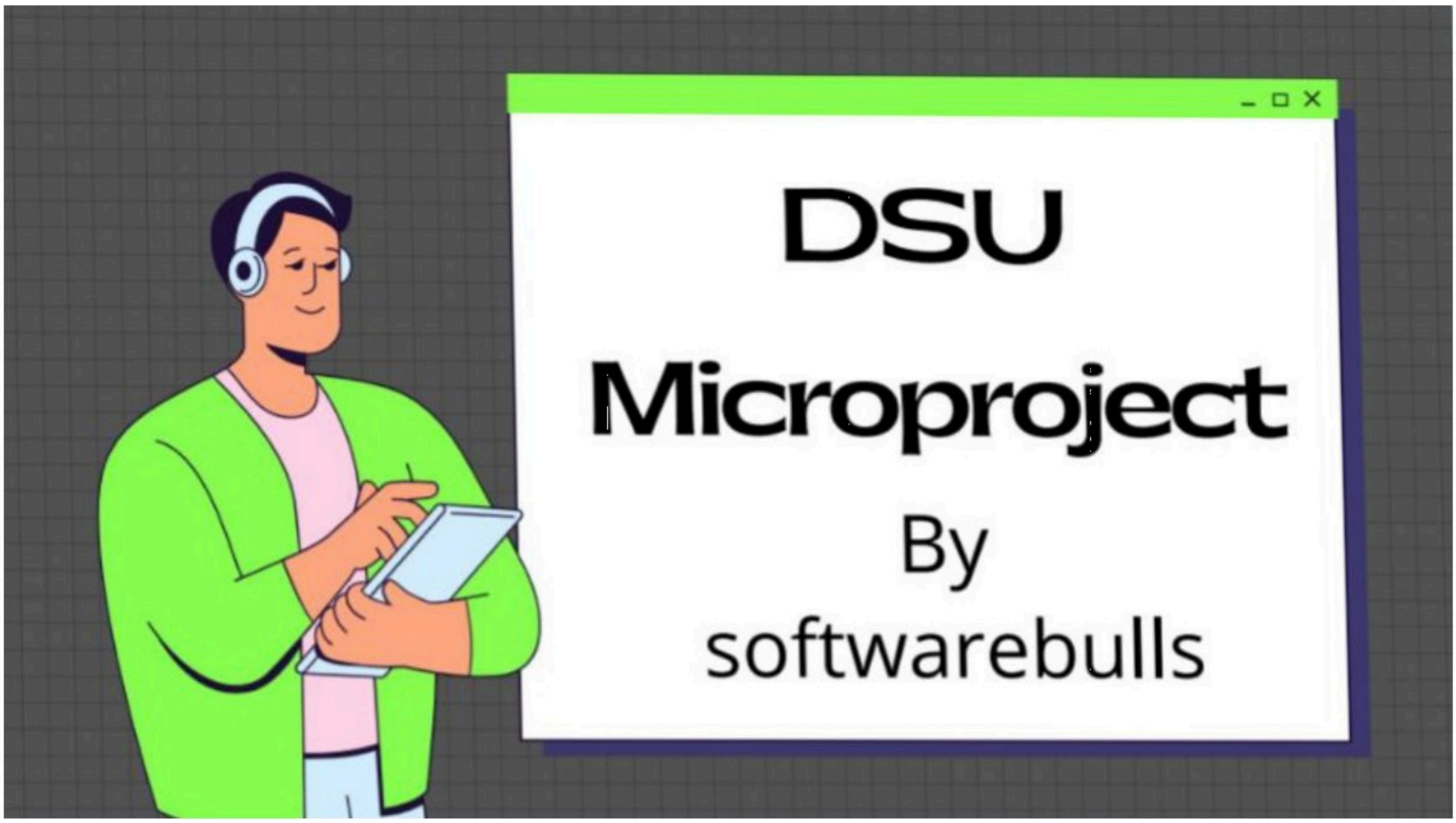
DSU
Microproject
By
softwarebulls

“A MICRO PROJECT REPORT”

ON

"ANLOAGE CLOCK”

SUBMITTED *BY*

1] SAYALI SUDHIR PATHAK

2] MOHD KASHIF MOHD ATEF

3] POOJA AMBADAS HIWALE

UNDER THE GUIDEOF

Ms. A. Borlepawar

DEPARTMENT OF COMPUTER ENGINEERING CSMSS

COLLEGE OF POLYTECHNIC,AURANGABAD

MAHARASTRA STATE,INDIA

2020-2021

Submission of Micro Project Entitled "ANLOAGE CLOCK"

Submitted to

CSMSS *COLLEGE OF* POLYTECHNIC, *AURANGABAD.*

BY

Research Students

1] SAYALI SUDHIR PATHAK [1911520079]

2] MOHD KASHIF MOHD ATEF [1911520069]

3] POOJA AMBADAS HIWALE [1911520056]

Under the Supervision of

MS. A.BORLEPAWAR

Project Guide

Department of Computer Engineering

CSMSS COLLEGE OF POLYTECHNIC, AURANGABAD,

Aurangabad. 431001(M.S.)

CSMSS COLLEGE OF

POLYTECHNIC

AURANGABAD

CERTIFICATE

This is to certify that **SAYALI PATAHK,MOHD KASHIF MOHD ATEF,POOJA HIWALE** of Third semester of Diploma in **Computer Engineering** of Institute **CSMSS College of Polytechnic (code:1152)** have completed The Micro Project satisfactorily in subject **COMPUTER GRAPHICS(22318)** for the academic year 2020-21as prescribed in the curriculum.

Place:…… **EnrollmentNO:……**

Date:…… **Exam SeatNo's:……...**

PROJECT GUIDE **HEAD OF THE DEPARTMENT**

PRINCIPAL

Acknowledgement

We would like to express our thanks to the people who have helped us most throughout our project. We would like to express our sincere thanks to the principal of CSMSS College of Polytechnic **MOHD KASHIF MOHD ATEF. G. B. DONGRE** for being always with us as a motivator. We are thankful to the H.O.D. of Applied Science Department **MR. V.N. SHAHANE** for her kind support. We are grateful to our Project Director **MRS. A. BORLEPAWAR** for continuous support and motivation for the project. His help made us possible to complete our project with all the accurate information. A special thanks of our goes to our friends who helped us in completing the project, where they all exchanged their own interesting ideas. We wish to thanks our parents for their personal support or attention who inspired us to go our own way. Finally, we would like to thank God who made all things possible for us till the end.

Sr. No.	Name of Student	Sign
1	SAYALI SUDHIR PATHAK (1911520079)	
2	MOHD KASHIF MOHD ATEF(1911520069)	
3	POOJA AMBADAS HIWALE (1911520056)	

ABSTRACT

In this present micro project we have studied different functions classes which was created by us. We have listed and have given information on the important functions of the 'C' language. We have understood importance of functions & syntax in **Data Structure Using C** Functions helps us in Programming. We have also learned how to create class and user defined functions and knew how to use these functions and class with help of object of the class.

Index

Introduction

What is Queue?

Queue is a linear data structure where elements are ordered in special fashion i.e. **FIFO (First In First Out)**. Which means element inserted first to the queue will be removed first from the queue.

In real life you have come across various queu

e examples. Such as queue of persons at ticket counter, where the first person entering queue gets ticket first.

Operations performed on Queue

On queue we generally perform two basic operations.

1. Enqueue (Insertion)
2. Dequeue (Removal)

Queue structure

Before you perform any operations on queue, you need a queue structure. Let us first define a custom type to represent our individual queue node.

```
typedef struct node
{
   int data;
   struct node * next;
} Queue;
```

Note: In the above declaration I have used typedef. It is used to create an alias for our new type. Hence, in program I will use Queue instead of struct node.

Read more about typedef in C language.

After defining queue structure node, we will need few variables to keep track of our queue. Let us define them one after another.

```
unsigned int size = 0;    // Size of queue
Queue *rear, *front;      // Reference of rear and front node in queue
```

How to enqueue an element in Queue using linked list?

Insertion of new element in queue is known as enqueue. You can enqueue a new element at rear of the queue, if its capacity is not full.

Step by step descriptive logic to enqueue an element in queue.

1. If queue size is more than capacity, then throw out of capacity error. Otherwise continue to next step.
2. Allocate memory for node of Queue type using malloc(). Say Queue *newNode = (Queue *) malloc (sizeof(Queue));.
3. Make sure that the newly created node points to nothing i.e. newNode->next = NULL;
4. Assign data to the new node, may be user input.
5. If queue is not empty then link rear node to newNode. Say (*rear)->next = newNode;.
6. Make newNode as rear. Since after each enqueue rear gets changed.
7. If its first node in queue then make it as front node too, say *front = *rear;.
8. Finally after each successful enqueue, increment size++ by one.

How to dequeue an element from Queue using linked list?

Removal of an existing element from queue is known as dequeue. You can perform dequeue from front of the queue, if its not empty.

Step by step descriptive logic to dequeue element from queue using linked list.

1. If queue is empty, then throw empty queue error. Otherwise continue to next step.
2. Get front element from queue, which is our required element to dequeue. Store it in some variable say Queue *toDequeue = *front;. Also store its data to some variable say int data = toDequeue->data;
3. Move front node ahead. To make sure that it points to next node after the first node. Say *front = (*front)->next;.
4. Decrement size--; by one.
5. Free the dequeued element from memory to save resources, say free(toDequeue);.
6. Return data, which is our required dequeued element.

Program Code

```
/**
 * Queue implementation using linked list in C.
 */

#include <stdio.h>
#include <stdlib.h>
#include <limits.h>


#define CAPACITY 100    // Queue max capacity


/* Queue structure definition */
typedef struct node
{
   int data;
   struct node * next;
} Queue;    // Queue is a typedef of struct node

/* Queue size */
unsigned int size = 0;


int enqueue(Queue ** rear, Queue ** front, int data);
int dequeue(Queue ** front);
int getRear(Queue * rear);
int getFront(Queue * front);
int isEmpty();
int isFull();


int main()
{
   int ch, data;
   Queue *rear, *front;

   rear  = NULL;
   front = NULL;

/* Run indefinitely until user manually terminates */
while (1)
{
   /* Queue menu */
   printf("-----------------------------------------\n");
   printf("  QUEUE LINKED LIST IMPLEMENTATION PROGRAM  \n");
   printf("-----------------------------------------\n");
   printf("1. Enqueue\n");
   printf("2. Dequeue\n");
```

```
printf("3. Size\n");
printf("4. Get Rear\n");
printf("5. Get Front\n");
printf("0. Exit\n");
printf("-----------------------------------------\n");
printf("Select an option: ");

scanf("%d", &ch);

/* Menu control switch */
switch (ch)
{
   case 1:
      printf("\nEnter data to enqueue: ");
      scanf("%d", &data);

      // Enqueue function returns 1 on success
      // otherwise 0
      if (enqueue(&rear, &front, data))
         printf("Element added to queue.");
      else
         printf("Queue is full.");

      break;

case 2:
   data = dequeue(&front);

   // on success dequeue returns element removed
   // otherwise returns INT_MIN
   if (data == INT_MIN)
      printf("Queue is empty.");
   else
      printf("Data => %d", data);

   break;

case 3:

   // isEmpty() function returns 1 if queue is emtpy
   // otherwise returns 0
   if (isEmpty())
      printf("Queue is empty.");
   else
      printf("Queue size => %d", size);

   break;

case 4:
```

```
            data = getRear(rear);

            if (data == INT_MIN)
                printf("Queue is empty.");
            else
                printf("Rear => %d", data);

            break;

        case 5:

            data = getFront(front);

            if (data == INT_MIN)
                printf("Queue is empty.");
            else
                printf("Front => %d", data);

            break;

        case 0:
            printf("Exiting from app.\n");
            exit(0);

        default:
                printf("Invalid choice, please input number between (0-5).");
                break;
        }

        printf("\n\n");
    }
}



/**
 * Enqueues/Insert an element at the rear of a queue.
 * Function returns 1 on success otherwise returns 0.
 */
int enqueue(Queue ** rear, Queue ** front, int data)
{
    Queue * newNode = NULL;

    // Check queue out of capacity error
    if (isFull())
    {
        return 0;
    }

    // Create a new node of queue type
```

```
newNode = (Queue *) malloc (sizeof(Queue));

// Assign data to new node
newNode->data = data;

// Initially new node does not point anything
newNode->next = NULL;

// Link new node with existing last node
if ( (*rear) )
{
   (*rear)->next = newNode;
}


// Make sure newly created node is at rear
*rear = newNode;

// Link first node to front if its NULL
if ( !( *front) )
{
   *front = *rear;
}

// Increment quque size
   size++;

   return 1;
}


/**
 * Dequeues/Removes an element from front of the queue.
 * It returns the element on success otherwise returns
 * INT_MIN as error code.
 */
int dequeue(Queue ** front)
{
   Queue *toDequque = NULL;
   int data = INT_MIN;

// Queue empty error
if (isEmpty())
{
   return INT_MIN;
}

// Get element and data to dequeue
toDequque = *front;
data = toDequque->data;
```

```
    // Move front ahead
    *front = (*front)->next;

    // Decrement size
    size--;

    // Clear dequeued element from memory
    free(toDequque);

    return data;
}


/**
 * Gets, element at rear of the queue. It returns the element
 * at rear of the queue on success otherwise return INT_MIN as
 * error code.
 */
int getRear(Queue * rear)
{
    // Return INT_MIN if queue is empty otherwise rear.
    return (isEmpty())
            ? INT_MIN
            : rear->data;
}


/**
 * Gets, element at front of the queue. It returns the element
 * at front of the queue on success otherwise return INT_MIN as
 * error code.
 */
int getFront(Queue * front)
{
    // Return INT_MIN if queue is empty otherwise front.
    return (isEmpty())
            ? INT_MIN
            : front->data;
}


/**
 * Checks, if queue is empty or not.
 */
int isEmpty()
{
    return (size <= 0);
}
```

```
/**
 * Checks, if queue is within the maximum queue capacity.
 */
int isFull()
{
   return (size > CAPACITY);
}
```

CONCLUSION

We have studied that how to do a programe in graphics using various function ,and also looping the statement how they wrok and it is used for the analog clock, it is show real time .

REFERANCES

WWW.youtube.com
www.programarrize.com
https://www.sanfoundary.com