Wᜉᜋn=tsN= Es=tdo nN= Is=l nN bohol=

**Bohol Island State University**

**mlkeN= pAdln= s Ag=sikpn= At= Ad=kitek=tod**

**College of Engineering and Architecture**

PRASSIGNMENT

**C – PROGRAMMING**

FEBRUARY 20, 2019

CPRRTRFHSCCCPPRC

|  |
| --- |
| **CLIFFORD JOHN C. TAMPOS** |
| **baapto 1**  **BSCpE 1-B** |

|  |
| --- |
| **Ag=sekp= m.A. peden=**  **ENGR. M.A. PERIN** |
| **Tgpg=todoo**  **INSTRUCTOR** |

**LIBRARY OF “#include<iostream>” through DEVC++**

// Standard iostream objects -\*- C++ -\*-

// Copyright (C) 1997-2014 Free Software Foundation, Inc.

// This file is part of the GNU ISO C++ Library. This library is free

// software; you can redistribute it and/or modify it under the

// terms of the GNU General Public License as published by the

// Free Software Foundation; either version 3, or (at your option)

// any later version.

// This library is distributed in the hope that it will be useful,

// but WITHOUT ANY WARRANTY; without even the implied warranty of

// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

// GNU General Public License for more details.

// Under Section 7 of GPL version 3, you are granted additional

// permissions described in the GCC Runtime Library Exception, version

// 3.1, as published by the Free Software Foundation.

// You should have received a copy of the GNU General Public License and

// a copy of the GCC Runtime Library Exception along with this program;

// see the files COPYING3 and COPYING.RUNTIME respectively. If not, see

// <http://www.gnu.org/licenses/>.

/\*\* @file include/iostream

\* This is a Standard C++ Library header.

\*/

//

// ISO C++ 14882: 27.3 Standard iostream objects

//

#ifndef \_GLIBCXX\_IOSTREAM

#define \_GLIBCXX\_IOSTREAM 1

#pragma GCC system\_header

#include <bits/c++config.h>

#include <ostream>

#include <istream>

namespace std \_GLIBCXX\_VISIBILITY(default) //mao ni ang default nga dapat i type before sa int main kay para dli na mag balik ug type sa std

{

\_GLIBCXX\_BEGIN\_NAMESPACE\_VERSION

/\*\*

\* @name Standard Stream Objects

\*

\* The &lt;iostream&gt; header declares the eight <em>standard stream

\* objects</em>. For other declarations, see

\* http://gcc.gnu.org/onlinedocs/libstdc++/manual/io.html

\* and the @link iosfwd I/O forward declarations @endlink

\*

\* They are required by default to cooperate with the global C

\* library's @c FILE streams, and to be available during program

\* startup and termination. For more information, see the section of the

\* manual linked to above.

\*/

//@{

extern istream cin; /// Linked to standard input

extern ostream cout; /// Linked to standard output

extern ostream cerr; /// Linked to standard error (unbuffered)

extern ostream clog; /// Linked to standard error (buffered)

#ifdef \_GLIBCXX\_USE\_WCHAR\_T

extern wistream wcin; /// Linked to standard input

extern wostream wcout; /// Linked to standard output

extern wostream wcerr; /// Linked to standard error (unbuffered)

extern wostream wclog; /// Linked to standard error (buffered)

#endif

//@}

// For construction of filebuffers for cout, cin, cerr, clog et. al.

static ios\_base::Init \_\_ioinit;

\_GLIBCXX\_END\_NAMESPACE\_VERSION

} // namespace

#endif /\* \_GLIBCXX\_IOSTREAM \*/

**<iostream>**

* Header that defines the standard input/output stream objects:
* Including this header may automatically include other headers, such as [<ios>](http://www.cplusplus.com/%3Cios%3E), [<streambuf>](http://www.cplusplus.com/%3Cstreambuf%3E), [<istream>](http://www.cplusplus.com/%3Cistream%3E), [<ostream>](http://www.cplusplus.com/%3Costream%3E) and/or [<iosfwd>](http://www.cplusplus.com/%3Ciosfwd%3E).

In order to read or write to the standard input/output streams you need to include it.

**Sample program**

|  |
| --- |
| int main( int argc, char \* argv[] )  {  std::cout << "Hello World!" << std::endl;  return 0;  } |

**Sample output**

Hello World!

That program will not compile unless you add #include <iostream>

The second line isn't necessary

**using namespace std;**

What that does is tell the compiler that symbol names defined in the std namespace are to be brought into your program's scope, so you can omit the namespace qualifier, and write for example

**Sample program**

|  |
| --- |
| #include <iostream>  using namespace std;  int main( int argc, char \* argv[] )  {  cout << "Hello World!" << endl;  return 0;  } |

**Sample output**

Hello World!

Notice you no longer need to refer to the output stream with the fully qualified name std::cout and can use the shorter name cout.

* This file is part of the GNU ISO C++ Library. This library is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3, or (at your option) any later version.
* This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
* This is a Standard C++ Library header.
* The &lt;iostream&gt; header declares the eight <em>standard stream objects</em>. They are required by default to cooperate with the global C library's @c FILE streams, and to be available during program startup and termination.
* For construction of filebuffers for cout, cin, cerr, clog et. al. static ios\_base::Init \_\_ioinit;

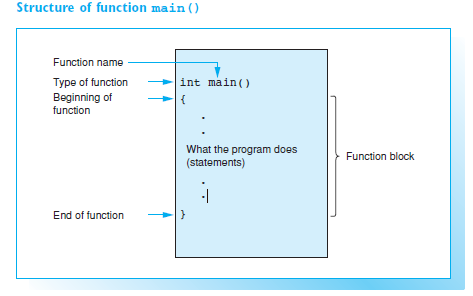
**Sample program**

|  |
| --- |
| #include <iostream>  using namespace std;  int main( int argc, char \* argv[] )  {  cout << "Enjoy Yourself with C++!" << endl;  return 0;  } |

**Sample output**

Enjoy Yourself with C++!

**STRUCTURE OF FUNCTION MAIN ()**



A C++ program is made up of objects with their accompanying *member functions* and *global functions*, which do not belong to any single particular class. Each function fulfills its own particular task and can also call other functions. You can create functions yourself or use ready-made functions from the standard library. You will always need to write the global function main() yourself since it has a special role to play; in fact it is the main program. The short programming example on the opposite page demonstrates two of the most important elements of a C++ program. The program contains only the function main() and displays a message. The first line begins with the number symbol, #, which indicates that the line is intended for the *preprocessor.* The preprocessor is just one step in the first translation phase and no object code is created at this time. You can type.

**#include <filename>**

to have the preprocessor copy the quoted file to this position in the source code. This allows the program access to all the information contained in the header file. The header file iostream comprises conventions for input and output streams. The word *stream* indicates that the information involved will be treated as a flow of data. Predefined names in C++ are to be found in the std (standard) namespace. The using directive allows direct access to the names of the std namespace. Program execution begins with the first instruction in function main(), and this is why each C++ program must have a main function. The structure of the function is shown on the opposite page. Apart from the fact that the name cannot be changed, this function’s structure is not different from that of any other C++ function. In our example the function main() contains two *statements*. The first statement

**cout << "Enjoy yourself with C++!" << endl;**

outputs the text string Enjoy yourself with C++! on the screen. The name cout (console output) designates an object responsible for output. The two less-than symbols, <<, indicate that characters are being “pushed” to the output stream. Finally endl (end of line) causes a line feed. The statement

**return 0;**

terminates the function main() and also the program, returning a value of 0 as an exit code to the calling program. It is standard practice to use the exit code 0 to indicate that a program has terminated correctly. Note that statements are followed by a semicolon. By the way, the shortest statement comprises only a semicolon and does nothing.

**USAGE OF “using namespace std;”**

* **using**: You are going to use it.
* **namespace**: To use what? A namespace.
* **std**: The std namespace (where features of the C++ Standard Library, such as string or vector, are declared).

After you write this instruction, if the compiler sees string it will know that you may be referring to std::string, and if it sees vector, it will know that you may be referring to std::vector. (Provided that you have included in your compilation unit the header files where they are defined, of course.)

If you *don't* write it, when the compiler sees string or vector it will not know what you are refering to. You will need to explicitly tell it std::string or std::vector, and if you don't, you will get a compile error.

## DIFFERENCE BETWEEN INT MAIN() AND VOID MAIN() AND MAIN()

Like any other function, main is also a function but with a special characteristic that the program execution always starts from the ‘main’. ‘int’ and ‘void’ are its return type. So, let’s discuss all of the three one by one.

* ***void main*** – The ANSI standard says "no" to the ‘void main’ and thus using it can be considered wrong. One should stop using the ‘void main’ if doing so.
* ***int main*** – ‘int main’ means that our function needs to return some integer at the end of the execution and we do so by returning 0 at the end of the program. 0 is the standard for the “successful execution of the program”.
* ***main*** – In C89, the unspecified return type defaults to **int**. So, ***main***is equivalent to **int main**in C89. But in C99, this is not allowed and thus one must use **int main**.

So, the preferred way is ***int main***