Download Link - https://github.com/mchlbrnhrd/mbLib.git

```
#include <CMBMenu.hpp>

const char MenuFoo_pc[] PROGMEM = {"1. Foo"};
const char MenuFooA_pc[] PROGMEM = {"1.1 FooA"};
const char MenuFooB_pc[] PROGMEM = {"1.2 FooB"};
const char MenuTest1_pc[] PROGMEM = {"1.2.1 Test1"};
const char MenuTest2_pc[] PROGMEM = {"1.2.2 Test2"};
const char MenuBar_pc[] PROGMEM = {"2. Bar"};
const char MenuBarA_pc[] PROGMEM = {"2.1 BarA"};

enum MenuFID {
  MenuDummy,
  MenuFoo,
  MenuFooA,
  MenuFooB,
  MenuTest1,
  MenuTest2,
  MenuBar,
  MenuBarA
};

// define key types
enum KeyType {
  KeyNone, // no key is pressed
  KeyLeft,
  KeyRight,
  KeyEnter,
  KeyExit
};


CMBMenu<100> g_Menu;

void setup()
{
  Serial.begin(9600);
  Serial.println("============================");
  Serial.println("mblib - example for CMBMenu");
  Serial.println("============================");
  Serial.println("");
  Serial.println("l: left, r: right, e: enter, x: exit, m: print menu");
  Serial.println("");

  // ** menu **
  // add nodes to menu (layer, string, function ID)
  g_Menu.addNode(0, MenuFoo_pc , MenuFoo);
```

```
  g_Menu.addNode(1, MenuFooA_pc, MenuFooA);
  g_Menu.addNode(1, MenuFooB_pc, MenuFooB);
  g_Menu.addNode(2, MenuTest1_pc, MenuTest1);
  g_Menu.addNode(2, MenuTest2_pc, MenuTest2);

  g_Menu.addNode(0, MenuBar_pc, MenuBar);
  g_Menu.addNode(1, MenuBarA_pc, MenuBarA);


  // ** menu **
  // build menu and print menu
  // (see terminal for output)
  const char* info;
  g_Menu.buildMenu(info);
  g_Menu.printMenu();

  // ** menu **
  // print current menu entry
   printMenuEntry(info);
   pinMode(5,OUTPUT);
digitalWrite(5,HIGH);
}

void loop()
{
  // function ID
  int fid = 0;

  // info text from menu
  const char* info;

  // go to deeper or upper layer?
  bool layerChanged=false;

  // determine pressed key
  KeyType key = getKey();

  // ** menu **
  // call menu methods regarding pressed key
  switch(key) {
    case KeyExit:
      g_Menu.exit();
      break;
    case KeyEnter:
      g_Menu.enter(layerChanged);
      break;
    case KeyRight:
      g_Menu.right();
      break;
```

```
    case KeyLeft:
      g_Menu.left();
      break;
    default:
      break;
  }

  // ** menu **
  // pint/update menu when key was pressed
  // and get current function ID "fid"
  if (KeyNone != key) {
    fid = g_Menu.getInfo(info);
    printMenuEntry(info);
  }

  // ** menu **
  // do action regarding function ID "fid"
  if ((0 != fid) && (KeyEnter == key) && (!layerChanged)) {
    switch (fid) {
      case MenuFooA:
        FooA();
        break;
      case MenuBarA:
        BarA();
        break;
      case MenuTest1:
        Test1();
        break;
      case MenuTest2:
        Test2();
        break;
      default:
        break;
    }
  }
}

// ******************************************
// ** menu **
// printMenuEntry
// ******************************************
void printMenuEntry(const char* f_Info)
{
  String info_s;
  MBHelper::stringFromPgm(f_Info, info_s);

  // when using LCD: add/replace here code to
  // display info on LCD
```

```
  Serial.println("----------------");
  Serial.println(info_s);
  Serial.println("----------------");
}


// ****************************************
// getKey
// ****************************************
KeyType getKey()
{
  KeyType key = KeyNone;

  // here for demonstration: get "pressed" key from terminal
  // replace code when using push buttons
  while(Serial.available() > 0) {
    String Key_s = Serial.readString();
    Key_s.trim();
    Serial.println("");
    if(Key_s.equals("l")) { // left
      key = KeyLeft;
      Serial.println("<left>");
    } else if (Key_s.equals("r")) { // right
      key = KeyRight;
      Serial.println("<right>");
    } else if (Key_s.equals("e")) { // enter
      key = KeyEnter;
      Serial.println("<enter>");
    } else if (Key_s.equals("x")) { // exit
      key = KeyExit;
      Serial.println("<exit>");
    } else if (Key_s.equals("m")) { // print menu
      g_Menu.printMenu();
    }
  }

 return key;
}

// ****************************************
// FooA
// ****************************************
void FooA()
{
  Serial.println("Function FooA() was called.");
digitalWrite(5,LOW);
Serial.println("Relay 1 is ON.");
}
```

```
// ********************************************
// Test1
// ********************************************
void Test1()
{
  Serial.println("Function Test1() was called.");
  digitalWrite(5,HIGH);
  Serial.println("Relay 1 is OFF.");
}

// ********************************************
// Test2
// ********************************************
void Test2()
{
  Serial.println("Function Test2() was called.");
}

// ********************************************
// BarA
// ********************************************
void BarA()
{
  Serial.println("Function BarA() was called.");

}
```