**Make Your Own Header File ?**

**Step1 : Type this Code**

|  |  |
| --- | --- |
| 1  2  3  4 | int add(int a,int b)  {  return(a+b);  } |

* In this Code write only function definition as you write in General C Program

**Step 2 : Save Code**

* **Save** Above Code with **[.h ] Extension** .
* Let name of our header file be **myhead** [ myhead.h ]
* Compile Code if required.

**Step 3 : Write Main Program**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | #include<stdio.h>  #include"myhead.h"    void main() {     int num1 = 10, num2 = 10, num3;     num3 = add(num1, num2);     printf("Addition of Two numbers : %d", num3);  } |

1. **Include Our New Header File** .
2. Instead of writing **< myhead.h>** use this terminology **“myhead.h”**
3. All the Functions defined in the **myhead.h header** file are now ready for use .
4. **Directly  call function add()**; [ Provide proper parameter and take care of return type ]

**Note**

1. While running your program precaution to be taken : Both files [ **myhead.h and sample.c** ] should be in same folder.

Example:2

void add(int a, int b)

{

    printf("Added value=%d\n", a + b);

}

void multiply(int a, int b)

{

    printf("Multiplied value=%d\n", a \* b);

}

**Using the created header file :**

// C program to use the above created header file

#include <stdio.h>

#include "myhead.h"

int main()

{

add(4, 6);

/\*This calls add function written in myhead.h

and therefore no compilation error.\*/

multiply(5, 5);

// Same for the multiply function in myhead.h

printf("BYE!See you Soon");

return 0;

}

Output:

Added value:10

Multiplied value:25

BYE!See you Soon

Example: 3

#include<stdio.h>

#include"swap.h"

void main()

{

int a=20;

int b=30;

swap (&a,&b);

printf ("a=%d\n", a);

printf ("b=%d\n",b);

}

Swap method is defined in swap.h file, which is used to swap two numbers by using a temporary variable.

Example:

void swap (int\* a, int\* b)

{

int tmp;

tmp = \*a;

\*a = \*b;

\*b = tmp;

}

**Summary**

* header file name must have a **.h** file extension.
* In this example, I have named **swap.h** header file.
* Instead of writing <swap.h> use this terminology **swap.h** for include custom header file.
* Both files **swap.h** and **main.c** must be in the same folder.

**Environment Variables**

When a program is executed, it receives information about the context in which it was invoked in two ways. The first mechanism uses the *argv* and *argc* arguments to its main function, and is discussed in [Program Arguments](https://www.gnu.org/software/libc/manual/html_node/Program-Arguments.html). The second mechanism uses *environment variables* and is discussed in this section.

The *argv* mechanism is typically used to pass command-line arguments specific to the particular program being invoked. The environment, on the other hand, keeps track of information that is shared by many programs, changes infrequently, and that is less frequently used.

Standard environment variables are used for information about the user’s home directory, terminal type, current locale, and so on; you can define additional variables for other purposes. The set of all environment variables that have values is collectively known as the *environment*.

Names of environment variables are case-sensitive and must not contain the character ‘=’. System-defined environment variables are invariably uppercase.

The values of environment variables can be anything that can be represented as a string. A value must not contain an embedded null character, since this is assumed to terminate the string.

|  |  |  |
| --- | --- | --- |
| • [Environment Access](https://www.gnu.org/software/libc/manual/html_node/Environment-Access.html) |  | How to get and set the values of environment variables. |
| • [Standard Environment](https://www.gnu.org/software/libc/manual/html_node/Standard-Environment.html) |  | These environment variables have standard interpretations. |

Example -2

/\*

\* C Program to Print Environment variables

\*/

#include <stdio.h>

void main(int argc, char \*argv[], char \* envp[])

{

int i;

for (i = 0; envp[i] != NULL; i++)

{

printf("\n%s", envp[i]);

}

}

$ cc arg7.c

$ a.out

HOSTNAME=localhost.localdomain

SELINUX\_ROLE\_REQUESTED=

SHELL=/bin/bash

TERM=xterm

HISTSIZE=1000

SSH\_CLIENT=192.168.7.43 49162 22

SELINUX\_USE\_CURRENT\_RANGE=

QTDIR=/usr/lib64/qt-3.3

QTINC=/usr/lib64/qt-3.3/include

SSH\_TTY=/dev/pts/8

USER=harika