

**Submitted by:**

**Haseeb ullah F20232661009**

**Submitted to:**

**Raiz Ahmed**

**Subject:**

**PF LAB**

**Section:**

**V21**

**assignment no 12:**

**Searching and Sorting Techniques.**

**Linear Search:**

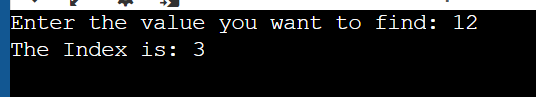
1. Find the first occurrence of a target number in an unsorted array.

* Define an array of integers.
* Prompt the user to enter a target number to search for.
* Use a linear search algorithm to iterate through the array and find the first index where the target number appears.
* Print the index if found, or a message if not found.

**Source Code**

|  |
| --- |
| #include <iostream> using namespace std; int main() {     int arr[5]={45,34,56,12,7};         cout << "Enter the value you want to find: ";     int search;     cin>> search;         bool check = false;         for(int i = 0; i < 5; i++)     {         if(arr[i]==search)         {             check = true;             cout << "The Index is: "<<i<<endl;             break;         }           }         if(check == false)     {             cout << "Not found.";     }     return 0; } |

**Output**



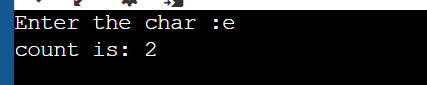
1. Count the occurrences of a specific character in a string.

* Define a string.
* Prompt the user to enter a character to count.
* Use a linear search algorithm to iterate through the string and count the occurrences of the character.
* Print the total count of the character.

**Source Code**

|  |
| --- |
| #include <iostream>  using namespace std;  int main()  {  string name="haseeb";  cout << "Enter the char :";  char ch;  cin>>ch;  int count = 0;  for(int i = 0; i<6;i++)  {  if (ch == name[i])  {  count++;  }  }  cout << "total ouccurences of the character is: " << count;  return 0;  } |

**Output**



**Binary Search:**

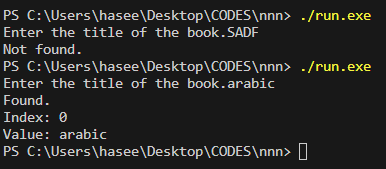
1. Determine if a given book is available in a sorted library catalog.

* Define an array of book titles in sorted order.
* Prompt the user to enter the title of the book they're searching for.
* Use a binary search algorithm to efficiently locate the book's position in the catalog.
* Print a message indicating whether the book is found or not.

**Source code:**

|  |
| --- |
| #include <iostream>  #include <string>  using namespace std;  int main()  {  const int size = 8;  string books[size] =  {  "arabic",  "english",  "ict",  "islamyet",  "programming",  "pakistanstudy",  "Physics",  "urdu"  };    int first = 0,  last = (size - 1),  middle, index = -1;  bool found = false;  string found\_value;      string search;  cout << "Enter the title of the book.";  getline(cin, search);    while(first <= last && !found)  {  middle = (first + last) / 2;    if(search == books[middle])  {  found = true;  index = middle;  found\_value = books[middle];  }  else if(search > books[middle])  {  first = middle + 1;    }else  {  last = middle - 1;  }  }    if (found) {  cout << "Found. \nIndex: " << index << " \nValue: " << found\_value << endl;  } else {  cout << "Not found." << endl;  }    return 0;  } |

**Output**

****

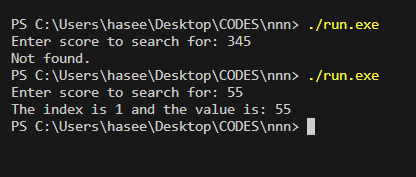
1. Find the index of a specific value within a sorted list of student scores.

* Define an array of student scores in ascending order.
* Prompt the user to enter a score to search for.
* Use a binary search algorithm to find the index of the score within the list.
* Print the index if found, or a message if not found.

**Source Code:**

|  |
| --- |
| // 2. Find the index of a specific value within a sorted list of student scores.  // • Define an array of student scores in ascending order.  // • Prompt the user to enter a score to search for.  // • Use a binary search algorithm to find the index of the score within the list.  // • Print the index if found, or a message if not found.  #include <iostream>  using namespace std;  int main()  {  const int size = 10;  int student\_scores[size] = {  46,55,56,62,73,80,87,88,90,93  };    int search;  cout << "Enter score to search for: ";  cin >> search;  int first = 0,  last = size - 1,  middle;  bool found = false;  int index = 0;  int found\_value;  while (!found && first <= last)  {  middle = (first + last) / 2;  if (student\_scores[middle] == search)  {  found = true;  index = middle;  found\_value=student\_scores[middle];  }else if (student\_scores[middle] > search)  {  last = middle - 1;  }else  {  first = middle + 1;  }  }    if (!found)  {  cout << "Not found.";  }else  {  cout << "The index is " << index << " and the value is: "<< found\_value;  }    return 0;  } |

**Output**

****

**--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

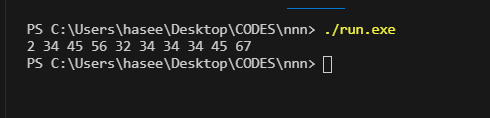
1. Sort a list of integers in descending order.

* Define an array of integers.
* Modify the selection sort algorithm to sort the numbers in descending order instead of ascending.
* Print the sorted list of integers.

**Source Code**

|  |
| --- |
| // 2. Sort a list of integers in descending order.  // • Define an array of integers.  // • Modify the selection sort algorithm to sort the numbers in descending order instead of ascending.  // • Print the sorted list of integers.  #include <iostream>  #include <string>  #include <iomanip>  using namespace std;  int main()  {  int arr[]={  34,45,56,67,34,32,45,34,34,2  };  int size = sizeof (arr) / sizeof(arr[0]);      for (int i = 0; i < size - 1; i++)  {  int min = i;  for (int j = i + 1; j < size; j++)  {  if (arr[i] > arr[j])  {  min = j;  }    }  int temp = arr[i];  arr[i] = arr[min];  arr[min] = temp;    }      for (int i = 0; i < size; i++)  {  cout << arr[i] << " ";  }  return 0;  } |

**Output**

****

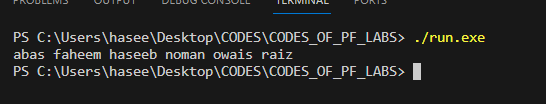
**Selection Sort:**

1. Sort a list of names alphabetically.

* Define an array of names.
* Implement the selection sort algorithm to arrange the names in alphabetical order.
* Print the sorted list of names.

|  |
| --- |
| #include <iostream>  #include <string>  using namespace std;  int main()  {  // Selection Sort:  // 1. Sort a list of names alphabetically.  // • Define an array of names.  // • Implement the selection sort algorithm to arrange the names in alphabetical order.  // • Print the sorted list of names.  const int size = 6;  string names[size] = {  "haseeb", "faheem", "noman", "abas", "raiz", "owais"  };  for (int i = 0; i < size - 1; i++)  {  int min = i;  for (int j = i + 1; j < size; j++)  {  if (names[j] < names[min])  {  min = j;  }    }  if (min != i)  {  string temp = names[i];  names[i] = names[min];  names[min] = temp;  }    }    for (int i = 0; i < size; i++)  {  cout << names[i] << " ";  }  return 0;  } |

**Output**



**Bubble Sort:**

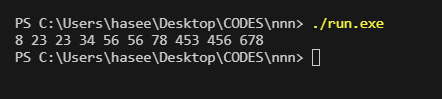
1. Sort a list of randomly generated numbers.

* Define an array and fill it with randomly generated numbers.
* Implement the bubble sort algorithm to arrange the numbers in ascending order.
* Print the sorted list of numbers.

**Source Code**

|  |
| --- |
| #include <iostream>  #include <cstdlib>  #include <ctime>  using namespace std;  int main()  {  const int size = 10;  int numbers[size];  srand(time(0));  for (int i = 0; i < size; i++)  {  numbers[i] = rand() % 1000;  }  cout << "Unsorted array: ";  for (int i = 0; i < size; i++)  {  cout << numbers[i] << " ";  }  cout << endl;  bool swap;  int temp;  do  {  swap = false;  for (int i = 0; i < size - 1; i++)  {  if (numbers[i] > numbers[i + 1])  {  temp = numbers[i];  numbers[i] = numbers[i + 1];  numbers[i + 1] = temp;  swap = true;  }  }  } while (swap);  cout << "Sorted array: ";  for (int i = 0; i < size; i++)  {  cout << numbers[i] << " ";  }  return 0;  } |

**Output**

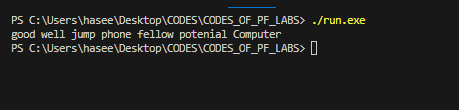
****

1. Sort a list of words based on their lengths.
   * Define an array of words.
   * Adapt the bubble sort algorithm to compare word lengths instead of numerical values.
   * Print the sorted list of words, arranged from shortest to longest.

**Source Code**

|  |
| --- |
| #include <iostream>  #include <string>  using namespace std;  int main()  {  string words[]={  "good", "fellow", "well", "potenial", "jump", "phone", "Computer"  };  int size = sizeof(words) / sizeof(words[0]);    bool swap;  string temp;  do  {  swap =false;  for (int i = 0; i < size - 1; i++)  {  if (words[i].length() > words[ i + 1 ].length())  {  swap = true;  temp = words[i];  words[i] = words[i + 1];  words[i + 1] = temp;  }    }    } while (swap);    for (int i = 0; i < size; i++)  {  cout << words[i] << " ";  }    return 0;  } |

**Output**

****