display the result in main function without returning it.

```
#include<stdio.h>
struct distance{
    int feet;
    floatinch;
};
voidAdd(structdistanced1,structdistanced2,structdistance*d3); int
main()
{
    structdistancedist1,dist2,dist3;
    printf("First distance\n");
    printf("Enter feet: ");
    scanf("%d",&dist1.feet);
    printf("Enter inch: ");
    scanf("%f",&dist1.inch);
    printf("Second distance\n");
    printf("Enter feet: ");
    scanf("%d",&dist2.feet);
    printf("Enter inch: ");
    scanf("%f",&dist2.inch);
    Add(dist1,dist2,&dist3);

/*passing structure variables dist1 and dist2 by value whereas passing
structure variable dist3 by reference */
    printf("\nSumofdistances=%d\'-%.1f\"",dist3.feet,dist3.inch); return
    0;
}
voidAdd(structdistanced1,structdistanced2,structdistance*d3)
{
/*Addingdistancesd1andd2andstoringitind3*/ d3-
     >feet=d1.feet+d2.feet;
     d3->inch=d1.inch+d2.inch;
     if (d3->inch>=12) {        /*ifinchisgreaterorequalto12, converting
it to feet. */
         d3->inch-=12;
         ++d3->feet;
    }
```

```
}
```

Output

```
Firstdistance
Enterfeet:12
Enterinch:6.8
Seconddistance
Enter feet: 5
Enterinch:7.5

Sumofdistances=18'-2.3"
```

Explaination

In this program, structure variables *dist1* and *dist2* are passed by value (because value of *dist1* and *dist2* does not need to be displayed in main function) and *dist3* is passed by reference ,i.e, address of *dist3* `(&dist3)` is passed as an argument. Thus, the structure pointer variable *d3* points to the address of *dist3*. If any change is made in *d3* variable, effect of it is seed in *dist3* variable in main function.

CProgrammingUnions

Unions are quite similar to the structures in C. Union is also a derived type as structure. Union can be defined in same manner as structures just the keyword used in defining union in **union** where keyword used in defining structure was **struct.**

```
unioncar{
  charname[50];
  int price;
};
```

Unionvariablescanbe createdinsimilarmanner asstructure variable.

```
unioncar{
  charname[50];
  int price;
}c1,c2,*c3;

OR;

unioncar{
  charname[50];
```

```
  intprice;
};
----------InsideFunction ---------------
unioncarc1,c2, *c3;
```

In both cases, union variables *c1*, *c2* and union pointer variable *c3* of type **`union car`**is created.

Accessingmembersofan union

The member of unions can be accessed in similar manner as that structure. Suppose, we you want to access price for union variable *c1* in above example, it can be accessed as `c1.price`. If you want to access price for union pointer variable *c3*, it can be accessed as `(*c3).price`or as `c3->price`.

Differencebetweenunionandstructure

Though unions are similar to structure in so many ways, the difference between them is crucial to understand. This can be demonstrated by this example:

```
#include<stdio.h>
union job {         //definingaunion
   char name[32];
   float salary;
   intworker_no;
}u;
struct job1 {
   charname[32];
   float salary;
   intworker_no;
}s;
intmain(){
   printf("size of union = %d",sizeof(u));
   printf("\nsizeofstructure=%d",sizeof(s)); return
   0;
}
```

Output

```
size of union = 32
sizeofstructure=40
```

There is difference in memory allocation between union and structure as suggested in above example. The amount of memory required to store a structure variables is the sum of memory size of all members.

Fig: Memory allocation in case of structure

But, the memory required to store a union variable is the memory required for largest element of an union.

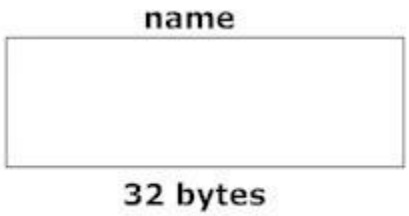


Fig: Memory allocation in case of union

Whatdifferencedoesitmakebetweenstructureandunion?

As you know, all members of structure can be accessed at any time. But, only one member of union can be accessed at a time in case of union and other members will contain garbage value.

```
#include<stdio.h>
union job {
   charname[32];
   float salary;
   intworker_no;
}u;
intmain(){
   printf("Enter name:\n");
   scanf("%s",&u.name);
   printf("Entersalary:\n");
   scanf("%f",&u.salary);
   printf("Displaying\nName:%s\n",u.name);
   printf("Salary: %.1f",u.salary);
   return0;
}
```

Output

```
Enter name
Hillary
Entersalary
1234.23
Displaying
Name: f%Bary
Salary:1234.2
```

Note: Youmay getdifferentgarbagevalueof name.

Why this output?

Initially,*Hillary* will be stored in `u.name`and other members of union will contain garbage value. But when user enters value of salary, 1234.23 will be stored in `u.salary` and other members will contain garbage value. Thus in output, salary is printedaccurately but, name displays some random string.

PassingUnionToa Function

Union can be passed in similar manner as structures in C programming. Visit this page to learn more about: How structure can be passed to function in C programming?

CProgrammingStructureExamples

This page contains examples and source code on structures in C programming language. To understand all examples in this page, you should have knowledge of following structure topics.

1. StructureIntroduction
2. StructureandPointers
3. PassingStructuretoFunction

ExampleofStructuresinCProgramming

CProgrammingExamplesandSourceCode

CProgramtoStoreInformation(name,rollandmarks)ofaStudentUsingStructure

In this program, a structure(student) is created which contains name, roll and marks as its data member. Then, a structure variable(s) is created. Then, data (name, roll and marks)is taken from user and stored in data members of structure variable `s`. Finally, the data entered by user is displayed.

CProgramtoStoreInformationofSingleVariable

```
#include<stdio.h>
struct student{
    charname[50];
    int roll;
    float marks;
};
intmain(){
    structstudents;
    printf("Enterinformationofstudents:\n\n");
    printf("Entername:");
```

```
    scanf("%s",s.name);
    printf("Enterrollnumber:");
    scanf("%d",&s.roll);
    printf("Enter marks: ");
    scanf("%f",&s.marks);
    printf("\nDisplayingInformation\n");
    printf("Name: %s\n",s.name);
    printf("Roll: %d\n",s.roll);
    printf("Marks: %.2f\n",s.marks);
    return 0;
}
```

Output

```
Enterinformationofstudents: Enter

name: Adele
Enterrollnumber:21 Enter
marks: 334.5

DisplayingInformation
name: Adele
Roll:21
Marks:334.50
```

CProgramtoAddTwoDistances(ininch-feet)SystemUsingStructures

Thisprogram takestwodistancesin inch-feetsystem and storesindata membersof twostructurevariables. Then, this program calculates the sum of two distances and displays it.

Sourcecodetoaddtwodistanceusingstructure

```
#include<stdio.h>
struct Distance{
    int feet;
    floatinch;
}d1,d2,sum;
intmain(){
    printf("Enterinformationfor1stdistance\n");
    printf("Enter feet: ");
    scanf("%d",&d1.feet);
    printf("Enterinch:");
    scanf("%f",&d1.inch);
    printf("\nEnterinformationfor2nddistance\n");
    printf("Enter feet: ");
    scanf("%d",&d2.feet);
    printf("Enterinch:");
    scanf("%f",&d2.inch);
```

```
    sum.feet=d1.feet+d2.feet;
    sum.inch=d1.inch+d2.inch;

/*Ifinchisgreaterthan12,changingittofeet.*/ if
    (sum.inch>12.0)
    {
        sum.inch=sum.inch-12.0;
        ++sum.feet;
    }
    printf("\nSumofdistances=%d\'-%.1f\"",sum.feet,sum.inch);
    return 0;
}
```

Output

```
Enterinformationfor1stdistance Enter
feet: 12
Enterinch:3.45

Enterinformationfor1stdistance Enter
feet: 12
Enterinch:9.2

Sumofdistances=25'-0.6"
```

Inthis program,a structure `Distance`is definedwith`inch`and`feet`as its members.Then,three variables(*d1*,*d2* and *sum*) of `struct Distance`type is created. Two variables(*d1* and *d2*) are used for taking distance from user and the sum of two distance is stored in variable *sum*and then, displayed.

CProgramtoAddTwoComplexNumbersbyPassingStructuretoaFunction

This program takes two distances in inch-feet system and stores in data members of two structure variables. Then, this program calculates the sum of two distances by passing itto a function and result is displayed in `main()` function.

SourceCodetoAddTwoComplexNumber

```
#include <stdio.h>
typedefstructcomplex{
    floatreal;
    floatimag;
}complex;
complexadd(complexn1,complexn2); int
main(){
    complexn1,n2,temp;
    printf("For1stcomplexnumber\n");
    printf("Enterrealandimaginaryrespectively:\n");
    scanf("%f%f",&n1.real,&n1.imag);
    printf("\nFor2ndcomplexnumber\n");
    printf("Enterrealandimaginaryrespectively:\n");
```

```
    scanf("%f%f",&n2.real,&n2.imag);
    temp=add(n1,n2);
    printf("Sum=%.1f+%.1fi",temp.real,temp.imag);
    return 0;
}
complexadd(complexn1,complexn2){ complex
      temp; temp.real=n1.real+n2.real;
      temp.imag=n1.imag+n2.imag;
      return(temp);
}
```

Output

```
For1stcomplexnumber
Enterrealandimaginaryrespectively:2.3 4.5

For1stcomplexnumber
Enterrealandimaginaryrespectively:3.4 5
Sum=5.7+9.5i
```

In this program structures *n1* and *n2* are passed as an argument of function `add()`. Thisfunctioncomputesthesumandreturnsthestructurevariable*temp*tothe`main()` function.

CProgramtoCalculateDifferenceBetweenTwoTimePeriod

In this program, user is asked to enter two time periods and these two periods are storedin structure variables. This program caculates the difference between these two time period. To perform this task, a function is created which calculates the difference and the result is displayed in `main()` function without returning it (Using call by reference technique).

CProgramtoCalculateDifferenceBetweenTwoTimePeriod

```
#include<stdio.h>
struct TIME{
  intseconds;
  intminutes;
  int hours;
};
voidDifference(structTIMEt1,structTIMEt2,structTIME*diff); int
main(){
    struct TIME t1,t2,diff;
    printf("Enterstarttime:\n");
    printf("Enterhours,minutesandsecondsrespectively:");
    scanf("%d%d%d",&t1.hours,&t1.minutes,&t1.seconds);
    printf("Enter stop time: \n");
    printf("Enterhours,minutesandsecondsrespectively:");
```

```
    scanf("%d%d%d",&t2.hours,&t2.minutes,&t2.seconds);
    Difference(t1,t2,&diff);
    printf("\nTIME            DIFFERENCE:            %d:%d:%d            -
",t1.hours,t1.minutes,t1.seconds);
    printf("%d:%d:%d ",t2.hours,t2.minutes,t2.seconds);
    printf("=%d:%d:%d\n",diff.hours,diff.minutes,diff.seconds);
    return 0;
}
voidDifference(structTIMEt1,structTIMEt2,structTIME*differ){
    if(t2.seconds>t1.seconds){
        --t1.minutes;
        t1.seconds+=60;
    }
    differ->seconds=t1.seconds-t2.seconds;
    if(t2.minutes>t1.minutes){
        --t1.hours;
        t1.minutes+=60;
    }
    differ->minutes=t1.minutes-t2.minutes;
    differ->hours=t1.hours-t2.hours;
}
```

Output

```
Enterstarttime:
Enterhours,minutesandsecondsrespectively:12 34
55
Enterstoptime:
Enterhours,minutesandsecondsrespectively:8 12
15

TIMEDIFFERENCE:12:34:55-8:12:15=4:22:40
```

CProgramtoStoreInformationof10StudentsUsingStructure

In this program, a structure(student) is created which contains name, roll and marks as its data member. Then, an array of structure of 10 elements is created. Then, data(name, roll and marks) for 10 elements is asked to user and stored in array of structure. Finally, the data entered by user is displayed.

SourceCodetoStoreInformationof10studentsUsingStructure

```
#include<stdio.h>
struct student{
    charname[50];
    int roll;
    float marks;
};
intmain(){
    structstudents[10];
    inti;
```

```
    printf("Enterinformationofstudents:\n");
    for(i=0;i<10;++i)
    {
        s[i].roll=i+1;
        printf("\nForrollnumber%d\n",s[i].roll);
        printf("Enter name: ");
        scanf("%s",s[i].name);
        printf("Entermarks:");
        scanf("%f",&s[i].marks);
        printf("\n");
    }
    printf("Displayinginformationofstudents:\n\n");
    for(i=0;i<10;++i)
    {
     printf("\nInformationforrollnumber%d:\n",i+1);
     printf("Name: ");
     puts(s[i].name);
     printf("Marks:%.1f",s[i].marks);
   }
   return0;
}
```

Output

```
Enterinformationofstudents: For

roll number 1
Entername:Tom
Entermarks:98

Forrollnumber2
Entername:Jerry
Enter marks: 89
.
.
.
Displayinginformationofstudents:

Informationforrollnumber1: Name:
Tom
Marks:98
.
.
.
```

CProgramtoStoreInformationUsingStructuresfornElementsDynamically

Thisprogramasksusertostorethevalueof*n*andallocatesthememoryforthe*n* structurevariabledynamicallyusing malloc()function.

SourceCodeDemonstratetheDynamicMemoryAllocationforStructure

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

CProgrammingExamplesandSourceCode

```
#include<stdlib.h>s
truct name {
   inta;
   charc[30];
};
intmain(){
   struct name *ptr;
   int i,n;
   printf("Entern:");
   scanf("%d",&n);

/*Allocatesthememoryfornstructureswithpointerptr pointingto the base
address. */
   ptr=(structname*)malloc(n*sizeof(structname));
   for(i=0;i<n;++i){
       printf("Enterstringandintegerrespectively:\n");
       scanf("%s%d",&(ptr+i)->c, &(ptr+i)->a);
   }
   printf("DisplayingInfromation:\n");
   for(i=0;i<n;++i)
       printf("%s\t%d\t\n",(ptr+i)->c,(ptr+i)->a);
   return 0;
}
```

Output

```
Entern:2
Enterstringandintegerrespectively:
Programming
22
Enterstring,integerandfloatingnumberrespectively: Structure
33

DisplayingInformation:
Programming      22
Structure        33
```

CProgrammingFilesI/O

InCprogramming, fileisa placeon diskwhereagroupofrelated datais stored.

Whyfilesareneeded?

When the program is terminated, the entire data is lost in C programming. If you want to keep large volume of data, it is time consuming to enter the entire data. But, if file is created, these information can be accessed using few commands.

Therearelargenumbers offunctionstohandlefileI/OinClanguage. Inthistutorial,you will learn to handle standard I/O(High level file I/O functions) in C.

HighlevelfileI/Ofunctionscanbecategorizedas:

1. Textfile
2. Binaryfile

FileOperations

1. Creatinganew file
2. Openinganexistingfile
3. Readingfromandwritinginformationtoafile
4. Closingafile

Workingwithfile

Whileworkingwithfile,youneedtodeclareapointeroftypefile.Thisdeclarationis needed for communication between file and program.

```
FILE*ptr;
```

Openingafile

Opening a file is performed using library function fopen(). The syntax for opening a file in standard I/O is:

```
ptr=fopen("fileopen","mode")

For Example:
fopen("E:\\cprogram\program.txt","w");

/* ---------------------------------------------------------------------------- */
 E:\\cprogram\program.txtisthelocationtocreatefile. "w"
 represents the mode for writing.
/* ---------------------------------------------------------------------------- */
```

Here,theprogram.txtfile isopenedforwriting mode.

OpeningModesinStandardI/O		
FileMode	**MeaningofMode**	**DuringInexistenceof file**
r	Openfor reading.	If the filedoes not exist, fopen() returns NULL.
w	Openfor writing.	Ifthe file exists, its contents are overwritten. If the file does not exist, it will be created.
a	Open for append. i.e, Data is added to end of file.	If the file does not exists, it will be created.

OpeningModesinStandardI/O		
FileMode	**MeaningofMode**	**DuringInexistenceof file**
r+	Open for both reading and writing.	If the filedoes not exist, fopen() returns NULL.
w+	Open for both reading and writing.	Ifthefileexists,itscontentsare overwritten.Ifthefiledoesnot exist, it will be created.
a+	Open for both reading and appending.	If the file does not exists, it will be created.

ClosingaFile

The file should be closed after reading/writing of a file. Closing a file is performed using library function fclose().

```
fclose(ptr);//ptristhefilepointerassociatedwithfiletobe closed.
```

TheFunctionsfprintf()andfscanf()functions.

Thefunctions`fprintf()`and`fscanf()`are the file version of `printf()`and `fscanf()`. Theonlydifference while using `fprintf()`and `fscanf()`is that, the first argument is a pointer to the structure FILE

Writingtoafile

```
#include<stdio.h>
int main()
{
   intn;
   FILE *fptr;
   fptr=fopen("C:\\program.txt","w");
   if(fptr==NULL){
      printf("Error!");
      exit(1);
   }
   printf("Entern:");
   scanf("%d",&n);
   fprintf(fptr,"%d",n);
   fclose(fptr);
   return0;
}
```

This program takes the number from user and stores in file. 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 that file, you can see the integer you entered.

Similarly,`fscanf()`canbeusedtoreaddatafromfile.

Readingfromfile

```
#include<stdio.h>
int main()
{
   intn;
   FILE*fptr;
   if((fptr=fopen("C:\\program.txt","r"))==NULL){
       printf("Error! opening file");
       exit(1);          /*Programexitsiffilepointerreturns NULL.
*/
   }
   fscanf(fptr,"%d",&n);
   printf("Valueofn=%d",n);
   fclose(fptr);
   return0;
}
```

If you have run program above to write in file successfully, you can get the integer back entered in that program using this program.

Otherfunctionslike`fgetchar()`,`fputc()`etc.canbeusedinsimilar way.

BinaryFiles

Dependinguponthewayfileisopenedforprocessing,afileisclassifiedintotextfileand binary file.

If a large amount of numerical data it to be stored, text mode will be insufficient. In such case binary file is used.

Workingofbinaryfilesissimilartotextfileswithfewdifferencesinopeningmodes, reading from file and writing to file.

Openingmodesofbinaryfiles

Opening modes of binary files are `rb`, `rb+`, `wb`, `wb+`,`ab`and `ab+`. The only difference between opening modes oftext and binaryfiles is that, b is appended to indicatethat, it is binary file.

Readingandwritingofabinaryfile.

Functions `fread()`and `fwrite()`are used for reading from and writing to a file on the disk respectively in case of binary files.

Function fwrite() takes four arguments, address of data to be written in disk, size of data to be written in disk, number of such type of data and pointer to the file where you want to write.

```
fwrite(address_data,size_data,numbers_data,pointer_to_file);
```

Function`fread()`alsotake4argumentssimilarto`fwrite()`functionas above.

CProgrammingFileExamples

ExamplesoffilesinCProgramming
CProgramtoreadnameandmarksofstudentsandstoreitinfile
C Program to read name and marks of students and store it in file. If filealready exists, add information to it.
CProgramtowritemembersofarraystoafileusingfwrite()

CProgramtoreadnameandmarksofstudentsandstoreitinfile

Write a C program to read name and marks of n number of students from user and store them in a file

```
#include<stdio.h>
int main(){
   charname[50];
   intmarks,i,n;
   printf("Enternumberofstudents:");
   scanf("%d",&n);
   FILE *fptr;
   fptr=(fopen("C:\\student.txt","w"));
   if(fptr==NULL){
      printf("Error!");
      exit(1);
   }
   for(i=0;i<n;++i)
   {
      printf("Forstudent%d\nEntername:",i+1);
      scanf("%s",name);
      printf("Entermarks:");
      scanf("%d",&marks);
      fprintf(fptr,"\nName:%s\nMarks=%d\n",name,marks);
   }
   fclose(fptr);
   return 0;
}
```

C Program to read name and marks of students and store it in file. If file already exists, add information toit.

Write a C program to read name and marks of n number of students from user and store them in a file. If the file previously exits, add the information of n students.

```
#include<stdio.h>
int main(){
   charname[50];
   intmarks,i,n;
   printf("Enternumberofstudents:");
   scanf("%d",&n);
   FILE *fptr;
   fptr=(fopen("C:\\student.txt","a"));
   if(fptr==NULL){
       printf("Error!");
       exit(1);
   }
   for(i=0;i<n;++i)
   {
      printf("Forstudent%d\nEntername:",i+1);
      scanf("%s",name);
      printf("Entermarks:");
      scanf("%d",&marks);
      fprintf(fptr,"\nName:%s\nMarks=%d\n",name,marks);
   }
   fclose(fptr);
   return 0;
}
```

CProgramtowritemembersofarraystoafileusingfwrite()

Write a C programto write all the membersof anarray of strcuresto a file usingfwrite(). Read the array from the file and display on the screen.

```
#include<stdio.h>
struct s
{
charname[50];
int height;
};
intmain(){
    structsa[5],b[5];
    FILE *fptr;
    int i;
    fptr=fopen("file.txt","wb");
    for(i=0;i<5;++i)
    {
        fflush(stdin);
        printf("Entername:");
```

```
        gets(a[i].name);
        printf("Enterheight:");
        scanf("%d",&a[i].height);
    }
    fwrite(a,sizeof(a),1,fptr);
    fclose(fptr);
    fptr=fopen("file.txt","rb");
    fread(b,sizeof(b),1,fptr);
    for(i=0;i<5;++i)
    {
        printf("Name:%s\nHeight:%d",b[i].name,b[i].height);
    }
    fclose(fptr);
}
```

CProgramtoStoreInformation(name,rollandmarks)ofaStudentUsingStructure

Tounderstandthisexample,youshouldhaveknowledgeoffollowing C programming topics:

CProgrammingStructure

In this program, a structure(student) is created which contains name, roll and marks as its data member. Then, a structure variable(s) is created. Then, data (name, roll and marks)is taken from user and stored in data members of structure variable `s`. Finally, the data entered by user is displayed.

CProgramtoStoreInformationofSingleVariable

```
#include<stdio.h>
struct student{
    charname[50];
    int roll;
    float marks;
};
intmain(){
    structstudents;
    printf("Enterinformationofstudents:\n\n");
    printf("Enter name: ");
    scanf("%s",s.name);
    printf("Enterrollnumber:");
    scanf("%d",&s.roll);
    printf("Enter marks: ");
    scanf("%f",&s.marks);
    printf("\nDisplayingInformation\n");
    printf("Name: %s\n",s.name);
    printf("Roll: %d\n",s.roll);
    printf("Marks: %.2f\n",s.marks);
    return 0;
}
```

Output

```
Enterinformationofstudents:
```

```
Enter name: Adele
Enterrollnumber:21
Enter marks: 334.5

DisplayingInformation
name: Adele
Roll:21
Marks:334.50
```

CProgramtoAddTwoDistances(ininch-feet)SystemUsingStructures

Tounderstandthisexample,youshouldhaveknowledgeoffollowing C programming topics:

CProgrammingStructure

This program takes two distances in inch-feet system and stores in data members of two structure variables. Then, this program calculates the sum of two distances and displaysit.

Sourcecodetoaddtwodistanceusingstructure

```
#include<stdio.h>
struct Distance{
    int feet;
    floatinch;
}d1,d2,sum;
intmain(){
    printf("Enterinformationfor1stdistance\n");
    printf("Enter feet: ");
    scanf("%d",&d1.feet);
    printf("Enterinch:");
    scanf("%f",&d1.inch);
    printf("\nEnterinformationfor2nddistance\n"); printf("Enter
    feet: ");
    scanf("%d",&d2.feet);
    printf("Enter inch: ");
    scanf("%f",&d2.inch);
    sum.feet=d1.feet+d2.feet;
    sum.inch=d1.inch+d2.inch;

/*Ifinchisgreaterthan12,changingittofeet.*/ if
    (sum.inch>12.0)
    {
        sum.inch=sum.inch-12.0;
        ++sum.feet;
    }
    printf("\nSumofdistances=%d\'-%.1f\"",sum.feet,sum.inch);
    return 0;
}
```

Output

```
Enterinformationfor1stdistance Enter
feet: 12
Enterinch:3.45

Enterinformationfor1stdistance Enter
feet: 12
Enterinch:9.2

Sumofdistances=25'-0.6"
```

In this program, a structure`Distance`is defined with `inch`and `feet`as its members. Then, three variables(*d1*, *d2* and *sum*) of `struct Distance` type is created. Two variables(*d1*and*d2*)areusedfortakingdistancefromuserandthesumof twodistanceis stored in variable *sum* and then, displayed.

CProgramtoAddTwoComplexNumbersbyPassingStructuretoaFunction

Tounderstandthisexample,youshouldhaveknowledgeoffollowing C programming topics:

CProgrammingStructure
CProgrammingStructureandFunction

This program takes two distances in inch-feet system and stores in data members of two structure variables. Then, this program calculates the sum of two distances by passing itto a function and result is displayed in `main()`function.

SourceCodetoAddTwoComplexNumber

```
#include <stdio.h>
typedefstructcomplex{
    floatreal;
    floatimag;
}complex;
complexadd(complexn1,complexn2); int
main(){
    complexn1,n2,temp;
    printf("For1stcomplexnumber\n");
    printf("Enterrealandimaginaryrespectively:\n");
    scanf("%f%f",&n1.real,&n1.imag);
    printf("\nFor 2nd complex number \n");
    printf("Enterrealandimaginaryrespectively:\n");
    scanf("%f%f",&n2.real,&n2.imag);
    temp=add(n1,n2);
    printf("Sum=%.1f+%.1fi",temp.real,temp.imag);
    return 0;
}
complexadd(complexn1,complexn2){
      complex temp;
      temp.real=n1.real+n2.real;
      temp.imag=n1.imag+n2.imag;
```

```
      return(temp);
}
```

Output

```
For1stcomplexnumber
Enterrealandimaginaryrespectively:2.3 4.5

For1stcomplexnumber
Enterrealandimaginaryrespectively:3.4 5
Sum=5.7+9.5i
```

In this program structures *n1* and *n2* are passed as an argument of function `add()`. Thisfunctioncomputesthesumandreturnsthestructurevariable*temp*tothe`main()`function.

CProgramtoCalculateDifferenceBetweenTwoTimePeriod

Tounderstandthisexample,youshouldhaveknowledgeoffollowingCprogramming topics:

CProgrammingStructure
CProgrammingStructureandFunctionC
Programming Structure and Pointer

In this program, user is asked to enter two time periods and these two periods are storedin structure variables. This program caculates the difference between these two time period. To perform this task, a function is created which calculates the difference and the result is displayed in `main()` function without returning it (Using call by reference technique).

CProgramtoCalculateDifferenceBetweenTwoTimePeriod

```
#include<stdio.h>
struct TIME{
  intseconds;
  intminutes;
  int hours;
};
voidDifference(structTIMEt1,structTIMEt2,structTIME*diff); int
main(){
    struct TIME t1,t2,diff;
    printf("Enterstarttime:\n");
    printf("Enterhours,minutesandsecondsrespectively:");
    scanf("%d%d%d",&t1.hours,&t1.minutes,&t1.seconds);
    printf("Enter stop time: \n");
    printf("Enterhours,minutesandsecondsrespectively:");
    scanf("%d%d%d",&t2.hours,&t2.minutes,&t2.seconds);
    Difference(t1,t2,&diff);
    printf("\nTIME            DIFFERENCE:            %d:%d:%d            -
",t1.hours,t1.minutes,t1.seconds);
    printf("%d:%d:%d",t2.hours,t2.minutes,t2.seconds);
```

```
    printf("=%d:%d:%d\n",diff.hours,diff.minutes,diff.seconds);
    return 0;
}
voidDifference(structTIMEt1,structTIMEt2,structTIME*differ){
    if(t2.seconds>t1.seconds){
        --t1.minutes;
        t1.seconds+=60;
    }
    differ->seconds=t1.seconds-t2.seconds;
    if(t2.minutes>t1.minutes){
        --t1.hours;
        t1.minutes+=60;
    }
    differ->minutes=t1.minutes-t2.minutes;
    differ->hours=t1.hours-t2.hours;
}
```

Output

```
Enterstarttime:
Enterhours,minutesandsecondsrespectively:12 34
55
Enterstoptime:
Enterhours,minutesandsecondsrespectively:8 12
15

TIMEDIFFERENCE:12:34:55-8:12:15=4:22:40
```

CProgramtoStoreInformationofStudentsUsingStructure

Tounderstandthisexample,youshouldhaveknowledgeoffollowing C programming topics:

C Programming ArraysCProgrammingStructure

In this program, a structure(student) is created which contains name, roll and marks as its data member. Then, an array of structure of 10 elements is created. Then, data(name, roll and marks) for 10 elements is asked to user and stored in array of structure. Finally, the data entered by user is displayed.

SourceCodetoStoreInformationof10studentsUsingStructure

```
#include<stdio.h>
struct student{
    charname[50];
    int roll;
    float marks;
};
intmain(){
    structstudents[10];
    int i;
```

```
    printf("Enterinformationofstudents:\n");
    for(i=0;i<10;++i)
    {
        s[i].roll=i+1;
        printf("\nForrollnumber%d\n",s[i].roll);
        printf("Enter name: ");
        scanf("%s",s[i].name);
        printf("Entermarks:");
        scanf("%f",&s[i].marks);
        printf("\n");
    }
    printf("Displayinginformationofstudents:\n\n");
    for(i=0;i<10;++i)
    {
     printf("\nInformationforrollnumber%d:\n",i+1); printf("Name:
     ");
     puts(s[i].name);
     printf("Marks:%.1f",s[i].marks);
   }
   return0;
}
```

Output

```
Enterinformationofstudents:

Forrollnumber1
Enter name: Tom
Enter marks: 98

Forrollnumber2
Entername:Jerry
Enter marks: 89
.
.
.
Displayinginformationofstudents:

Informationforrollnumber1:
Name:Tom
Marks:98
.
.
.
```

CProgramtoStoreInformationUsingStructureswithDynamicallyMemoryAllocation

Tounderstandthisexample,youshouldhaveknowledgeoffollowing C programming topics:

CProgrammingPointers
CProgrammingDynamicMemoryAllocationC
Programming Structure

Thisprogramasksusertostorethevalueof*n*andallocatesthememoryforthe*n* structurevariabledynamicallyusing<u>malloc()function</u>.

<u>SourceCodeDemonstratetheDynamicMemoryAllocationforStructure</u>

```
#include<stdio.h>#
include<stdlib.h>s
truct name {
   inta;
   charc[30];
};
intmain(){
   struct name *ptr;
   int i,n;
   printf("Entern:");
   scanf("%d",&n);

/*Allocatesthememoryfornstructureswithpointerptr pointingto the base
address. */
   ptr=(structname*)malloc(n*sizeof(structname));
   for(i=0;i<n;++i){
       printf("Enterstringandintegerrespectively:\n");
       scanf("%s%d",&(ptr+i)->c, &(ptr+i)->a);
   }
   printf("DisplayingInfromation:\n");
   for(i=0;i<n;++i)
       printf("%s\t%d\t\n",(ptr+i)->c,(ptr+i)->a);
   return 0;
}
```

Output

```
Entern:2
Enterstringandintegerrespectively:
Programming
22
Enterstring,integerandfloatingnumberrespectively:
Structure
33

DisplayingInformation:
Programming       22
Structure         33
```