Gang Hyun Kim

40097242

Lab 2

CDS

Give the main advantage and the main disadvantage for implementing a LED Blink as a CDS. Main advantage? Main disadvantage?

Advantages

* User-defined data structures are visible to all users
* Global access is very efficient

Disadvantages

* It is not possible to create more than one copy of the object.
* It is possible to have direct access to the data which is a lack of information hiding (an issue in procedural approach)

ADS

Give the main advantage and the main disadvantage for implementing a LED Blink as an ADS. Main advantage? Main disadvantage?

Advantages

* Data abstraction is applied which forces users to access through public functions
* Implementation can be modified without affecting users
* Data is protected in the implementation

Disadvantages

* Accessing data is less efficient
* Implementation file needs to be modified to add new functions
* Only one instance can be created

ADT

In the previous example, find the main advantage and the main disadvantage for implementing a Bling LED as an ADT. Main advantage? Main disadvantage?

Advantage

* New data types can be added to the language

Disadvantages

* Accessing data is even less efficient than ADS due to additional parameter
* Dynamic memory allocation is done which is less efficient than static allocations

1. The purpose of this lab is to access different input/output ports by manipulating registers and to apply different paradigms of software development being procedural, object based and object-oriented paradigms.
2. For this lab, there are some hardware resources that are required which are: computer/laptop, breadboard, Arduino nano microchip and a mini-usb to usb converter. The Arduino needs to be placed on the breadboard and the converter needs to be plugged in the microchip connected to the computer. This is only required for the accessing of input/output ports part of the lab. For the paradigms, only a computer or laptop is required.
3. For this lab, there are some software requirements. A compiler that can compile C and C++ code is required. It is recommended to use the AVR compiler due to its compatibility with working with microchips. For the full installation of this compiler, both the avr-gcc and avrdude is needed. Both these compilers need to be set as environment variables of the computer to be able to be recognized as a command on the command prompt. VSCode is recommended IDE for this lab because support multiple different programming languages which is very convenient. To compile and run the C code in the Arduino, it must be first transformed into a binary file then into a hex file which can then be pushed into the microchip. The following commands are used for these three steps.

avr-gcc -Wall -mmcu=atmega328p -Os -o myprogram.bin myfile1.c

avr-objcopy -O ihex myprogram.bin myprogram.hex

avrdude -c arduino -p m328p -P myserport -b 57600 -D -U flash:w:myprogram.hex:i

1. From tasks 1 to 4. The main goal is to make the Arduino’s built-in light to blink by using different programming paradigms. To initialize these tasks, the data direction register needs to be modified so that the selected port is in the input/output mode. This could be done by using the bitwise OR operator. To turn on the LED, a port register can be modified using the same operator. Then to turn off the LED, the port register can be cleared by using the bitwise AND operator. The code for the initialization and the on/off should look similar to the following image throughout the tasks from 1 to 4:

Text

Description automatically generated

A screenshot of a computer

Description automatically generated with low confidenceTask 2

By writing typedef struct for this concrete data structure (CDS), a user-defined datatype is created with the name Led. Which then some memory is allocated by creating a variable of type Led. With the implementation of the methods explained earlier, it is possible to setup the microchip and make the LED blink by executing the on and off methods in a while loop which runs indefinitely.

Text

Description automatically generatedTask 3

The implementation of the CDS to ADS is very simple. By removing the naming of the struct and by adding the static, it is possible to create a layer of abstraction. This forces the user to use the implemented methods to access the data which, in this case, is the value of the port register those controls whether the LED is on or off.

Task 4

In this task the blinking is created as an abstract data type (ADT). To do so, a structure needs to be created which is done in the code on line 5. Then a pointer to this structure is made which is also made into a datatype by calling typedef which is seen on line 6. The rest of the methods are implemented the same as all the previous tasks.

A screenshot of a computer

Description automatically generated with medium confidence

Task 5

The queue ADT is implemented in a similar fashion to the task 4. The only difference is that it needs to have attributes in the struct to store the queue data. When removing an element from a queue, the data will be returned, and each element will be shifted. When adding an element, it will be just added to the end of the list. To check if the queue is empty, each element, up to the queue pointer’s value, will be checked if each element is equal to null.

Text

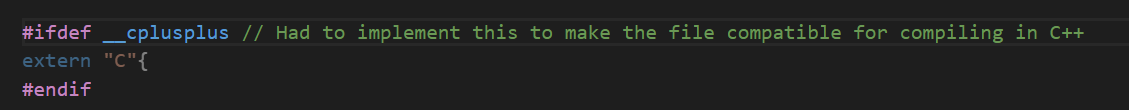
Description automatically generated

Text

Description automatically generated

Task 6

The goal of this task is to implement the ADT created in Task5 with a C++ wrapper. To do this, there are some changes that need to be made in Task5.



The code above needs to be implemented in Task5.h because of the differences in the way C and C++ is compiled. The rest of the methods are implemented the same way as the previous task.

Task 7

For task 7, there are some minor adjustments to be made from the C code’s implementation to the C++’s. For example, NULL needs to be implemented as nullptr. However, for the most part, the method implementation is similar. In C++, a class will need to be created with an integer array type attribute which will store the values.

1. The avr/io library was used to be able to directly access the Arduino microchip’s registers and the util/delay library was also used to be able to create delay between the turn on and turn off functions. fstream and io stream libraries were used to be able to print into the text file required for aunit testing.
2. In conclusion, this lab was a great way to learn about how one would manipulate registers in a microchip and understand the applications of different programming paradigms. I think one thing for me to improve upon is my understand between the compatibility of C and C++ code and how they differ when compiling because I struggled quite a bit when working on the queue ADT with a C++ wrapper. This lab has taught me to be able to implement CDS, ADS and ADT which I hope to be useful for me in the future.