Relazione del Progetto di Informatica

Abu Shahid Islam & Giovanni Sebastiani

26 novembre 2024

Indice

1 Struttura

Il programma è diviso in 2 **Menu** principali:

1.1 Menu Principale

- Qui l'utente può scegliere tra 9 opzioni:
  - **Aggiungi un libro**
  - **Visualizza i libri**
  - **Rimuovi un libro**
  - **Cerca un libro per autore**
  - **Libro più costoso**
  - **Ordina libri in base al prezzo**
  - **Visualizza per fascia di prezzo**
  - **Prestiti**
  - **Esci**

1.2 Prestiti

- Qui l'utente può scegliere tra 4 opzioni:
  - **Prendi in prestito**
  - **Restituisci il prestito**
  - **Visualizza i libri in prestito**
  - **Esci**

  .
- Questo programma anche se fatto con Array statici (Array a dimensione fissa) dá la possibilità all'utente illimitati libri

2 Menu Principale

2.1 Gestione Array

Funzionamento del Menu Principale:

- Chiamato **Program.cs** contiene il *main*.
- Il main contiene tutti gli array per la gestione della libreria

```
string[] Titoli = new string[2];
string[] Autori = new string[2];
double[] prezzo = new double[2];
string[] categoria = new string[2];
string[] casa_editrice = new string[2];
int[] copie = new int[2];
```

Listing 1: Array della libreria

- Contiene gli array per la gestione dei prestiti

```
int prestiti = 0;
string[] libri_prestito = new string[2];
string[] utenti_prestito_nome = new string[2];
string[] utenti_prestito_cognome = new string[2];
string[] giorno_preso = new string[2];
string[] tempo_trattenuto = new string[2];
int[] codice_prestito = new int[2];
int retuns = 0;
```

Listing 2: Array dei prestiti

- Alcune funzioni necessitano il numero di libri presenti nella libreria

```
int libri_unici = 0;
int libri_unici_copia = 0;
```

Listing 3: Funzione per il numero di libri

2.2 Visual

- Tutta la parte visuale è gestita da un menu comandabile con le freccie ***MenuArrow.cs***.
  Questa funzione ha come *return* la scelta dell'utente.
- In base alla scelta viene scelta la funzione da eseguire.
-

```
        //Menu
while (true)
{
    Console.Clear();

    //Menu con le freccie
    int scelta = MenuArrow.menuArrow();

    switch (scelta)
    {
        case 1:

            AddBook.addBook(ref Titoli, ref Autori, ref prezzo, ref categoria, ref casa_editrice,
                ref copie, ref libri_unici, ref libri_unici_copia);
            break;

        case 2:

            ShowBook.showBook(Titoli, copie);
            break;

        case 3:

            RemoveBook.removeBook(ref Titoli, ref Autori, ref prezzo, ref categoria, ref
                casa_editrice, ref copie, ref libri_unici, ref libri_unici_copia);
            break;

        case 4:

            SearchByAuthor.searchByAuthor(Titoli, Autori, copie);
            break;
        case 5:

            MostExpeBook.mostExpeBook(Titoli, prezzo);
            break;

        case 6:

            Reorder.reorder(ref Titoli, ref Autori, ref prezzo, ref categoria, ref casa_editrice,
                ref copie);
            break;

        case 7:

            PriceRange.priceRange(Titoli, prezzo, copie);
            break;

        case 8:

            MenuPrestiti.menuPrestiti(Titoli, Autori, prezzo, categoria, casa_editrice, copie, ref
                libri_unici_copia , ref prestiti, ref libri_prestito,
                ref utenti_prestito_nome, ref utenti_prestito_cognome, ref giorno_preso, ref
                    tempo_trattenuto, ref codice_prestito, ref retuns);
            break;

        case 9:

            Console.Clear();
            Console.WriteLine("Arrivederci!");
            Console.ReadKey();

            return;
    }

}
```

Listing 4: Funzione per il menu

2.2.1 MenuArrow.cs

Questa funzione permette di navigare nel menu con le frecce. Le frecce funzionano tramite un array strings che contiene le scelte del menu. Un intero pos che indica la posizione attuale nel menu; e leggendo il tasto premuto, se è una freccia, cambia la posizione.

Nel seguente codice si può vedere come funziona la chat colorata e la freccia che indica la scelta dell'utente.

```
    //Stampa il menu
static public void Menu(int pos)
{
    // Menu
    string[] strings =
    {
    "Inserisci un libro",
    "Consulta tutti i libri della biblioteca",
    "Rimuovi un libro",
    "Ricerca libri per autore",
    "Stampa libro piu costoso",
    "Ordina i libri in base al prezzo",
    "Stampa i libri per fascia di prezzo",
    "Prestito",
    "Esci"
    };

    // Calcola la larghezza della console

    int consoleWidth = Console.WindowWidth;

    for (int i = 0; i < strings.Length; i++)
    {
        // Calcola il numero di spazi iniziali per centrare la stringa
        int padding = (consoleWidth - strings[i].Length) / 2;

        // Assicurati che il padding sia almeno 0
        if(padding < 0)
        {
            padding = 0;
        }


        // Colore verde per l'opzione selezionata
        if (i == pos - 1)
        {
            Console.ForegroundColor = ConsoleColor.Green;
            Console.WriteLine(new string(' ', padding) + $"> {strings[i]}");
        }
        else
        {
            Console.ForegroundColor = ConsoleColor.White;
            Console.WriteLine(new string(' ', padding) + $"  {strings[i]}");
        }
    }

    // Resetta il colore
    Console.ResetColor();
}
```

Listing 5: MenuArrow.cs

- **Console.WindowWidth** restituisce la larghezza della console.
- **Console.ForegroundColor** cambia il colore del testo.
- **Console.ResetColor** resetta il colore del testo.
- **padding** calcola il numero di spazi iniziali per centrare la stringa.
- **new string(' ', padding)** crea una stringa di spazi in base al padding.

Invece qui si può vedere come funziona la freccia e la scelta dell'utente.

```
//Stampa il menu con la freccia

static public int menuArrow()
{
    //Leggo il tasto premuto
    ConsoleKeyInfo key;
    int pos = 1;
    do
    {
        Console.Clear();
        Console.WriteLine();
        Bibleoteca.Logo();
        Menu(pos);
        Console.WriteLine();
        key = Console.ReadKey();
        Se premo freccia giu incremento la posizione quindi va all opzione dopo
        if (key.Key == ConsoleKey.DownArrow)
        {
            if (pos < 9)
            {
                pos++;
            }
        }
        //Se premo freccia su decremento la posizione quindi va all'opzione prima
        else if (key.Key == ConsoleKey.UpArrow)
        {
            if (pos > 1)
            {
                pos--;
            }
        }
    } while (key.Key != ConsoleKey.Enter);
    return pos;
}
```

Listing 6: MenuArrow.cs

- **ConsoleKeyInfo** legge il tasto premuto.
- **Console.Clear** pulisce la console.
- **Console.ReadKey** legge il tasto premuto.
- **ConsoleKey.DownArrow** controlla se è stata premuta la freccia giù.
- **ConsoleKey.UpArrow** controlla se è stata premuta la freccia su.
- **ConsoleKey.Enter** controlla se è stato premuto il tasto invio.
- **pos** indica la scelta corrente.

2.3 AddBook.cs

- Questa funzione permette di aggiungere un libro alla libreria.

- ### 2.3.1 Step 1

  Primo step è controllare se ci sono posti disponibili dove mettere i libri. In caso contrario, il programma crea un nuovo spazio tramite *Array.Resize*.

```
Console.Clear();
//Se abbiamo raggiunto il massimo
if (libri_unici == Titoli.Length)
{
    //Resize degli array

    Array.Resize(ref Titoli, Titoli.Length + 1);
    Array.Resize(ref Autori, Autori.Length + 1);
    Array.Resize(ref prezzo, prezzo.Length + 1);
    Array.Resize(ref categoria, categoria.Length + 1);
    Array.Resize(ref casa_editrice, casa_editrice.Length + 1);
    Array.Resize(ref copie, copie.Length + 1);
}
```

Listing 7: Controllo spazio disponibile

2.3.2 Step 2

- Inserimento dei dati del libro.
- Il programma controlla all'inserimento se il libro è già presente.
- in caso sia un libro nuovo chiede le altre informazioni.
- Se avviene qualche errore, Es. inserimento di una stringa al posto di un double, il programma annulla l'inserimento.

```
Console.WriteLine("Inserisci il titolo del libro");
string titolo = Console.ReadLine();

    //Libro gia presente o meno

for (int i = 0; i < Titoli.Length; i++)
{
    if (Titoli[i] == null)
    {
        continue;
    }
    if (ToLower.ToLowerString(Titoli[i]) == ToLower.ToLowerString(titolo))
    {
        copie[i]++;
        Console.WriteLine("Il libro e gia presente nei nostri archivi, non servono le altre informazioni"
            );
        Console.ReadKey();
        return;
    }
}

try{
    Titoli[libri_unici] = titolo;

    Console.WriteLine("Inserisci l'autore del libro");
    Autori[libri_unici] = Console.ReadLine();

    Console.WriteLine("Inserisci il prezzo");
    prezzo[libri_unici] = double.Parse(Console.ReadLine());

    Console.WriteLine("Inserisci la categoria");
    categoria[libri_unici] = Console.ReadLine();

    Console.WriteLine("Inserisci la casa editrice");
    casa_editrice[libri_unici] = Console.ReadLine();

    copie[libri_unici] = 1;
}
catch (Exception e)
{
    Console.WriteLine("Errore, Qualcosa e andato storto");
    Console.ReadKey();
    Array.Resize(ref Titoli, Titoli.Length - 1);
    Array.Resize(ref Autori, Autori.Length - 1);
    Array.Resize(ref prezzo, prezzo.Length - 1);
    Array.Resize(ref categoria, categoria.Length - 1);
    Array.Resize(ref casa_editrice, casa_editrice.Length - 1);
    Array.Resize(ref copie, copie.Length - 1);
    return;
}
```

Listing 8: Inserimento dei dati

- **Array.Resize** aumenta/diminuisce la dimensione dell'array.

2.3.3 Step 3

- Ultimo step è la stampa dei dati inseriti

```
Console.WriteLine("Libro aggiunto con successo");
Console.ReadKey();

string[] strings =
{
    $"Libro: ",
    Titoli[libri_unici],
    $"Autore: ",
    Autori[libri_unici],
    $"Prezzo: ",
    prezzo[libri_unici].ToString(),
    $"Categoria: ",
    categoria[libri_unici],
    $"Casa Editrice",
    casa_editrice [libri_unici],

};

Console.Clear();
int padding = 15;

for(int i=0; i < strings.Length; i+=2)
{

    Console.Write(new string(' ', padding) + strings[i]);
    int padding2 = Console.WindowHeight - (padding - strings[i].Length)/2;

    Console.Write(new string(' ', padding2-i) + strings[i+1]);

    if(i == 4)
    {
        Console.Write(" Eur");
    }
    Console.WriteLine();
}

//Aumento libri unici
libri_unici++;
libri_unici_copia++;

Console.ReadKey();
```

Listing 9: Stampa dei dati inseriti

2.4 ShowBook.cs

2.4.1 Step 1

Lo scopo di questa funzione è di visualizzare tutti i libri presenti nella libreria.

Questa funzione mostra un menu con tutti i libro, e alla selezione di ogni libro, mostra le informazioni di quel libro. Il modo di funzionare è simile al menu principale, ma con l'obbiettivo di visualizzare i libri.

```
public static void Menu(int pos, string[] Titoli)
{
    Console.Clear();
    Console.WriteLine("Indice Libri");

    for(int i=0; i<Titoli.Length; i++)
    {
        if (i == pos-1)
        {
            Console.ForegroundColor = ConsoleColor.Green;
            Console.WriteLine($"> [{i+1}] {Titoli[i]}");
            Console.ResetColor();
        }
        else
        {
            Console.WriteLine($"  [{i+1}] {Titoli[i]}");
        }
    }

}

public static void showBook(string[] Titoli, int[] copie, string[] Autori, double[] prezzo, string[]
    categoria, string[] casa_editrice)
{

    ConsoleKeyInfo key;
    int pos = 1;
    do{
        Menu(pos, Titoli);
        key = Console.ReadKey();

        if(key.Key == ConsoleKey.DownArrow)
        {
            if(pos == Titoli.Length)
            {
                pos = 1;
            }
            else
            {
                pos++;
            }
        }
        else if(key.Key == ConsoleKey.UpArrow)
        {
            if(pos == 1)
            {
                pos = Titoli.Length;
            }
            else
            {
                pos--;
            }
        }
    }while(key.Key != ConsoleKey.Enter);

    Console.Clear();

    //Stampa libri
    Console.WriteLine("Titolo: " + Titoli[pos-1]);
    Console.WriteLine("Autore: " + Autori[pos-1]);
    Console.WriteLine("Prezzo: " + prezzo[pos-1] + "Eur");
    Console.WriteLine("Categoria: " + categoria[pos-1]);
    Console.WriteLine("Casa Editrice: " + casa_editrice[pos-1]);
    Console.WriteLine("Copie: " + copie[pos-1]);

    Console.WriteLine("Premi un tasto per tornare al menu");
    Console.ReadKey();

    return;
}
```

2.5 RemoveBook.cs

2.5.1 Step 1

Lo scopo di questa funzione è di rimuovere un libro dalla libreria. Funziona con lo stesso identico schema di Menu visto precedentemente. Ogni libro ha un numero associato, e alla selezione di quel numero, il libro viene rimosso. In caso ci sia piú di una copia elimina prima una copia, e poi il libro.

```
public static void removeBook(ref string[] Titoli, ref string[] Autori, ref double[] prezzo, ref string[]
     categoria, ref string[] casa_editrice, ref int[] copie, ref int libri_unici, ref int
    libri_unici_copia)
{
    ConsoleKeyInfo key;
    int lettura = 1;
    do{
        Menu(lettura, Titoli, copie);
        key = Console.ReadKey();

        if(key.Key == ConsoleKey.DownArrow)
        {
            if(lettura == Titoli.Length)
            {
                lettura = 1;
            }
            else
            {
                lettura++;
            }
        }
        else if(key.Key == ConsoleKey.UpArrow)
        {
            if(lettura == 1)
            {
                lettura = Titoli.Length;
            }
            else
            {
                lettura--;
            }
        }
    }while(key.Key != ConsoleKey.Enter);
}
```

- **Menu** è la funzione che stampa il menu.
- **ConsoleKey.DownArrow** controlla se è stata premuta la freccia giù.
- **ConsoleKey.UpArrow** controlla se è stata premuta la freccia su.
- **ConsoleKey.Enter** controlla se è stato premuto il tasto invio.

2.5.2 Step 2

Eliminazione copia

```
    if (copie[lettura - 1] > 1)
    {
        copie[lettura - 1]--;
        Console.Clear();
        Console.WriteLine("Copia eliminata con successo");
        Console.ReadKey();
        Console.Clear();
        return;

    }
```

- **copie[lettura - 1] ¿ 1** controlla se ci sono piú di una copia.
- **copie[lettura - 1]–** elimina una copia.
- **Console.Clear** pulisce la console.

2.5.3 Step 3

Eliminazione libro

```
    Titoli[lettura - 1] = "";
    Autori[lettura - 1] = "";
    prezzo[lettura - 1] = 0;
    categoria[lettura - 1] = "";
    casa_editrice[lettura - 1] = "";
    copie[lettura - 1] = 0;

    for (int i = lettura - 1; i < Titoli.Length - 1; i++)
    {
        string temp;
        int tempint;
        double tempdouble;

        temp = Titoli[i + 1];
        Titoli[i] = temp;

        temp = Autori[i + 1];
        Autori[i] = temp;

        tempdouble = prezzo[i + 1];
        prezzo[i] = tempdouble;

        temp = categoria[i + 1];
        categoria[i] = temp;

        temp = casa_editrice[i + 1];
        casa_editrice[i] = temp;

        tempint = copie[i + 1];
        copie[i] = tempint;
    }

    //Resize degli array
    Array.Resize(ref Titoli, Titoli.Length - 1);
    Array.Resize(ref Autori, Autori.Length - 1);
    Array.Resize(ref prezzo, prezzo.Length - 1);
    Array.Resize(ref categoria, categoria.Length - 1);
    Array.Resize(ref casa_editrice, casa_editrice.Length - 1);
    Array.Resize(ref copie, copie.Length - 1);

    libri_unici--;
    libri_unici_copia--;

    Console.Clear();
    Console.WriteLine("Il libro e stato eliminato");
    Console.ReadKey();
    Console.Clear();
```

- **Titoli[lettura - 1] = ””** elimina il titolo.
- **Array.Resize** diminuisce la dimensione dell'array.
- **libri_unici**– diminuisce il numero di libri unici.
- **libri_unici_copia**– diminuisce il numero di libri unici copia.

2.6 SearchByAuthor.cs

2.6.1 Step 1

Menu (guardare sopra per spiegazione) e ricerca per autore.

Il seguente codice, crea un array con tutti gli autori presenti, e permette di scegliere un autore.

```
static public void searchByAuthor(string[] Titoli, string[] Autori, int[] copie)
{
    Console.Clear();
    Console.WriteLine("Autori Presenti");

    string[] autori_unici = new string[1];

    //Metto dentro l'array tutti i autori
    for (int i = 0; i < Autori.Length; i++)
    {
        if(Autori[i] == null) {
            continue;
        }
        //Controllo che non ci siano doppioni
        bool doppione = false;
        for (int j = 0; j < autori_unici.Length; j++)
        {
            if (autori_unici[j] == null)
            {
                autori_unici[j] = Autori[i];
                break;
            }

            if (ToLower.ToLowerString(Autori[i]) == ToLower.ToLowerString(autori_unici[j]))
            {
                doppione = true;
                break;
            }
        }

        if (!doppione)
        {
            Array.Resize(ref autori_unici, autori_unici.Length + 1);
            autori_unici[autori_unici.Length - 1] = Autori[i];
        }

    }

    if(Autori[0] == null)
    {
        Console.WriteLine("Nessun autore presente");
        Console.ReadKey();
        return;
    }
```

- **autori_unici** contiene tutti gli autori presenti.
- **ToLower.ToLowerString** trasforma la stringa in minuscolo.
- **Array.Resize** aumenta la dimensione dell'array.

2.6.2 Step 2

Gestore della scelta dell'autore

```
    //Stampo
    ConsoleKeyInfo key;
    int scelta = 1;
    do{
        Menu(scelta, autori_unici);
        key = Console.ReadKey();

        if(key.Key == ConsoleKey.DownArrow)
        {
            if(scelta == autori_unici.Length)
            {
                scelta = 1;
            }
            else
            {
                scelta++;
            }
        }
        else if(key.Key == ConsoleKey.UpArrow)
        {
            if(scelta == 1)
            {
                scelta = autori_unici.Length;
            }
            else
            {
                scelta--;
            }
        }
    }while(key.Key != ConsoleKey.Enter);
```

- **Menu** è la funzione che stampa il menu.
- **ConsoleKey.DownArrow** controlla se è stata premuta la freccia giù.
- **ConsoleKey.UpArrow** controlla se è stata premuta la freccia su.
- **ConsoleKey.Enter** controlla se è stato premuto il tasto invio.
- **scelta** indica la scelta dell'utente.

2.6.3 Step 3

Stampa dei libri dell'autore scelto, tramite un ciclo for che controlla gli autori di tutti i libri e in caso affermativo stampa

```
    //Stampo i libri dell'autore scelto
    Console.Clear();
    Console.WriteLine($"Libri di {autori_unici[scelta - 1]}");
    int pos = 0;
    for (int i = 0; i < Autori.Length; i++)
    {
        if (ToLower.ToLowerString(Autori[i]) == ToLower.ToLowerString(autori_unici[scelta - 1]))
        {
            Console.WriteLine($"[{pos + 1}] {Titoli[i]} \t x{copie[i]}");
            pos++;
        }

    }

    Console.ReadKey();
    return;
}
```

2.7 MostExpeBook.cs

2.7.1 Step 1

Lo scopo di questa funzione è di trovare il libro piú costoso. Cerca la posizione del prezzo piu alto e stampa il libro.

```
public static void mostExpeBook(string[] Titoli, double[] prezzo)
{
    Console.Clear();
    double max = -1;
    int pos = 0;

    for (int i = 0; i < prezzo.Length; i++)
    {
        if (prezzo[i] > max)
        {
            max = prezzo[i];
            pos = i;
        }
    }

    Console.WriteLine($"Il libro piu costoso: {Titoli[pos]} \t {prezzo[pos]} Eur");
    Console.ReadKey();
    return;
}
```

2.8 Reorder.cs

2.8.1 Step 1

Lo scopo di questa funzione è di ordinare i libri in base al prezzo. Viene usato un algoritmo di ordinamento Bubble Sort. Si ha una scelta tra ordinamento crescente e decrescente.

Menu

```
    Console.Clear();
    Console.WriteLine("Scegli come riordinare");
    Console.WriteLine("[1] Crescente \t [2] Decrescente");
    int scelta;
    try
    {
        scelta = int.Parse(Console.ReadLine());
    }
    catch (Exception e)
    {
        Console.WriteLine("Errore: Inserire un numero valido");
        Console.ReadKey();
        return;
    }

    if(scelta != 1 && scelta != 2)
    {
        Console.WriteLine("Errore: Inserire un numero valido");
        Console.ReadKey();
        return;
    }
```

2.8.2 Step 2

Ordinamento crescente

```
if (scelta == 1)
    {
        for(int i = 0; i < prezzo.Length-1; i++)
        {
            for (int j = i + 1; j < prezzo.Length; j++)
            {
                if (prezzo[i] > prezzo[j])
                {

                    string temp = Titoli[i];
                    Titoli[i] = Titoli[j];
                    Titoli[j] = temp;

                    temp = Autori[i];
                    Autori[i] = Autori[j];
                    Autori[j] = temp;

                    double temp2 = prezzo[i];
                    prezzo[i] = prezzo[j];
                    prezzo[j] = temp2;

                    temp = categoria[i];
                    categoria[i] = categoria[j];
                    categoria[j] = temp;

                    temp = casa_editrice[i];
                    casa_editrice[i] = casa_editrice[j];
                    casa_editrice[j] = temp;

                    int temp3 = copie[i];
                    copie[i] = copie[j];
                    copie[j] = temp3;
                }
            }
        }
    }
```

Ordinamento Decrescente

```
if (scelta == 1)
{
    for(int i = 0; i < prezzo.Length-1; i++)
    {
        for (int j = i + 1; j < prezzo.Length; j++)
        {
            if (prezzo[i] < prezzo[j])
            {

                string temp = Titoli[i];
                Titoli[i] = Titoli[j];
                Titoli[j] = temp;

                temp = Autori[i];
                Autori[i] = Autori[j];
                Autori[j] = temp;

                double temp2 = prezzo[i];
                prezzo[i] = prezzo[j];
                prezzo[j] = temp2;

                temp = categoria[i];
                categoria[i] = categoria[j];
                categoria[j] = temp;

                temp = casa_editrice[i];
                casa_editrice[i] = casa_editrice[j];
                casa_editrice[j] = temp;

                int temp3 = copie[i];
                copie[i] = copie[j];
                copie[j] = temp3;
            }
        }
    }
}
```

2.8.3 Step 3

Stampa dei libri ordinati

```
Console.WriteLine("Libri riordinati con successo. Premere un tasto per continuare");
Console.ReadKey();
Console.Clear();
Console.WriteLine("Indice  - Titolo - Autore - Prezzo - Categoria - Casa Editrice - Copie");
for (int  i = 0;  i < Titoli.Length;  i++)
{
    Console.WriteLine($"{i+1} - {Titoli[i]} - {Autori[i]} - {prezzo[i]} - {categoria[i]} - {
        casa_editrice[i]} - x{copie[i]}");
}

Console.ReadKey();
return;
```

2.9 PriceRange.cs

2.9.1 Step 1

Lo scopo di questa funzione è di visualizzare i libri in base ad una fascia di prezzo. Viene chiesto all'utente di inserire un prezzo minimo e massimo, e vengono stampati i libri in quella fascia.

```
public static void priceRange(string[] Titoli, double[] prezzo, int[] copie)
{
    Console.Clear();
    Console.WriteLine("Inserisci la fascia di prezzo");
    double min, max;
    try
    {
        Console.WriteLine("Minimo");
        min = double.Parse(Console.ReadLine());
        Console.WriteLine("Massimo");
        max = double.Parse(Console.ReadLine());
    }
    catch (Exception e)
    {
        Console.WriteLine("Errore: Inserire un numero valido");
        Console.ReadKey();
        return;

    }

    if (min > max)
    {
        Console.WriteLine("Errore: Il minimo non puo essere maggiore del massimo");
        Console.ReadKey();
        return;
    }

    for(int i = 0; i < prezzo.Length; i++)
    {
        if (prezzo[i] >= min && prezzo[i] <= max)
        {
            Console.WriteLine($"[{i}] {Titoli[i]} \t {copie[i]} Eur \t x{copie[i]}");

            Console.WriteLine();
        }
    }

    Console.ReadKey();
    return;

}
```

3 Prestiti

3.1 MenuPrestiti.cs

3.1.1 Step 1

Lo scopo di questa funzione è di gestire i prestiti. Viene chiesto all'utente di scegliere tra 4 opzioni:

- Prendi in prestito
- Restituisci il prestito
- Visualizza i libri in prestito
- Esci

Viene tutto salvato negli array citati all'inizio. Essendo il menu uguale a quello principale, non verrà spiegato.

```
    ConsoleKeyInfo key;
    int pos = 1;
    do
    {
        Console.Clear();
        Console.WriteLine();
        LogoPrestito.logoPrestito();
        Menu(pos);
        key = Console.ReadKey();
        //Se premo freccia giu incremento la posizione quindi va all'opzione dopo
        if (key.Key == ConsoleKey.DownArrow)
        {
            if (pos < 4)
            {
                pos++;
            }
            if(pos == 5)
            {
                pos = 1;
            }
        }
        //Se premo freccia su decremento la posizione quindi va all'opzione prima
        else if (key.Key == ConsoleKey.UpArrow)
        {
            if (pos > 1)
            {
                pos--;
            }
            if (pos == 0)
            {
                pos = 4;
            }
        }
    } while (key.Key != ConsoleKey.Enter);
```

switch per la scelta dell'utente

```
    switch (pos)
    {
        case 1:

            TakeBorrow.takeBorrow(Titoli,copie, ref libri_prestito, ref prestiti, ref
                utenti_prestito_nome, ref utenti_prestito_cognome, ref giorno_preso, ref tempo_trattenuto,
                 ref codice_prestito, ref returns);

            break;

        case 2:

            ReturnBook.returnBook(ref libri_prestito,ref utenti_prestito_nome, ref
                utenti_prestito_cognome, ref giorno_preso, ref tempo_trattenuto, ref codice_prestito, ref
                returns);

            break;

        case 3:

            ShowBorrowesBooks.showBorrowedBooks(libri_prestito, utenti_prestito_nome,
                utenti_prestito_cognome, giorno_preso, tempo_trattenuto, codice_prestito, prestiti);

            break;

        case 4:

            Console.Clear();
            Console.WriteLine("Premi per ritornare al menu principale");

            break;

    }
```

3.2 TakeBorrow.cs

3.2.1 Step 1

In questo primo step andiamo a rimuovere dall'array dei libri e copie tutti i libri gia in prestito. Per poi controllare la presenza di libri o meno.

```
    Console.Clear();
    string[] strings = Titoli;
    int[] ints = copie;


    for (int i=0; i < libri_prestito.Length; i++)
    {
        for(int j = 0; j < strings.Length; j++)
        {
            if (libri_prestito[i] == null)
            {
                continue;
            }
            if (libri_prestito[i] == strings[j])
            {
                if (ints[j] > 1)
                {
                    ints[j]--;
                }
                else
                {
                    strings[j] = null;
                }
            }
        }
    }

    //controllo se ci sono libri disponibili
    bool flag = false;
    for (int i = 0; i < strings.Length; i++)
    {
        if (strings[i] != null)
        {
            flag = true;
            break;
        }
    }
    if (!flag)
    {
        Console.WriteLine("Non ci sono libri disponibili");
        Console.ReadKey();
        return;
    }
```

3.2.2 Step 2

Come step 2 l'utente seleziona tramite il menu quale libro prendere in prestito

```
    //Stampo i libri disponibili e faccio il menu per il prestito
    ConsoleKeyInfo key;
    int pos = 0;
    do
    {

        Menu(strings, pos);
        key = Console.ReadKey();

        //Freccie
        if (key.Key == ConsoleKey.DownArrow)
        {
            pos++;
            if (pos >= strings.Length)
            {
                pos = 0;
            }
            while (strings[pos] == null)
            {
                pos++;
                if (pos >= strings.Length)
                {
                    pos = 0;
                }
            }

        }
        if (key.Key == ConsoleKey.UpArrow)
        {
            pos--;
            if (pos < 0)
            {
                pos = strings.Length - 1;
            }
            while(strings[pos] == null)
            {
                pos--;
                if (pos < 0)
                {
                    pos = strings.Length - 1;
                }
            }

        }
    } while (key.Key != ConsoleKey.Enter);
```

3.2.3 Step 3

Come step 3 l'utente inserisce il proprio nome, cognome e la data di restituzione, e il programma genera un codice prestito. Il codice infine stampa i dati completi del prestito.

```
    //Aggiungo il libro preso in prestito
    string nome, cognome, giorno_restituisci;

    Console.WriteLine("Inserisci il tuo nome");
    nome = Console.ReadLine();
    Console.WriteLine("Inserisci il tuo cognome");
    cognome = Console.ReadLine();
    Console.WriteLine("Inserisci il giorno in cui devi restituire il libro, Inserisci GG/MM/AAAA");
    giorno_restituisci = Console.ReadLine();



    if (prestiti >= 2)
    {
        Array.Resize(ref libri_prestito, libri_prestito.Length + 1);
        Array.Resize(ref utenti_prestito_nome, utenti_prestito_nome.Length + 1);
        Array.Resize(ref utenti_prestito_cognome, utenti_prestito_cognome.Length + 1);
        Array.Resize(ref giorni_preso, giorni_preso.Length + 1);
        Array.Resize(ref tempo_trattenuto, tempo_trattenuto.Length + 1);
        Array.Resize(ref codice_prestito, codice_prestito.Length + 1);


    }
    libri_prestito[prestiti - returns] = strings[pos ];
    utenti_prestito_nome[prestiti - returns] = nome;
    utenti_prestito_cognome[prestiti - returns] = cognome;
    giorni_preso[prestiti - returns] = DateTime.Now.ToString("dd/MM/yyyy");
    tempo_trattenuto[prestiti - returns] = giorno_restituisci;
    codice_prestito[prestiti - returns] = prestiti;
    prestiti++;

    Console.WriteLine("Libro preso in prestito con successo");

    Console.WriteLine("Libro : " + libri_prestito[prestiti - 1 - returns]);
    Console.WriteLine("Utente : " + utenti_prestito_nome[prestiti - 1 - returns] + " " +
        utenti_prestito_cognome[prestiti - 1 - returns]);
    Console.WriteLine("Giorno preso : " + giorni_preso[prestiti - 1 - returns]);
    Console.WriteLine("Giorno da restituire : " + tempo_trattenuto[prestiti - 1 - returns]);
    Console.WriteLine("Codice prestito : " + codice_prestito[prestiti - 1 - returns]);

    Console.ReadKey();

    return;
```

3.3 ReturnBook.cs

3.3.1 Step 1

In questo primo step l'utente inserisce il proprio codice e il programma controlla la sua validitá.

```
    Console.Clear();
    Console.WriteLine("Inserisci il codice del prestito da restituire");
    int codice;
    try
    {
        codice = int.Parse(Console.ReadLine());
    }
    catch (Exception e)
    {
        Console.WriteLine("Errore: Inserire un numero");
        return;
    }

    int pos;
    //Flag codice non presente
    bool flag = false;
    for (pos = 0; pos < codice_prestito.Length; pos++)
    {
        if (codice_prestito[pos] == codice)
        {
            break;
        }
        if(pos == codice_prestito.Length - 1)
        {
            flag = true;
        }
    }

    if(flag)
    {
        Console.WriteLine("Codice non presente");
        Console.ReadKey();
        return;
    }
```

(Nota: I codici non sono stati descritti in quanto sono molto simili a quelli visti precedentemente)

3.3.2 Step 2

Step 2, stampare le informazioni del prestito e chiedere conferma.

```
    //Prestito restituito si/no menu
    ConsoleKeyInfo key;
    int pos3 = 0;
    do
    {
        Console.Clear();

        string[] strings =
        {
        "Nome: " + utenti_prestito_nome[pos],
        "Cognome: " + utenti_prestito_cognome[pos],
        "Libro: " + libro_prestito[pos],
        "Giorno preso: " + giorni_preso[pos],
        "Tempo trattenuto: " + tempo_trattenuto[pos],
        "Codice prestito: " + codice_prestito[pos],
        "Confermi la restituzione? "

        };
        int consoleWidth = Console.WindowWidth;
        for (int i = 0; i < strings.Length; i++)
        {
            int padding = (consoleWidth - strings[i].Length) / 2;
            if (padding < 0)
            {
                padding = 0;
            }
            Console.WriteLine(new string(' ', padding) + $"  {strings[i]}");
        }
        Menu(pos3);
        key = Console.ReadKey();
        if (key.Key == ConsoleKey.LeftArrow)
        {
            if (pos3 == 1)
                pos3--;

        }
        else if (key.Key == ConsoleKey.RightArrow)
        {
            if(pos3 == 0)
                pos3++;
        }
    } while (key.Key != ConsoleKey.Enter);
```

(Nota: I codici non sono stati descritti in quanto sono molto simili a quelli visti precedentemente)

3.3.3 Step 3

Data la conferma, il programma rimuove il prestito e stampa un messaggio di conferma.

```
if (pos3 == 1)
    {
        Console.WriteLine("Azione annullata");
        Console.ResetColor();
        Console.ReadKey();
        return;
    }
    else
    {
        //Restituzione libro
        libro_prestito[pos] = null;
        utenti_prestito_nome[pos] = null;
        utenti_prestito_cognome[pos] = null;
        giorni_preso[pos] = null;
        tempo_trattenuto[pos] = null;
        codice_prestito[pos] = 0;

        for (int i = pos; i < libro_prestito.Length - 1; i++)
        {
            libro_prestito[i] = libro_prestito[i + 1];
            utenti_prestito_nome[i] = utenti_prestito_nome[i + 1];
            utenti_prestito_cognome[i] = utenti_prestito_cognome[i + 1];
            giorni_preso[i] = giorni_preso[i + 1];
            tempo_trattenuto[i] = tempo_trattenuto[i + 1];
            codice_prestito[i] = codice_prestito[i + 1];
        }

        Array.Resize(ref libro_prestito, libro_prestito.Length - 1);
        Array.Resize(ref utenti_prestito_nome, utenti_prestito_nome.Length - 1);
        Array.Resize(ref utenti_prestito_cognome, utenti_prestito_cognome.Length - 1);
        Array.Resize(ref giorni_preso, giorni_preso.Length - 1);
        Array.Resize(ref tempo_trattenuto, tempo_trattenuto.Length - 1);
        Array.Resize(ref codice_prestito, codice_prestito.Length - 1);
        returns++;
        Console.WriteLine("Libro restituito con successo");
    }

    Console.ReadKey();
    return;
```

(Nota: I codici non sono stati descritti in quanto sono molto simili a quelli visti precedentemente)

3.4 ShowBorrowedBooks.cs

3.4.1 Step 1

Lo scopo di questa funzione è di visualizzare i libri in prestito. Vengono stampadi tutti i prestiti attivi al momento.

```
    Console.Clear();
    Console.WriteLine("Prestiti:");
    for (int i = 0; i < prestiti; i++)
    {
        if(libri_prestito.Length <= i)
        {
            break;
        }
        Console.WriteLine("Codice prestito: " + codice_prestito[i]);
        Console.WriteLine($"\tLibro: {libri_prestito[i]}");
        Console.WriteLine($"\tUtente: {utenti_prestito_nome[i]} {utenti_prestito_cognome[i]}");
        Console.WriteLine($"\tGiorno preso: {giorni_preso[i]}");
        Console.WriteLine($"\tTempo trattenuto: {tempo_trattenuto[i]}");
        Console.WriteLine();
    }
    Console.WriteLine("Premi un tasto per tornare al menu");
    Console.ReadKey();
```

4 Utils e Logo

4.1 ToLower.cs

Una funzione gia esistente in C# che trasforma una stringa in minuscolo, ma per alcuni bug ne abbiamo creato una noi.

```
1     public static string ToLowerString(string str)
2     {
3         //Converte la stringa in minuscolo
4         string lower = "";
5         for(int i = 0; i < str.Length; i++)
6         {
7             //Le lettere maiuscole vengono convertite in minuscole
8             if (str[i] >= 'A' && str[i] <= 'Z')
9             {
10                lower += str[i].ToString().ToLower();
11            }
12            else
13            {
14                lower += str[i];
15            }
16
17        }
18        return lower;
19    }
```

4.2 LogoBiblioteca.cs —— LogoPrestito.cs

Logo della biblioteca e Prestito fatte tramite un sito TopSecret

(Nota: Il codice dei loghi sono reperibili su github!)

Figura 1: Logo biblioteca.

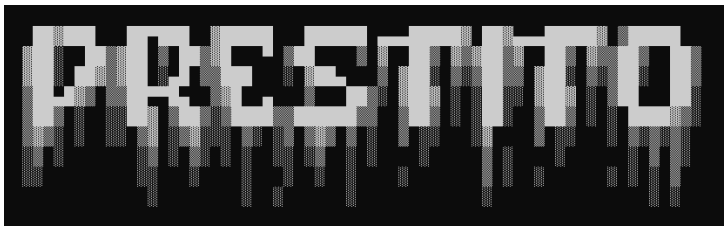

Figura 2: Logo Prestito.

5 Git

Git é un programma di controllo creato per gestire progetti anche di grande dimensione in modo efficiente Git permette ai sviluppatori di poter lavorare a versioni diverse di programma contemporaneamente

Come funziona?

- **Area di Lavoro**
  - Directory locale sul tuo pc dove lavori al Progetto
  - Puoi aggiungere, Eliminare , Modificare i file
- **Area di preparazione**
  - Zona di decisione sul cosa caricare, infatti puoi caricare dei file specifici
  - Questo ti permette di salvare i cambiamenti definitivi
- **Repository**
  - La Repository contiene tutte le versioni del tuo programma
  - Puo essere salvato in locale (Sul tuo pc) o su cloud (Github)

Caratterestiche principali di Git

- **Distribuito**
  - La repository contiene tutti la cronologia del progetto, rendendo Git piu stabile dato che non è gestito da un server
- **Leggero**
  - Git è progettato per essere veloce e gestire grandi progetti
- **Risoluzione dei conflitti**
  - Se 2 utenti modificano nello stesso punto del codice creando un conflitto, git permette di risolvere la questione manualmente
- **Sicuro**
  - Una volta salvati vengono criptati

Git non ha una GUI, e funziona tutto di codici nel Prompt dei Comandi di Git

6 GitHub

Github è una piattaforma online che va a usare il funzionamento di Git, dando possibilità di salvare i file in cloud. Offre anche strumenti avanti per il lavoro di squadra.

Perché usarlo?

- **Collaborazione Globale**
  - Gli sviluppatori possono lavorare al codice in open-source o privato, da tutto il mondo
- **Integrazione con Git**
  - Unisce le funzionalitá di git con un interfaccia facile da usare

Come usarlo

- **Commit**
  - La funzione Commit è il punto dove salvi in locale il codice, in un punto di controllo. Sarebbe come scattare una foto dello stato attuale del Progetto
- **Push**
  - La funzione Push serve a caricare il codice in commit al cloud su github. Serve a caricare le modifiche fatte localmente e unirle al progetto su github
- **Pull**
  - La funzione Pull serve invece caricare il progetto dal cloud al pc locale aggiornando le modifiche fatte dagli altri collaboratori

7 Conclusioni

Il progetto ci ha richiesto molto tempo. Abbiamo deciso di creare un Menu che funziona direttamente con la tastiera, per differenziarci dagli altri progetti, senza tralasciare nessun punto.
Abbiamo avuto problemi con il tempo per finire la presentazione ma apparte quello nessun problema Abbiamo imparato nuove informazioni, e speriamo sia utile anche per gli altri.

Github

- Codice Sorgente