

4IT Erasmus+Project 2024-1-RO01-KA121-VET-000214264
Beneficiary Colegiul National Gheorghe Vranceanu, Bacau, Romania

Sending organization: Colegiul National ”Roman-Vodă” Roman, Romania

MOBILITY PORTFOLIO
Paphos, Cyprus

Mobility period:
PAPHOS, CYPRUS, 13.04.2025-27.04.2025 – practice period 14.04.2025-25.04.2025

Receiving

Practice tutor: Lazaros Lazarou

Participant: Murariu Cosmin-Andrei

DISCLAIMER: The European Commission’s support for the production of this publication does not constitute an endorsement of the contents, which reflect the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Cosmin’s Insight - Site for selection


Cosmin's Insight
Pagina Principala
Galerie de imagini
Despre mine
Contact
Pagina Principală
„Suntem aici pentru a face o gaură în univers. Altfel, de ce să mai fim aici?” - Steve Jobs


Cosmin's Insight
Pagina Principala
Galerie de imagini
Despre mine
Contact
Cosmin's Insight
Acest nume este inspirat din ideea de cunoaștere, de perspicacitate și este o reprezentare a personalității mele. De asemenea, logoul reprezintă creativitatea și inovația, valori pe care le susțin.
Îmi doresc din suflet să răspândesc ideea de inovație în rândul adolescenților, noi fiind cei care putem aduce schimbări radicale viitorului societății.
COSMIN'S
INSIGHT
FOR A BETTER FUTURE
Ce mă motivează să particip?
Principalul motiv pentru care îmi doresc să particip la acest proiect este oportunitatea de formare profesională. Experiența pe care o voi avea la finalul proiectului va fi de folos în viitorul apropiat, deoarece doresc să am un loc de muncă în domeniul IT.

Cosmin's Insight

Pagina Principala | Galerie de imagini | Despre mine | Contact

La ce proiecte am mai participat?

De-a lungul timpului am participat la diferite proiecte și activități, precum *"Competiția Națională Arduino"*, unde împreună cu doi colegi am realizat un sistem de securitate pentru școli, folosind platforma **Arduino** pentru dezvoltarea componentei fizice a proiectului și **HTML, CSS, PHP, MySQL** pentru crearea unui website necesar și a unei baze de date simple.





Proiectul 1 Iunie

De asemenea am participat la proiectul *"1 Iunie"*, realizat de asociația People To People International Roman Chapter, în calitate de **voluntar**.

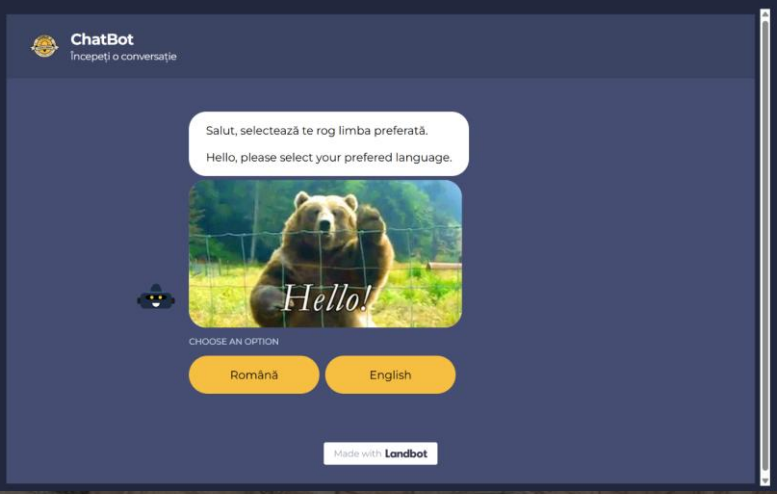
La acest proiect voluntarii asociației au organizat diferite activități pentru copiii de toate vârstele din orașul Roman, proiectul fiind desfășurat în aer liber.

Copiii au participat la activități precum **pictat pe față, cursă cu saci, volei, fotbal și trasee de viteză.**

Cosmin's Insight

Pagina Principala | Galerie de imagini | Despre mine | Contact

ChatBot realizat cu aplicația LandBot



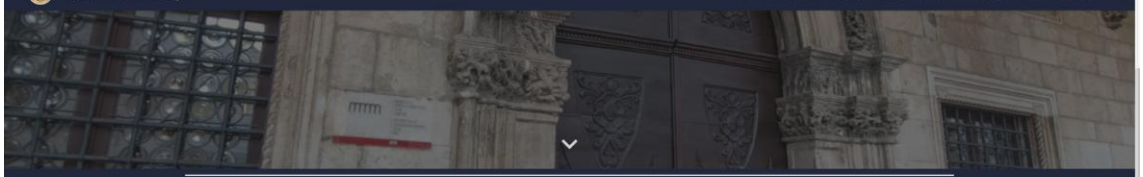
Cosmin's Insight

Pagina Principala | Galerie de imagini | Despre mine | Contact

Galerie de imagini

Cosmin's Insight

Pagina Principala **Galerie de imagini** Despre mine Contact





Fotografii din România

Această galerie conține fotografii de pe teritoriul României.

Printre primele fotografii apar locuri precum Mănăstirea Bârsana, Lacul Roșu, și Mănăstirea Almaș.

În următoarele fotografii apar fotografii din zona Romanului, capturând momentele ale zilei din anotimpuri diferite.

Cosmin's Insight

Pagina Principala **Galerie de imagini** Despre mine Contact

Fotografii din călătorii

Această galerie conține fotografii din Dubrovnik, Croația.

Prima fotografie reprezintă Catedrala Romano-Catolică din Dubrovnik, situată în inima orașului vechi și este un exemplu de arhitectură barocă.

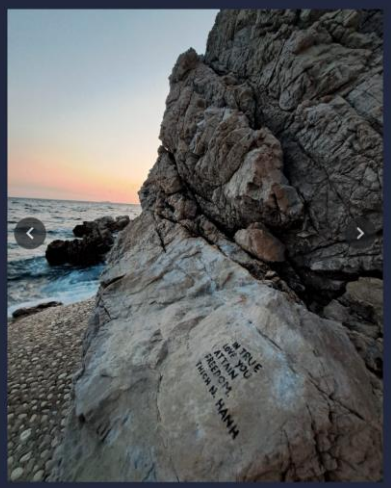
Apoi urmează fotografii din cadrul Tvrđava Minčeta, una dintre cele mai emblematice fortărețe ale orașului Dubrovnik. Situată în colțul nordic al zidurilor orașului vechi, Minčeta este un simbol al rezistenței și al protecției orașului împotriva invaziilor.

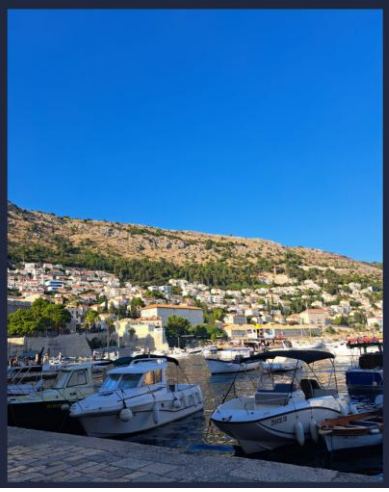


Cosmin's Insight

Pagina Principala **Galerie de imagini** Despre mine Contact

Alte Fotografii





Cosmin's Insight

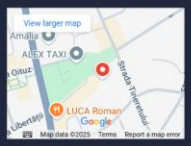
Pagina Principala **Galerie de imagini** Despre mine Contact

Imagini Vectoriale

Această galerie conține imaginile vectoriale create pentru acest proiect cu ajutorul aplicației InkScape.

Deoarece Google Sites nu suportă adăugarea de imagini vectoriale în "caruselele de imagini" sau în secțiunea logo, aceste imagini sunt adăugate în format ".png".

Imaginile vectoriale sunt adăugate pe pagina principală și pe pagina "Despre Mine", sub formă de cod XML.



Colegiul Național "Roman-Vodă"

Str. Mihai Eminescu Nr. 4, Loc. Roman, jud.Neamt,

Cod postal 611028, Romania

cnrv.ro

Cosmin's Insight

Pagina Principala Galerie de imagini **Despre mine** Contact

Despre mine

Cosmin's Insight

Pagina Principala Galerie de imagini **Despre mine** Contact

Motto-ul pe care îl urmez

„Suntem aici pentru a face o gaură în univers. Altfel, de ce să mai fim aici?” - ***Steve Jobs***



Cine sunt eu?

Numele meu este **Murariu Cosmin Andrei** și sunt elev în clasa a 11-a la ***Colegiul Național "Roman-Vodă"*** din orașul Roman, județul Neamț. De la o vârstă fragedă am dezvoltat o pasiune puternică pentru tehnologie, viața mea învârtindu-se în jurul acesteia.

De ce am ales să particip la proiect?



Am ales să particip la acest proiect din mai multe motive, principalul fiind ***experiența profesională*** pe care acesta mi-o poate oferi. Un alt motiv este posibilitatea de a-mi dezvolta ***abilitățile de comunicare*** și de ***lucru în echipă***, acestea fiind necesare pentru un viitor loc de muncă în domeniul IT.



Ce valori promovez?

Mereu am fost o persoană empatică, deschisă și prietenoasă. Astfel, valorile promovate de mine sunt ***integritatea***, ***empatia***, ***respectul*** și ***gratitudinea***. Din rândul valorilor profesionale mă regăsesc în ***transparență***, ***colaborare*** și ***excelență***.

Cosmin's Insight
Pagina Principala Galerie de imagini **Despre mine** Contact

Care sunt pasiunile mele?

Fiind un elev pe profil real, pasiunile mele sunt stereotipice în mare parte. Printre ele se numără **informatica**, **matematica** și **cinematografia**. Acestea sunt descrise mai jos, prin grupurile ce se pot restrânge.



Informatica

Informatica este o pasiune pe care mi-am însușit-o încă de la o vârstă fragedă, mereu având o atracție spre acest domeniu. De-a lungul timpului am căpătat cunoștințe în arii precum rețelistica, dezvoltarea web, IoT(Internet of Things) și ingineria hardware.

Acest domeniu este în continuă dezvoltare, așadar încerc mereu să rămân la curent cu cele mai noi tehnologii și aplicații, astfel dezvoltându-mi abilitățile informatice într-un ritm spontan.

Matematica

Cosmin's Insight
Pagina Principala Galerie de imagini **Despre mine** Contact

Matematica

Matematica a avut mereu un loc special în viața mea, fiind singura știință abstractă care nu are în cele mai multe situații corespondențe în viața cotidiană. Aceasta se bazează pe concepte pure și necesită un efort intelectual ridicat.

Rezolvarea problemelor la matematică este o activitate satisfăcătoare, care îmi dezvoltă procesele cognitive mai mult decât orice altă știință.

Cinematografia

Cinematografia, în opinia mea, este cea mai frumoasă formă de artă contemporană. Ea a evoluat împreună cu tehnologia, în prezent echipamentele de producție video fiind atât de complexe încât o persoană obișnuită ar crede că sunt dintr-o altă lume.

Am dezvoltat această pasiune vizionând de-a lungul timpului o mulțime vastă de filme și seriale, unele vizionându-le în premieră direct la cinema. Sunt la curent cu producțiile ce urmează să fie difuzate și urcate pe platformele de streaming, urmărind diverse pagini de știri în acest domeniu.

Cosmin's Insight

Pagina Principala Galerie de imagini **Despre mine** Contact

Persoane care mă inspiră

Câteva dintre persoanele care mă inspiră sunt personalităţi precum ***Steve Jobs***, prin discursurile lui şi prin creaţiile sale, ***Mark Zuckerberg*** pentru impactul pe care l-a adus socializării online şi ***Jen-Hsun Huang***, CEO-ul NVIDIA, care a revoluţionat semnificaţia inteligenţei artificiale.





Colegiul Naţional "Roman-Vodă"

Str. Mihai Eminescu Nr. 4, Loc. Roman, jud.Neamt,

Cod postal 611028, Romania

cnrv.ro

Cosmin's Insight

Pagina Principala Galerie de imagini Despre mine **Contact**

Contact

Click pe logo pentru redirecţionare

Cosmin's Insight

Pagina Principala Galerie de imagini Despre mine **Contact**

Facebook

facebook.com/cosmin.murariu.77

Instagram

@its.cozmin

WhatsApp

+40 726 870 332

Email

Adresă liceu

Applications created during the mobility using C++

14.04.2025

```
#include <iostream>

using namespace std;

/*
   Write a program where you will use a dynamic array to save characters.
   the user will give you numbers(integers) that range from 65 to 90. as soon as
   you read the number you will convert the character and save it into a dynamic
   character array. your program will be able to make the array bigger if it is full
   at the end you must display the result
*/

int main()
{
   int arrSize = 0; // Declare the size of our dynamic array
   int n = 0, tempN = 0;  // n is the counter for our array, tempN is the number the user inputs if
                          //we need more space in the array
   int input = 0; // The input we read every time from the user
   char* characterArray = NULL, * tempArray = NULL;
   // We declare our arrays, tempArray is the array used for a size bigger than the initial one
   cout << "How many ASCII codes would you like to enter?: ";
   cin >> arrSize; // We read the initial array size
   characterArray = new char[arrSize]; // Dynamically create the initial character array
   do
   {
      cout << "Enter a number between 65 and 90 at position " << n << " ->";
      // Prompt the user  to input a number based on the specified criteria
      cin >> input;  // Read the user input
      if (input < 65 || input>90) // Checks if the user's input is not valid
      {
         cout << "That value is not valid. Please enter a value between 65 and 90:";
          // Prompt the user that the value is not valid and asks for a correct one
         cin >> input; // Read the value again
      }
      else  // If the value is valid
      {
         characterArray[n] = (char)input; // Save the value as a character (type casting) in the
array
         n++; // Increase the counter
```

```
            cout << "Saved!" << '\n';  // Prompt the user that we successfully save the input
            if (n >= arrSize) // We check if we ran out of space, if so we grow the array
            {
               cout << "Out of space! How many more numbers would you like to enter?(0 to exit)";
             // Alert the user that they ran of space and ask them if they want more numbers
               cin >> tempN;
              // Read the number of characters that the user needs more of, 0 if we want to exit
               if (tempN > 0) // If we did not exit
               {
                  arrSize = arrSize + tempN; // Increase the size of the original array
                  tempArray = new char[arrSize]; // Create the new auxiliary array
                  for (int i = 0; i < n; i++) // Copy the contents of the old array in the new created one
                  {
                     tempArray[i] = characterArray[i];
                  }
                  delete[] characterArray; // Free the memory that the old array was using
                  characterArray = tempArray; // Point the original array at the auxiliary one
               }
            }
         }
      } while (n < arrSize); // While the user wants to input ASCII codes

      cout << "The array content will be printed now: "; // Print the content of the character array
      for (int i = 0; i < arrSize; i++) // We loop through the array and print each character
      {
         cout << characterArray[i] << " ";
      }
      delete[] characterArray; // Free the memory
      return 0;
   }
```

15.04.2025

```
#include <iostream>
#include <string>
using namespace std;

/*
   Collect information about students. collect name, telephone number, use 2d arrays to do
that.
         - Search for the phone number for a name given by the user.
   Convert the above program in order to be able to not only read information but also:
         - search for a name and phone number, do it continuously using while
         - make the program search for a phone number and gives the name
*/

void searchName(string *nameArr, string *phoneArr, int arrSize)
{
   bool found = false;
   string name = "0";
   do
   {
      cout << "Enter the name of the student you want the phone number for,  "exit" to leave: ";
      cin >> name; // Read the name so we can search for its phone number

      for (int i = 0; i < arrSize && found == false; i++)
      // loop through the names array and stop if we found the given name
      {
         if (nameArr[i] == name)
         {
            found = true;
            // a breakpoint if we find the name
            cout << "The phone number of " << name << " is " << phoneArr[i] << '\n';
            // print the phone number of the student
         }
      }

      if (found == false)
         cout << "The student cannot be found \n";
           // error message if the name is not in the name array
   } while (name != "exit");
}

void searchPhone(string *nameArr, string *phoneArr, int arrSize)
{
   bool found = false;
```

```
    string phone;

    do
    {
        cout << "Enter the number of the student you want to find, type exit to leave: ";
        cin >> phone;
        for (int i = 0; i < arrSize && found == false; i++)
        {
            if (phoneArr[i] == phone)
            {
                found = true;
                cout << "The name is " << nameArr[i] << '\n';
            }
        }

        if (found == false)
            cout << "The student cannot be found";
    } while (phone != "exit");
}

void showStudents(string *nameArr, string *phoneArr, int arrSize)
{
    cout << "Now printing the names and phone numbers of the students ---------- \n";
    for (int i = 0; i < arrSize; i++)
    {
        cout << "Name: " << nameArr[i] << " Phone number: " << phoneArr[i] << "\n";
    }
    cout << "--------------------------------------------------------------------------\n";
}

int main()
{
    int arrSize = 0;
    string *nameArr = nullptr;
    string *phoneArr = nullptr;
    string *tempName = nullptr;
    string *tempPhone = nullptr;
    int n = 0, tmpN = 0;

    cout << "How many students do you want to enter?: ";
    cin >> arrSize;
    nameArr = new string[arrSize]; // creating the 2d dynamic arrays
    phoneArr = new string[arrSize];

    do
    { // dynamically grow the arrays
        cout << "The name of the student: ";
```

```
    cin >> nameArr[n];
    cout << "The phone number of the student: ";
    cin >> phoneArr[n];
    n++;
    if (n >= arrSize)
    {
        cout << "Out of space! How many more student do you want to enter?(0 to exit): ";
        cin >> tmpN;
        if (tmpN > 0)
        {
            arrSize = arrSize + tmpN;
            tempName = new string[arrSize];
            tempPhone = new string[arrSize];
            for (int i = 0; i < n; i++)
            {
                tempName[i] = nameArr[i];
                tempPhone[i] = phoneArr[i];
            }
            delete[] nameArr;
            delete[] phoneArr;
            nameArr = tempName;
            phoneArr = tempPhone;
        }
    }
} while (n < arrSize);
cout << "What do you want to do now?\n";
int option = 0;
do
{
    cout << "1.Show all the students\n";
    cout << "2.Search a student by their name\n";
    cout << "3.Search a student by their phone number\n";
    cout << "0 to exit\n";
    cin >> option;
    switch (option)
    {
    case 1:
        showStudents(nameArr, phoneArr, arrSize);
        break;
    case 2:
        searchName(nameArr, phoneArr, arrSize);
        break;
    case 3:
        searchPhone(nameArr, phoneArr, arrSize);
        break;
    default:
        break;
```

```
        }
    } while (option != 0);
    // showStudents(nameArr,phoneArr,arrSize);
    // searchName(nameArr, phoneArr, arrSize);
    // function that takes a name for the user and prints out the phone number
    // searchPhone(nameArr, phoneArr, arrSize);
    delete[] nameArr; // free the memory
    delete[] phoneArr;
    return 0;
}
```

16.04.2025

```
#include <iostream>
#include <string>
using namespace std;

// Homework exercise 1

class car
{
private:
   string brand;
   string model;
   string plateNumber;
   string ownerName;

public:
   car() // initialize with default values
   {
      brand = "EMPTY";
      model = "EMPTY";
      plateNumber = "XX 00 XXX";
      ownerName = "EMPTY";
   }
   void setBrand(string input) { this->brand = input; }
   void setModel(string input) { this->model = input; }
   void setPlateNumber(string input) { this->plateNumber = input; }
   void setOwnerName(string input) { this->ownerName = input; }
   string getBrand() { return brand; }
   string getModel() { return model; }
   string getPlateNumber() { return plateNumber; }
   string getOwnerName() { return ownerName; }
};

void addMultipleCars(car*& carsArr, int &capacity, int &count)
{
   int n;
   cout << "How many cars do you want to add? ";
   cin >> n;
   cin.ignore(); // clear newline from the input buffer

   // Check if current capacity is enough. If not, create a temporary array with extra space.
   if(count + n > capacity)
   {
```

```
        int newCapacity = capacity;

        while(count + n > newCapacity)
        {
            newCapacity += n;
        }
        car* temp = new car[newCapacity];
        for (int i = 0; i < count; i++)
        {
            temp[i] = carsArr[i];
        }
        delete [] carsArr;
        carsArr = temp;
        capacity = newCapacity;
        cout << "Array capacity increased to " << capacity << "." << endl;
    }

    // Input the new cars using getline to capture full inputs (including spaces)
    for (int i = 0; i < n; i++)
    {
        cout << "\nEntering details for car " << count + 1 << ":" << endl;
        string input;

        cout << "Brand: ";
        getline(cin, input);
        carsArr[count].setBrand(input);

        cout << "Model: ";
        getline(cin, input);
        carsArr[count].setModel(input);

        cout << "Plate Number (e.g., ABC 123): ";
        getline(cin, input);
        carsArr[count].setPlateNumber(input);

        cout << "Owner's Name: ";
        getline(cin, input);
        carsArr[count].setOwnerName(input);
        count++;
        cout << "------------------------" << endl;
    }
}

void outputCars(car* carsArr, int count)
{
    cout << "\nPrinting all cars:" << endl;
    for (int i = 0; i < count; i++)
```

```
    {
        cout << "Car " << i + 1 << ":" << endl;
        cout << "Brand: " << carsArr[i].getBrand() << endl;
        cout << "Model: " << carsArr[i].getModel() << endl;
        cout << "Plate Number: " << carsArr[i].getPlateNumber() << endl;
        cout << "Owner's Name: " << carsArr[i].getOwnerName() << endl;
        cout << "------------------------" << endl;
    }
}

int main()
{
    int capacity = 5;
    int count = 0;
    car* myCars = new car[capacity];
    int choice = -1;

    do
    {
        cout << "\nCar Management Menu" << endl;
        cout << "1. Add car information" << endl;
        cout << "2. Display all cars" << endl;
        cout << "0. Exit" << endl;
        cout << "Enter your choice: ";
        cin >> choice;
        cin.ignore(); // clear newline

        switch (choice)
        {
            case 1:
                addMultipleCars(myCars, capacity, count);
                break;
            case 2:
                outputCars(myCars, count);
                break;
            case 0:
                cout << "Exiting the program." << endl;
                break;
            default:
                cout << "Invalid choice. Please try again." << endl;
        }
    } while (choice != 0);

    delete [] myCars;
    return 0;
}
```

16.04.2025

```
#include <iostream>
#include <string>

using namespace std;

// Homework exercise 2

/*
-EXERCISE
Write a program for a supermarket which is going to
take the information for the different products a supermarket sells
and it will save the following information:
        -Brand of product
        -Type of product
        -Price
        -Stock Remaining
        -Barcode

Your program must be able to search for products by name and by Type
*/

class product
{
private:
   string brand;
   string type;
   int price;
   int stockRemaining;
   string barCode;

public:
   product()
   {
      brand = "EMPTY";
      type = "EMPTY";
      price = 0;
      stockRemaining = 0;
      barCode = "EMPTY";
   }
   void setBrand(string input) { this->brand = input; }
   void setType(string input) { this->type = input; }
   void setPrice(int input) { this->price = input; }
```

```
void setStockRemaining(int input) { this->stockRemaining = input; }
void setBarCode(string input) { this->barCode = input; }
string getBrand() { return brand; }
string getType() { return type; }
int getPrice() { return price; }
int getStockRemaining() { return stockRemaining; }
string getBarCode() { return barCode; }

// Function to search for a product by name
static void searchByName(product products[], int size, string name)
{
   bool found = false;
   for (int i = 0; i < size; i++)
   {
      if (products[i].getBrand() == name)
      {
         cout << "Product found: " << endl;
         cout << "Brand: " << products[i].getBrand() << endl;
         cout << "Type: " << products[i].getType() << endl;
         cout << "Price: " << products[i].getPrice() << endl;
         cout << "Stock Remaining: " << products[i].getStockRemaining() << endl;
         cout << "Barcode: " << products[i].getBarCode() << endl;
         found = true;
         break;
      }
   }
   if (!found)
   {
      cout << "Product with name '" << name << "' not found." << endl;
   }
}

// Function to search for products by type
static void searchByType(product products[], int size, string type)
{
   bool found = false;
   for (int i = 0; i < size; i++)
   {
      if (products[i].getType() == type)
      {
         if (!found)
         {
            cout << "Products with type '" << type << "':" << endl;
         }
         cout << "-----------------------" << endl;
         cout << "Brand: " << products[i].getBrand() << endl;
         cout << "Type: " << products[i].getType() << endl;
```

```
            cout << "Price: " << products[i].getPrice() << endl;
            cout << "Stock Remaining: " << products[i].getStockRemaining() << endl;
            cout << "Barcode: " << products[i].getBarCode() << endl;
            found = true;
        }
      }
      if (!found)
      {
        cout << "No products with type '" << type << "' found." << endl;
      }
    }
};

int main()
{
    const int MAXPRODUCTS = 1;
    product myProducts[MAXPRODUCTS];
    for (int i = 0; i < MAXPRODUCTS; i++)
    {
      string tempStr = "";
      int tempInt = 0;

      cout << "Brand " << i << ": ";
      cin >> tempStr;
      myProducts[i].setBrand(tempStr);

      cout << "Type " << i << ": ";
      cin >> tempStr;
      myProducts[i].setType(tempStr);

      cout << "Price " << i << ": ";
      cin >> tempInt;
      myProducts[i].setPrice(tempInt);

      cout << "Stock Remaining " << i << ": ";
      cin >> tempInt;
      myProducts[i].setStockRemaining(tempInt);

      cout << "Barcode " << i << ": ";
      cin >> tempStr;
      myProducts[i].setBarCode(tempStr);
    }

    cout << "Now searching for products: " << endl;

    string searchName, searchType;
    cout << "Enter a product name to search: ";
```

```
    cin >> searchName;
    product::searchByName(myProducts, MAXPRODUCTS, searchName);

    cout << "Enter a product type to search: ";
    cin >> searchType;
    product::searchByType(myProducts, MAXPRODUCTS, searchType);

    return 0;
}
```

17.04.2025

```
#include <iostream>
#include <string>
using namespace std;

/*
Break the program into functions with a menu of options:
        1. Add car information (add multiple cars at once)
        2. Display all cars
        3. Search by brand
        4. Search by model
        5. Search by owner
        6. Search by plate number
        0. Exit

The program uses a dynamic (growing) array of cars.
*/

class car
{
private:
   string brand;
   string model;
   string plateNumber;
   string ownerName;

public:
   car() // initialize with default values
   {
      brand = "EMPTY";
      model = "EMPTY";
      plateNumber = "EMPTY";
      ownerName = "EMPTY";
   }
   void setBrand(string input) { this->brand = input; }
   void setModel(string input) { this->model = input; }
   void setPlateNumber(string input) { this->plateNumber = input; }
   void setOwnerName(string input) { this->ownerName = input; }
   string getBrand() { return brand; }
   string getModel() { return model; }
   string getPlateNumber() { return plateNumber; }
   string getOwnerName() { return ownerName; }
```

```
};

void addMultipleCars(car*& carsArr, int &capacity, int &count)
{
   int n;
   cout << "How many cars do you want to add? ";
   cin >> n;

   // Check if current capacity is enough. If not, create a temporary array with extra space.
   if(count + n > capacity)
   {
      int newCapacity = capacity;
      while(count + n > newCapacity)
      {
         newCapacity += n;
      }
      car* temp = new car[newCapacity];
      for (int i = 0; i < count; i++)
      {
         temp[i] = carsArr[i];
      }
      delete [] carsArr;
      carsArr = temp;
      capacity = newCapacity;
      cout << "Array capacity increased to " << capacity << "." << endl;
   }

   // Input the new cars
   for (int i = 0; i < n; i++)
   {
      cout << "\nEntering details for car " << count + 1 << ":" << endl;
      string temp, temp2;

      cout << "Brand: ";
      cin >> temp;
      carsArr[count].setBrand(temp);

      cout << "Model: ";
      cin >> temp;
      carsArr[count].setModel(temp);

      cout << "Plate Number (enter two parts, e.g., ABC 123): ";
      cin >> temp >> temp2;
      temp = temp + " " + temp2;
      carsArr[count].setPlateNumber(temp);

      cout << "Owner's Name: ";
```

```
        cin >> temp;
        carsArr[count].setOwnerName(temp);

        count++;
        cout << "-----------------------" << endl;
    }
}

void outputCars(car* carsArr, int count)
{
    cout << "\nPrinting all cars:" << endl;
    for (int i = 0; i < count; i++)
    {
        cout << "Car " << i + 1 << ":" << endl;
        cout << "Brand: " << carsArr[i].getBrand() << endl;
        cout << "Model: " << carsArr[i].getModel() << endl;
        cout << "Plate Number: " << carsArr[i].getPlateNumber() << endl;
        cout << "Owner's Name: " << carsArr[i].getOwnerName() << endl;
        cout << "-----------------------" << endl;
    }
}

void searchByBrand(car* carsArr, int count)
{
    string searchBrand;
    cout << "Enter the brand you want to search: ";
    cin >> searchBrand;
    bool found = false;
    cout << "\nSearch results for brand: " << searchBrand << endl;
    for (int i = 0; i < count; i++)
    {
        if(carsArr[i].getBrand() == searchBrand)
        {
            cout << "Car " << i + 1 << ":" << endl;
            cout << "Brand: " << carsArr[i].getBrand() << endl;
            cout << "Model: " << carsArr[i].getModel() << endl;
            cout << "Plate Number: " << carsArr[i].getPlateNumber() << endl;
            cout << "Owner's Name: " << carsArr[i].getOwnerName() << endl;
            cout << "-----------------------" << endl;
            found = true;
        }
    }
    if (!found)
    {
        cout << "No car with brand " << searchBrand << " was found." << endl;
    }
}
```

```
void searchByModel(car* carsArr, int count)
{
   string searchModel;
   cout << "Enter the model you want to search: ";
   cin >> searchModel;
   bool found = false;
   cout << "\nSearch results for model: " << searchModel << endl;
   for (int i = 0; i < count; i++)
   {
      if(carsArr[i].getModel() == searchModel)
      {
         cout << "Car " << i + 1 << ":" << endl;
         cout << "Brand: " << carsArr[i].getBrand() << endl;
         cout << "Model: " << carsArr[i].getModel() << endl;
         cout << "Plate Number: " << carsArr[i].getPlateNumber() << endl;
         cout << "Owner's Name: " << carsArr[i].getOwnerName() << endl;
         cout << "-----------------------" << endl;
         found = true;
      }
   }
   if (!found)
   {
      cout << "No car with model " << searchModel << " was found." << endl;
   }
}

void searchByOwner(car* carsArr, int count)
{
   string searchOwner;
   cout << "Enter the owner's name you want to search: ";
   cin >> searchOwner;
   bool found = false;
   cout << "\nSearch results for owner: " << searchOwner << endl;
   for (int i = 0; i < count; i++)
   {
      if(carsArr[i].getOwnerName() == searchOwner)
      {
         cout << "Car " << i + 1 << ":" << endl;
         cout << "Brand: " << carsArr[i].getBrand() << endl;
         cout << "Model: " << carsArr[i].getModel() << endl;
         cout << "Plate Number: " << carsArr[i].getPlateNumber() << endl;
         cout << "Owner's Name: " << carsArr[i].getOwnerName() << endl;
         cout << "-----------------------" << endl;
         found = true;
      }
```

```
    }

    if (!found)
    {
        cout << "No car with owner name " << searchOwner << " was found." << endl;
    }
}

void searchByPlate(car* carsArr, int count)
{
    string searchPlate;
    cout << "Enter the plate number you want to search (e.g., ABC 123): ";
    getline(cin >> ws, searchPlate);
    bool found = false;
    cout << "\nSearch results for plate number: " << searchPlate << endl;
    for (int i = 0; i < count; i++)
    {
        if(carsArr[i].getPlateNumber() == searchPlate)
        {
            cout << "Car " << i + 1 << ":" << endl;
            cout << "Brand: " << carsArr[i].getBrand() << endl;
            cout << "Model: " << carsArr[i].getModel() << endl;
            cout << "Plate Number: " << carsArr[i].getPlateNumber() << endl;
            cout << "Owner's Name: " << carsArr[i].getOwnerName() << endl;
            cout << "-----------------------" << endl;
            found = true;
        }
    }
    if (!found)
    {
        cout << "No car with plate number " << searchPlate << " was found." << endl;
    }
}

int main()
{
    int capacity = 0;
    int count = 0;
    car* myCars = new car[capacity];
    int choice = -1;
    do
    {
        cout << "\nCar Management Menu" << endl;
        cout << "1. Add car information" << endl;
        cout << "2. Display all cars" << endl;
        cout << "3. Search cars by brand" << endl;
        cout << "4. Search cars by model" << endl;
```

```
        cout << "5. Search cars by owner" << endl;
        cout << "6. Search cars by plate number" << endl;
        cout << "0. Exit" << endl;
        cout << "Enter your choice: ";
        cin >> choice;
        switch (choice)
        {
            case 1:
                addMultipleCars(myCars, capacity, count);
                break;
            case 2:
                outputCars(myCars, count);
                break;
            case 3:
                searchByBrand(myCars, count);
                break;
            case 4:
                searchByModel(myCars, count);
                break;
            case 5:
                searchByOwner(myCars, count);
                break;
            case 6:
                searchByPlate(myCars, count);
                break;
            case 0:
                cout << "Exiting the program." << endl;
                break;
            default:
                cout << "Invalid choice. Please try again." << endl;
        }
    } while (choice != 0);

    delete [] myCars;
    return 0;
}
```

18.04.2025

```
#include <iostream>
#include <string>

using namespace std;

class Product
{
private:
   string brand;
   string type;
   float price;
   int stockRemaining;
   string barcode;

public:
   Product()
   {
      brand = "EMPTY";
      type = "EMPTY";
      price = 0.0;
      stockRemaining = 0;
      barcode = "EMPTY";
   }
   void setBrand(string B) { brand = B; }
   void setType(string T) { type = T; }
   void setPrice(float P) { price = P; }
   void setStockRemaining(int S) { stockRemaining = S; }
   void setBarcode(string B) { barcode = B; }
   string getBrand() { return brand; }
   string getType() { return type; }
   float getPrice() { return price; }
   int getStockRemaining() { return stockRemaining; }
   string getBarcode() { return barcode; }
};

void addProducts(Product *&prodArray, int &capacity, int &count)
{
   int n;
   cout << "How many products would you like to add? ";
   cin >> n;

   // Grow array if needed by asking the user how many extra spots to allocate.
   if (count + n > capacity)
```

```
  {
    cout << "Not enough space. Currently capacity is " << capacity << " and " << count << "
products stored." << endl;
    cout << "Enter additional capacity to allocate (must be at least " << (count + n - capacity) <<
"): ";
    int extra;
    cin >> extra;
    int newCapacity = capacity + extra;
    if (newCapacity < count + n) {
      newCapacity = count + n; // Ensure capacity is enough
    }
    Product *temp = new Product[newCapacity];
    for (int i = 0; i < count; i++)
    {
      temp[i] = prodArray[i];
    }
    delete[] prodArray;
    prodArray = temp;
    capacity = newCapacity;
    cout << "Product array capacity increased to " << capacity << "." << endl;
  }
  // Input product details
  for (int i = 0; i < n; i++)
  {
    string inputStr;
    float inputPrice;
    int inputStock;
    cout << "\nEntering details for product " << count + 1 << ":" << endl;

    cout << "Product Brand: ";
    cin >> inputStr;
    prodArray[count].setBrand(inputStr);
    cout << "Product Type: ";
    cin >> inputStr;
    prodArray[count].setType(inputStr);
    cout << "Product Price: ";
    cin >> inputPrice;
    prodArray[count].setPrice(inputPrice);
    cout << "Stock Remaining: ";
    cin >> inputStock;
    prodArray[count].setStockRemaining(inputStock);
    cout << "Product Barcode: ";
    cin >> inputStr;
    prodArray[count].setBarcode(inputStr);

    count++;
  }
```

```
}

void searchByBrand(Product *prodArray, int count)
{
   string target;
   cout << "Enter the product brand to search for: ";
   cin >> target;
   bool found = false;
   for (int i = 0; i < count; i++)
   {
      if (prodArray[i].getBrand() == target)
      {
         cout << "\nProduct found:" << endl;
         cout << "Brand: " << prodArray[i].getBrand() << endl;
         cout << "Type: " << prodArray[i].getType() << endl;
         cout << "Price: " << prodArray[i].getPrice() << endl;
         cout << "Stock Remaining: " << prodArray[i].getStockRemaining() << endl;
         cout << "Barcode: " << prodArray[i].getBarcode() << endl;
         found = true;
      }
   }
   if (!found)
   {
      cout << "No product with brand \"" << target << "\" found." << endl;
   }
}

void searchByType(Product *prodArray, int count)
{
   string target;
   cout << "Enter the product type to search for: ";
   cin >> target;
   bool found = false;
   for (int i = 0; i < count; i++)
   {
      if (prodArray[i].getType() == target)
      {
         cout << "\nProduct found:" << endl;
         cout << "Brand: " << prodArray[i].getBrand() << endl;
         cout << "Type: " << prodArray[i].getType() << endl;
         cout << "Price: " << prodArray[i].getPrice() << endl;
         cout << "Stock Remaining: " << prodArray[i].getStockRemaining() << endl;
         cout << "Barcode: " << prodArray[i].getBarcode() << endl;
         found = true;
      }
   }
   if (!found)
```

```
    {
        cout << "No product with type \"" << target << "\" found." << endl;
    }
}

void searchByBarcode(Product *prodArray, int count)
{
    string target;
    cout << "Enter the product barcode to search for: ";
    cin >> target;
    bool found = false;
    for (int i = 0; i < count; i++)
    {
        if (prodArray[i].getBarcode() == target)
        {
            cout << "\nProduct found:" << endl;
            cout << "Brand: " << prodArray[i].getBrand() << endl;
            cout << "Type: " << prodArray[i].getType() << endl;
            cout << "Price: " << prodArray[i].getPrice() << endl;
            cout << "Stock Remaining: " << prodArray[i].getStockRemaining() << endl;
            cout << "Barcode: " << prodArray[i].getBarcode() << endl;
            found = true;
        }
    }
    if (!found)
    {
        cout << "No product with barcode \"" << target << "\" found." << endl;
    }
}

void displayAllProducts(Product *prodArray, int count)
{
    cout << "\n---------- All Products ----------" << endl;
    for (int i = 0; i < count; i++)
    {
        cout << "\nProduct " << i + 1 << ":" << endl;
        cout << "Brand: " << prodArray[i].getBrand() << endl;
        cout << "Type: " << prodArray[i].getType() << endl;
        cout << "Price: " << prodArray[i].getPrice() << endl;
        cout << "Stock Remaining: " << prodArray[i].getStockRemaining() << endl;
        cout << "Barcode: " << prodArray[i].getBarcode() << endl;
    }
}


int main()
{
```

```
    int capacity = 0;              // no initial capacity
    int count = 0;                 // number of products currently stored
    Product *productsArray = nullptr; // initially no products
    int choice;
    do
    {
        cout << "\n----------------- Product Management Menu -----------------" << endl;
        cout << "1. Add products" << endl;
        cout << "2. Search by brand" << endl;
        cout << "3. Search by type" << endl;
        cout << "4. Search by barcode" << endl;
        cout << "5. Display all products" << endl;
        cout << "0. Exit" << endl;
        cout << "Enter your choice: ";
        cin >> choice;
        cin.ignore(); // clear newline

        switch (choice)
        {
        case 1:
            addProducts(productsArray, capacity, count);
            break;
        case 2:
            searchByBrand(productsArray, count);
            break;
        case 3:
            searchByType(productsArray, count);
            break;
        case 4:
            searchByBarcode(productsArray, count);
            break;
        case 5:
            displayAllProducts(productsArray, count);
            break;
        case 0:
            cout << "Exiting program." << endl;
            break;
        default:
            cout << "Invalid option. Please try again." << endl;
            break;
        }
    } while (choice != 0);
    delete[] productsArray; // free dynamic memory
    return 0;
}
```

21.04.2025

```
#include <iostream>
#include <string>

using namespace std;

class Product
{
private:
   string brand;
   string type;
   float price;
   int stockRemaining;
   string barcode;

public:
   Product()
   {
      brand = "EMPTY";
      type = "EMPTY";
      price = 0.0;
      stockRemaining = 0;
      barcode = "EMPTY";
   }
   void setBrand(string B) { brand = B; }
   void setType(string T) { type = T; }
   void setPrice(float P) { price = P; }
   void setStockRemaining(int S) { stockRemaining = S; }
   void setBarcode(string B) { barcode = B; }
   string getBrand() { return brand; }
   string getType() { return type; }
   float getPrice() { return price; }
   int getStockRemaining() { return stockRemaining; }
   string getBarcode() { return barcode; }
};

Product* addProducts(Product *prodArray, int &capacity, int &count)
{
   int n;
   cout << "How many products would you like to add? ";
   cin >> n;



   // Grow array if needed by asking the user how many extra spots to allocate.
```

```
    if (count + n > capacity)
    {
        cout << "Not enough space. Currently capacity is " << capacity << " and " << count << "
products stored." << endl;
        cout << "Enter additional capacity to allocate (must be at least " << (count + n - capacity) <<
"): ";
        int extra;
        cin >> extra;
        int newCapacity = capacity + extra;
        if (newCapacity < count + n)
        {
            newCapacity = count + n; // Ensure capacity is enough
        }
        Product *temp = new Product[newCapacity];
        for (int i = 0; i < count; i++)
        {
            temp[i] = prodArray[i];
        }
        delete[] prodArray;
        prodArray = temp;
        capacity = newCapacity;
        cout << "Product array capacity increased to " << capacity << "." << endl;
    }

    // Input product details
    for (int i = 0; i < n; i++)
    {
        string inputStr;
        float inputPrice;
        int inputStock;
        cout << "\nEntering details for product " << count + 1 << ":" << endl;

        cout << "Product Brand: ";
        cin >> inputStr;
        prodArray[count].setBrand(inputStr);

        cout << "Product Type: ";
        cin >> inputStr;
        prodArray[count].setType(inputStr);

        cout << "Product Price: ";
        cin >> inputPrice;
        prodArray[count].setPrice(inputPrice);

        cout << "Stock Remaining: ";
        cin >> inputStock;
```

```
        prodArray[count].setStockRemaining(inputStock);

        cout << "Product Barcode: ";
        cin >> inputStr;
        prodArray[count].setBarcode(inputStr);

        count++;
    }

    return prodArray;
}

void searchByBrand(Product *prodArray, int count)
{
    string target;
    cout << "Enter the product brand to search for: ";
    cin >> target;
    bool found = false;
    for (int i = 0; i < count; i++)
    {
        if (prodArray[i].getBrand() == target)
        {
            cout << "\nProduct found:" << endl;
            cout << "Brand: " << prodArray[i].getBrand() << endl;
            cout << "Type: " << prodArray[i].getType() << endl;
            cout << "Price: " << prodArray[i].getPrice() << endl;
            cout << "Stock Remaining: " << prodArray[i].getStockRemaining() << endl;
            cout << "Barcode: " << prodArray[i].getBarcode() << endl;
            found = true;
        }
    }
    if (!found)
    {
        cout << "No product with brand \"" << target << "\" found." << endl;
    }
}

void searchByType(Product *prodArray, int count)
{
    string target;
    cout << "Enter the product type to search for: ";
    cin >> target;
    bool found = false;
    for (int i = 0; i < count; i++)
    {
        if (prodArray[i].getType() == target)
        {
```

```
            cout << "\nProduct found:" << endl;
            cout << "Brand: " << prodArray[i].getBrand() << endl;
            cout << "Type: " << prodArray[i].getType() << endl;
            cout << "Price: " << prodArray[i].getPrice() << endl;
            cout << "Stock Remaining: " << prodArray[i].getStockRemaining() << endl;
            cout << "Barcode: " << prodArray[i].getBarcode() << endl;
            found = true;
        }
    }
    if (!found)
    {
        cout << "No product with type \"" << target << "\" found." << endl;
    }
}

void searchByBarcode(Product *prodArray, int count)
{
    string target;
    cout << "Enter the product barcode to search for: ";
    cin >> target;
    bool found = false;
    for (int i = 0; i < count; i++)
    {
        if (prodArray[i].getBarcode() == target)
        {
            cout << "\nProduct found:" << endl;
            cout << "Brand: " << prodArray[i].getBrand() << endl;
            cout << "Type: " << prodArray[i].getType() << endl;
            cout << "Price: " << prodArray[i].getPrice() << endl;
            cout << "Stock Remaining: " << prodArray[i].getStockRemaining() << endl;
            cout << "Barcode: " << prodArray[i].getBarcode() << endl;
            found = true;
        }
    }
    if (!found)
    {
        cout << "No product with barcode \"" << target << "\" found." << endl;
    }
}

void displayAllProducts(Product *prodArray, int count)
{
    cout << "\n---------- All Products ----------" << endl;


    for (int i = 0; i < count; i++)
    {
```

```
        cout << "\nProduct " << i + 1 << ":" << endl;
        cout << "Brand: " << prodArray[i].getBrand() << endl;
        cout << "Type: " << prodArray[i].getType() << endl;
        cout << "Price: " << prodArray[i].getPrice() << endl;
        cout << "Stock Remaining: " << prodArray[i].getStockRemaining() << endl;
        cout << "Barcode: " << prodArray[i].getBarcode() << endl;
    }
}

int main()
{
    int capacity = 0;              // no initial capacity
    int count = 0;                 // number of products currently stored
    Product *productsArray = nullptr; // initially no products
    int choice;

    do
    {
        cout << "\n---------- Product Management Menu ----------" << endl;
        cout << "1. Add products" << endl;
        cout << "2. Search by brand" << endl;
        cout << "3. Search by type" << endl;
        cout << "4. Search by barcode" << endl;
        cout << "5. Display all products" << endl;
        cout << "0. Exit" << endl;
        cout << "Enter your choice: ";
        cin >> choice;
        cin.ignore(); // clear newline

        switch (choice)
        {
        case 1:
            productsArray = addProducts(productsArray, capacity, count);
            break;
        case 2:
            searchByBrand(productsArray, count);
            break;
        case 3:
            searchByType(productsArray, count);
            break;
        case 4:
            searchByBarcode(productsArray, count);
            break;
        case 5:
            displayAllProducts(productsArray, count);
            break;
        case 0:
```

```
            cout << "Exiting program." << endl;
            break;
        default:
            cout << "Invalid option. Please try again." << endl;
            break;
        }
    } while (choice != 0);

    delete[] productsArray; // free dynamic memory
    return 0;
}
```

22.04.2025

```
#include <iostream>

using namespace std;

// Homework for 22 april 2025
class car
{
private:
   string brand;
   string model;

public:
   car()
   {
      brand = "EMPTY";
      model = "EMPTY";
   }
   void setBrand(string B) { brand = B; }
   void setModel(string M) { model = M; }
   string getBrand() { return brand; }
   string getModel() { return model; }
};

// declare the functions
car *readData(car *, int &, int &);
void showData(car *, int, int);
void searchByBrand(car *, int, string);

int main()
{
   int ARRAYSIZE = 4; // initial size of the array
   // the size of the array is set to 4, but it can be changed by the user
   int nextAvailPos = 0;         // the next available position in the array
   car *carsArray;               // pointer to the array of car objects
   carsArray = new car[ARRAYSIZE]; // allocate memory for the array of car objects
   int choice = -1;              // variable to store the user's choice from the menu
   string brand;                 // variable to store the brand to search for


   do // loop until the user chooses to exit
   {
      cout << "Please Choose from the menu below: \n";
      cout << "1. Enter Data\n";
```

```
        cout << "2. Display Data\n";
        cout << "3. Search By Brand\n";
        cout << "0. Exit\n";
        cin >> choice;

        switch (choice) // switch statement to handle the user's choice
        {
        case 1:
            carsArray = readData(carsArray, ARRAYSIZE, nextAvailPos);
            break;
        case 2:
            showData(carsArray, ARRAYSIZE, nextAvailPos);
            break;
        case 3:
            cout << "Enter the brand to search for: ";
            cin >> brand;
            searchByBrand(carsArray, nextAvailPos, brand);
            break;
        case 0:
            cout << "Exiting \n";
            break;
        default:
            break; // invalid choice, do nothing
        }
    } while (choice != 0);

    delete[] carsArray; // release the memory allocated for the array
    return 0;
}

//  function readData(car *, int)
//  the function populates the array with information from the user
//  the function accepts a pointer to a car object(the array)
//  and the size of the array
//  the function returns back a pointer to a car object which is the array
//  and is populated with data
car *readData(car *arr, int &arrSize, int &nextPos)
{
    int numCars = 0; // numCars = how many cars the user is going to input
    int newSize = 0; // newSize=the new size of the array
    string tstring;  // temporary string to get user input
    car *tmp;       // temporary car object to copy everything from the car array and grow the
array
    cout << "How many cars are you entering: ";
    cin >> numCars; // read from the user how many cars the user needs to input
    for (int i = 0; i < numCars; i++)
    {
```

```
      if (nextPos >= arrSize) // first check if the array is full
      {
         newSize = arrSize + numCars; // the new size of the array which is calculated by adding
arraySize + numCars
         tmp = new car[newSize];     // allocate temporary space for the new bigger array

         for (int j = 0; j < arrSize; j++)
         {
            tmp[j] = arr[j]; // copy the data from the old array to the new one
         }

         arrSize = newSize; // update the size of the array
         delete[] arr;     // release the old array to the OS
         arr = tmp;        // set the pointer of the array to point to the new allocated space
      }

      cout << "Enter brand for car[" << nextPos << "]";
      cin >> tstring;              // read the brand from the user
      arr[nextPos].setBrand(tstring); // set the brand
      cout << "Enter model for car[" << nextPos << "]";
      cin >> tstring;
      arr[nextPos].setModel(tstring); // set the model

      nextPos++; // increment the next available position in the array
   }
   return arr; // return the array with the new data
}

//  function showData(car *, int, int)
//  the function shows the data from the array of car objects
//  the function accepts a pointer to a car object(the array)
//  the size of the array and the next available position in the array
void showData(car *arr, int arrSize, int nextPos)
{
   cout << "Showing the cars \n";
   for (int i = 0; i < nextPos; i++) // loop through the array of car objects
   {
      cout << "Car[" << i << "]\n";
      cout << "Brand: " << arr[i].getBrand() << "\n";
      cout << "Model: " << arr[i].getModel() << "\n";
   }
}

//  function searchByBrand(car *, int, string)
//  the function searches for a car by brand in the array of car objects
//  the function accepts a pointer to a car object(the array)
//  the size of the array, the next available position in the array
```

```
//  and the brand to search for

void searchByBrand(car *arr, int nextPos, string brand)
{
   cout << "Searching for the car with brand: " << brand << "\n";
   bool found = false;            // variable to check if the car is found
   for (int i = 0; i < nextPos; i++) // loop through the array of car objects
   {
      if (arr[i].getBrand() == brand)
      {
         found = true;
         cout << "Car[" << i << "]\n";
         cout << "Brand: " << arr[i].getBrand() << "\n";
         cout << "Model: " << arr[i].getModel() << "\n";
      }
   }
   if (found == false) // if the car is not found, print a message
   {
      cout << "No car with brand: " << brand << "\n";
   }
}
```

23.04.2025

```
#include <iostream>

using namespace std;

// Homework for 23 april 2025
class car
{
private:
   string brand;
   string model;

public:
   car()
   {
      brand = "EMPTY";
      model = "EMPTY";
   }
   void setBrand(string B) { brand = B; }
   void setModel(string M) { model = M; }
   string getBrand() { return brand; }
   string getModel() { return model; }
};

car *readData(car *, int &, int &);
void displayData(car *, int);
int searchByBrand(car *, int);
void editData(car *, int);

int main()
{
   int ARRAYSIZE = 4;0
   int nextAvailPos = 0;
   car *carsArray;
   carsArray = new car[ARRAYSIZE];
   int choice = -1;

   do
   {
      cout << "Please Choose from the menu below: \n";
      cout << "1. Enter Data\n";
      cout << "2. Display Data\n";
      cout << "3. Search By Brand\n";
      cout << "4. Edit Data\n";
```

```
        cout << "0. Exit\n";
        cin >> choice;

        switch (choice)
        {
        case 1:
            carsArray = readData(carsArray, ARRAYSIZE, nextAvailPos);
            break;
        case 2:
            // because we are growing the array dynamically, the variable ARRAYSIZE is the total
            // size of the array which may contain empty positions.
            // The variable nextAvailPos tells us the last index which contains data.
            displayData(carsArray, nextAvailPos);
            break;
        case 3:
            searchByBrand(carsArray, nextAvailPos);
            break;
        case 4:
            editData(carsArray, nextAvailPos);
            break;
        case 0:
            break;
        default:
            cout << "Invalid choice. Please try again.\n";
            break;
        }
    } while (choice != 0);

    return 0;
}

int searchByBrand(car *arr, int arrSize)
{
    string tstr; // temporary string to get user input
    int index;
    cin >> tstr;
    for (int i = 0; i < arrSize; i++)
    {
        if (arr[i].getBrand() == tstr)
        {
            cout << "Index found at position: " << i << "\n";
            index = i;
        }
    }
    return index;
}
```

```
// void displayData(car*,int)
// function displayData(car*,int) will print all data
// that we saved in our object array at any given time
// the function receives as arguments a car object array and the size of the array
void displayData(car *arr, int arrSize)
{
   for (int i = 0; i < arrSize; i++)
   {
      cout << "----- Car " << i + 1 << "-----"
         << "\n";
      cout << arr[i].getBrand() << "\n";
      cout << arr[i].getModel() << "\n";
   }
}

// function editData(car*,int)
// function editData(car*,int) will allow editing of data in the array
void editData(car *arr, int arrSize)
{
   int index = searchByBrand(arr, arrSize); // Call searchByBrand to get the index
   if (index == -1)
   {
      cout << "Car not found. Cannot edit.\n";
      return;
   }

   cout << "Current data at index " << index << ":\n";
   cout << "Brand: " << arr[index].getBrand() << "\n";
   cout << "Model: " << arr[index].getModel() << "\n";

   string newBrand, newModel;
   cout << "Enter new brand: ";
   cin >> newBrand;
   cout << "Enter new model: ";
   cin >> newModel;

   arr[index].setBrand(newBrand); // Update the brand
   arr[index].setModel(newModel); // Update the model

   cout << "Data updated successfully.\n";
}

//  function readData(car *, int)
//  the function populates the array with information from the user
//  the function accepts a pointer to a car object(the array)
//  and the size of the array
//  the function returns back a pointer to a car object which is the array
```

```
//  and is populated with data

car *readData(car *arr, int &arrSize, int &nextPos)
{
   int numCars = 0, newSize = 0; // numCars = how many cars the user is going to input ;
newSize=the new size of the array
   string tstring;            // temporary string to get user input
   car *tmp;                  // temporary car object to copy everything from the car array and grow
the array
   cout << "How many cars are you entering: ";
   cin >> numCars; // read from the user how many cars the user needs to input
   for (int i = 0; i < numCars; i++)
   {
      if (nextPos >= arrSize) // first check if the array is full
      {
         newSize = arrSize + numCars; // the new size of the array which is calculated by adding
arraySize + numCars
         tmp = new car[newSize];     // allocate temporary space for the new bigger array

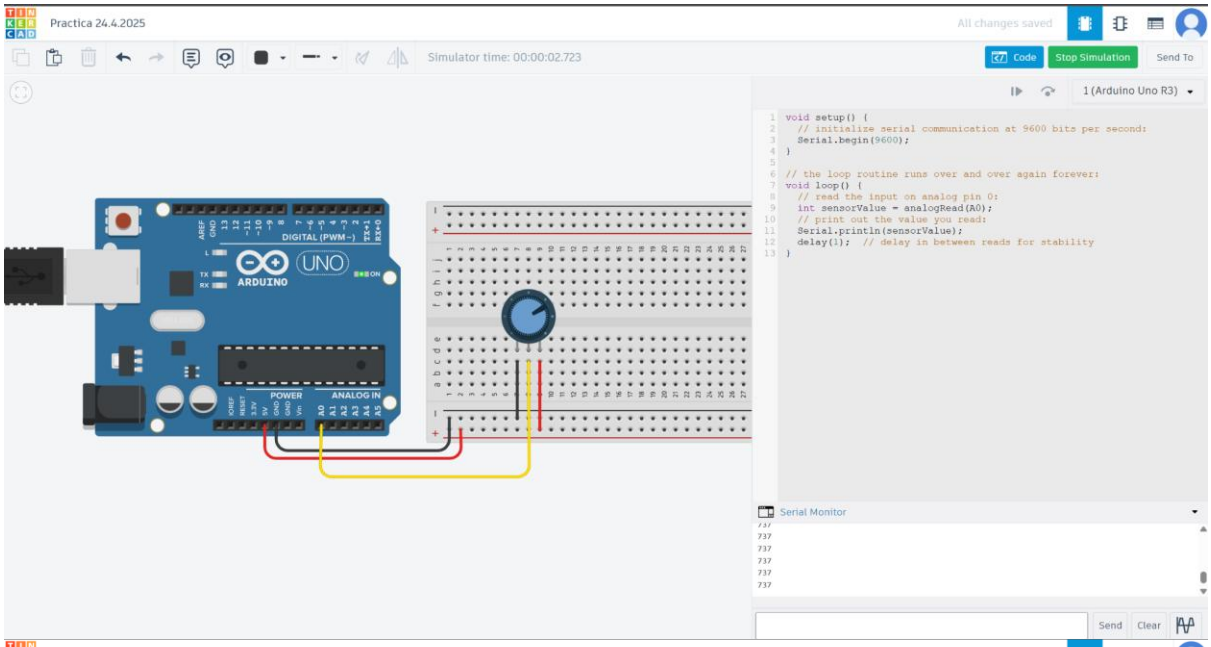
         for (int j = 0; j < arrSize; j++)
         {
            tmp[j] = arr[j]; // copy the data from the old array to the new one
         }

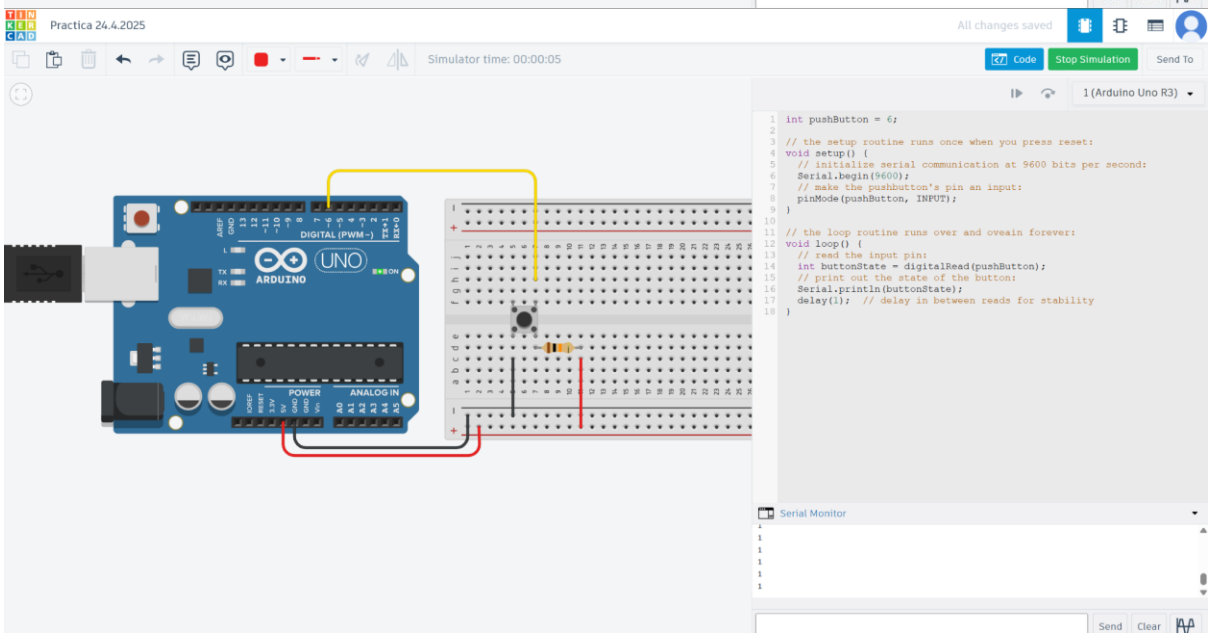
         arrSize = newSize; // update the size of the array
         delete[] arr;      // release the old array to the OS
         arr = tmp;        // set the pointer of the array to point to the new allocated space
      }

      cout << "Enter brand for car[" << nextPos << "]";
      cin >> tstring;
      arr[nextPos].setBrand(tstring); // set the brand
      cout << "Enter model for car[" << nextPos << "]";
      cin >> tstring;
      arr[nextPos].setModel(tstring); // set the model

      nextPos++; // increment the next available position in the array
   }
   return arr; // return the array with the new data
}
```

Arduino


Practica 24.4.2025
All changes saved
Simulator time: 00:00:02.723
Code
Stop Simulation
Send To
1 (Arduino Uno R3)
void setup() {
// initialize serial communication at 9600 bits per second:
Serial.begin(9600);
}
// the loop routine runs over and over again forever:
void loop() {
// read the input on analog pin 0:
int sensorValue = analogRead(A0);
// print out the value you read:
Serial.println(sensorValue);
delay(1); // delay in between reads for stability
}
Serial Monitor
737
737
737
737
737
737
Send
Clear


Practica 24.4.2025
All changes saved
Simulator time: 00:00:05
Code
Stop Simulation
Send To
1 (Arduino Uno R3)
int pushButton = 6;
// the setup routine runs once when you press reset:
void setup() {
// initialize serial communication at 9600 bits per second:
Serial.begin(9600);
// make the pushbutton's pin an input:
pinMode(pushButton, INPUT);
}
// the loop routine runs over and oveain forever:
void loop() {
// read the input pin:
int buttonState = digitalRead(pushButton);
// print out the state of the button:
Serial.println(buttonState);
delay(1); // delay in between reads for stability
}
Serial Monitor
1
1
1
1
1
Send
Clear

Cum să îți găsești un loc de muncă






Interviu
Cum mă pregătesc? Cum mă asigur că îl voi promova?
• Structura generală a unui interviu
• Structura unui interviu de tip coding
• Sfaturi utile

Structura unui interviu de tip Coding
• O problema de încălzire
• 1+ probleme ușoare/medii
• 1+ probleme ulterioare
LeetCode
HackerRank

Sfaturi utile

- **Informează-te** despre **companie** și despre **postul** pe care dorești să îl ocupi
- Pregătește-te să discuți despre **interesele** și **experiențele** tale
- Pune **întrebări**
- Gândește cu **voce tare**

