**Model Question paper of Programming in C and Cpp**

**MCA Semester I**

1. **Explain all data types in C?**

Data types specify how we enter data into our programs and what type of data we enter. C language has some predefined set of data types to handle various kinds of data that we can use in our program. These data types have different storage capacities.

C language supports 2 different type of data types:

Primary data types:These are fundamental data types in C namely integer(int), floating point(float), character(char) and void.

Derived data types:Derived data types are nothing but primary datatypes but a little twisted or grouped together like array, stucture, union and pointer. These are discussed in details later.

Data type determines the type of data a variable will hold. If a variable x is declared as int. it means x can hold only integer values. Every variable which is used in the program must be declared as what data-type it is.



**Integer type**

Integers are used to store whole numbers.

**Size and range of Integer type on 16-bit machine:**

|  |  |  |
| --- | --- | --- |
| **Type** | **Size(bytes)** | **Range** |
| int or signed int | 2 | -32,768 to 32767 |
| unsigned int | 2 | 0 to 65535 |
| short int or signed short int | 1 | -128 to 127 |
| unsigned short int | 1 | 0 to 255 |
| long int or signed long int | 4 | -2,147,483,648 to 2,147,483,647 |
| unsigned long int | 4 | 0 to 4,294,967,295 |

**Floating point type**

Floating types are used to store real numbers.

**Size and range of Integer type on 16-bit machine**

|  |  |  |
| --- | --- | --- |
| **Type** | **Size(bytes)** | **Range** |
| Float | 4 | 3.4E-38 to 3.4E+38 |
| Double | 8 | 1.7E-308 to 1.7E+308 |
| long double | 10 | 3.4E-4932 to 1.1E+4932 |

**Character type**

Character types are used to store characters value.

**Size and range of Integer type on 16-bit machine**

|  |  |  |
| --- | --- | --- |
| **Type** | **Size(bytes)** | **Range** |
| char or signed char | 1 | -128 to 127 |
| unsigned char | 1 | 0 to 255 |

**void type**

void type means no value. This is usually used to specify the type of functions which returns nothing. We will get acquainted to this data type as we start learning more advanced topics in C language, like functions, pointers etc.

1. **Define functions? What is function Overloading?**

Function refers to a segment that groups code to perform a specific task.

Two or more functions having same name but different argument(s) are known as overloaded functions.

In C++ programming, two functions can have same name if number and/or type of arguments passed are different.

These functions having different number or type (or both) of parameters are known as overloaded functions. For example:

int test() { }

int test(int a) { }

float test(double a) { }

int test(int a, double b) { }

Here, all 4 functions are overloaded functions because argument(s) passed to these functions are different.

The return type of all these 4 functions are not same. Overloaded functions may or may not have different return type but it should have different argument(s).

1. **Differentiate between structures and unions?**



1. **Explain File Handling in C?**

A file represents a sequence of bytes on the disk where a group of related data is stored. File is created for permanent storage of data. It is a readymade structure.

In C language, we use a structure pointer of file type to declare a file.

 FILE \*fp;

C provides a number of functions that helps to perform basic file operations. Following are the functions,

|  |  |
| --- | --- |
| **Function** | **Description** |
| fopen() | create a new file or open a existing file |
| fclose() | closes a file |
| getc() | reads a character from a file |
| putc() | writes a character to a file |
| fscanf() | reads a set of data from a file |
| fprintf() | writes a set of data to a file |
| getw() | reads a integer from a file |
| putw() | writes a integer to a file |
| fseek() | set the position to desire point |
| ftell() | gives current position in the file |
| rewind() | set the position to the beginning point |

1. **Define Inheritance? Mention types of Inheritance?**

In [object-oriented programming](http://searchsoa.techtarget.com/definition/object-oriented-programming), inheritance is the concept that when a [class](http://whatis.techtarget.com/definition/class) of [object](http://searchsoa.techtarget.com/definition/object)s is defined, any subclass that is defined can inherit the definitions of one or more general classes. This means for the programmer that an object in a subclass need not carry its own definition of data and methods that are generic to the class (or classes) of which it is a part. This not only speeds up program development; it also ensures an inherent validity to the defined subclass object (what works and is consistent about the class will also work for the subclass).

Types of Inheritance

In C++, we have 5 different types of Inheritance. Namely,

1. Single Inheritance
2. Multiple Inheritance
3. Hierarchical Inheritance
4. Multilevel Inheritance
5. Hybrid Inheritance (also known as Virtual Inheritance)
6. **Explain constructors?**

Constructors are special class functions which performs initialization of every object. The Compiler calls the Constructor whenever an object is created. Constructors intialize values to object members after storage is allocated to the object.

class A

{

 int x;

 public:

 A(); //Constructor

};

While defining a constructor you must remember that the name of constructor will be same as the name of the class, and constructors never have return type.

Constructors can be defined either inside the class definition or outside class definition using class name and scope resolution :: operator.

class A

{

 int i;

 public:

 A(); //Constructor declared

};

A::A() // Constructor definition

{

 i=1;

}

#### Types of Constructors

Constructors are of three types :

1. Default Constructor
2. Parameterized Constructor
3. Copy Constructor

#### Default Constructor

Default constructor is the constructor which doesn't take any argument. It has no parameter.

Syntax :

class\_name ()

{ Constructor Definition }

#### Parameterized Constructor

These are the constructors with parameter. Using this Constructor you can provide different values to data members of different objects, by passing the appropriate values as argument.

#### Copy Constructor

These are special type of Constructors which takes an object as argument, and is used to copy values of data members of one object into other object.

1. **What are virtual functions?**

A virtual function a member function which is declared within base class and is re-defined (Overriden) by derived class. When you refer to a derived class object using a pointer or a reference to the base class, you can call a virtual function for that object and execute the derived class’s version of the function.

* Virtual functions ensure that the correct function is called for an object, regardless of the type of reference (or pointer) used for function call.
* They are mainly used to achieve[Runtime polymorphism](https://www.geeksforgeeks.org/polymorphism-in-c/)
* Functions are declared with a virtual keyword in base class.
* The resolving of function call is done at Run-time.

Rules for Virtual Functions

1. They Must be declared in public section of class.
2. Virtual functions cannot be static and also cannot be a friend function of another class.
3. Virtual functions should be accessed using pointer or reference of base class type to achieve run time polymorphism.
4. The prototype of virtual functions should be same in base as well as derived class.
5. They are always defined in base class and overridden in derived class. It is not mandatory for derived class to override (or re-define the virtual function), in that case base class version of function is used.
6. A class may have [virtual destructor](https://www.geeksforgeeks.org/virtual-destructor/) but it cannot have a virtual constructor.
7. **Explain Friend Function?**

Friend functions are those functions which can access all the functions and  variables of a class though it is not a member function of that class. Actually to share a function among two or more classes friend functions are used. If it is declared so, then it will able to access all variables and functions of those classes.

Some uses of the friend function are

1. When certain operator overloading is required friend functions can be useful.
2. Friend function makes the creation of some types of I/O functions easier.
3. Sometimes two or more classes may contain interrelated members which may need to be operated at a time. In such times a friend function is required.

1. **Explain Exception Handling in Cpp?**

An exception is a problem that arises during the execution of a program. A C++ exception is a response to an exceptional circumstance that arises while a program is running, such as an attempt to divide by zero.

Exceptions provide a way to transfer control from one part of a program to another. C++ exception handling is built upon three keywords: try, catch,and throw.

* **throw** − A program throws an exception when a problem shows up. This is done using a **throw** keyword.
* **catch** − A program catches an exception with an exception handler at the place in a program where you want to handle the problem. The **catch** keyword indicates the catching of an exception.
* **try** − A **try** block identifies a block of code for which particular exceptions will be activated. It's followed by one or more catch blocks.

Assuming a block will raise an exception, a method catches an exception using a combination of the try and catch keywords. A try/catch block is placed around the code that might generate an exception. Code within a try/catch block is referred to as protected code, and the syntax for using try/catch as follows −

try {

 // protected code

} catch( ExceptionName e1 ) {

 // catch block

} catch( ExceptionName e2 ) {

 // catch block

} catch( ExceptionName eN ) {

 // catch block

}

1. **Write a program to matrix addition?**

#include <stdio.h>

#include<conio.h>

int main()

{

 int m, n, c, d, first[10][10], second[10][10], sum[10][10];

 printf("Enter the number of rows and columns of matrix\n");

 scanf("%d%d", &m, &n);

 printf("Enter the elements of first matrix\n");

  for (c = 0; c < m; c++)

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

 scanf("%d", &first[c][d]);

 printf("Enter the elements of second matrix\n");

 for (c = 0; c < m; c++)

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

 scanf("%d", &second[c][d]);

 printf("Sum of entered matrices:-\n");

 for (c = 0; c < m; c++) {

 for (d = 0 ; d < n; d++) {

 sum[c][d] = first[c][d] + second[c][d];

 printf("%d\t", sum[c][d]);

 }

 printf("\n");

 }

 return 0;

}

**Prepared By**

**Kavita Lal**

**Guest Faculty**

**MCA Dept**