**Write a generic function template named findMinimum in C++ that takes an array of any data type T and its size n as arguments. The function should return the minimum element present in the array.**

#include <iostream>

using namespace std;

template <typename T>

T findMinimum(const T\* array, int n)

{

T minElement = array[0];

for (int i = 1; i < n; ++i)

{

if (array[i] < minElement)

{

minElement = array[i];

}

}

return minElement;

}

int main()

{

int intArray[] = {8, 2, 3, 6, 5, 4, 7, 9, 3, 5};

int intSize = sizeof(intArray) / sizeof(intArray[0]);

cout << "Minimum value in intArray is: " << findMinimum(intArray, intSize) << endl;

double doubleArray[] = {4.213, 1.341, 5.414, 1.732, 0.294, 2.618};

int doubleSize = sizeof(doubleArray) / sizeof(doubleArray[0]);

cout << "Minimum value in doubleArray is: " << findMinimum(doubleArray, doubleSize) << endl;

string stringArray[] = {"fuuit", "vegetable", "car", "bike", "chair"};

int strSize = sizeof(stringArray) / sizeof(stringArray[0]);

cout << "Minimum value in stringArray is: " << findMinimum(stringArray, strSize) << endl;

return 0;

}

**OUTPUT:**

**Minimum value in intArray is: 2**

**Minimum value in doubleArray is: 0.294**

**Minimum value in stringArray is: bike**

**Swap Elements:**

**Problem: Write a function template swap that takes two pointers to variables of any data type T and swaps their values.**

Constraints: The function should only modify the values pointed to by the arguments, not the arguments themselves (pass by reference).

#include <iostream>

using namespace std;

template <typename T>

void swap(T\* a, T\* b)

{

T temp = \*a;

\*a = \*b;

\*b = temp;

}

int main()

{

int x = 5;

int y = 10;

cout << "Before swap: x = " << x << ", y = " << y << endl;

swap(&x, &y);

cout << "After swap: x = " << x << ", y = " << y << endl;

double dx = 2.8;

double dy = 3.7;

cout << "Before swap: dx = " << dx << ", dy = " << dy << endl;

swap(&dx, &dy);

cout << "After swap: dx = " << dx << ", dy = " << dy << endl;

string str1 = "Praneeth";

string str2 = "Reddy";

cout << "Before swap: str1 = " << str1 << ", str2 = " << str2 << endl;

swap(&str1, &str2);

cout << "After swap: str1 = " << str1 << ", str2 = " << str2 << endl;

return 0;

}

**OUTPUT:**

**Before swap: x = 5, y = 10**

**After swap: x = 10, y = 5**

**Before swap: dx = 2.8, dy = 3.7**

**After swap: dx = 3.7, dy = 2.8**

**Before swap: str1 = Praneeth, str2 = Reddy**

**After swap: str1 = Reddy, str2 = Praneeth**

**Find Maximum:**

**Problem: Similar to findMinimum, create a function template findMaximum that returns the maximum element in an array of any data type T.**

#include <iostream>

using namespace std;

template <typename T>

T findMaximum(T arr[], int n)

{

T max = arr[0];

for (int i = 1; i < n; ++i)

{

if (arr[i] > max)

{

max = arr[i];

}

}

return max;

}

int main()

{

int intArr[] = {8, 1, 4, 3, 5, 8, 2, 1, 5};

int nInt = sizeof(intArr) / sizeof(intArr[0]);

cout << "Maximum element in integer array: " << findMaximum(intArr, nInt) << endl;

double doubleArr[] = {3.09, 2.14, 1.41, 1.62, 2.73};

int nDouble = sizeof(doubleArr) / sizeof(doubleArr[0]);

cout << "Maximum element in double array: " << findMaximum(doubleArr, nDouble) << endl;

string stringArr[] = {"cat", "mat", "sat", "bat"};

int nStr = sizeof(stringArr) / sizeof(stringArr[0]);

cout << "Maximum element in string array: " << findMaximum(stringArr, nStr) << endl;

return 0;

}

**OUTPUT:**

**Maximum element in integer array: 8**

**Maximum element in double array: 3.09**

**Maximum element in string array: sat**

**Class Template:**

#include <iostream>

using namespace std;

template<class T>

class A

{

public:

T num1=5;

T num2=6;

void add()

{

std::cout<<"Addition of num1 and num2:"<<num1+num2<<std::endl;

}

};

int main()

{

A<int>d;

d.add();

return 0;

}

**OUTPUT:**

**Addition of num1 and num2:11**

**Class Template with multiple parameters**

#include <iostream>

using namespace std;

template<class T1, class T2>

class A

{

T1 a;

T2 b;

public:

A(T1 x,T2 y)

{

a=x;

b=y;

}

void display()

{

std::cout<<"Values of a and b are:"<<a<<","<<b<<std::endl;

}

};

int main()

{

A<int,float>d(5,6.5);

d.display();

return 0;

}

**OUTPUT:**

**Values of a and b are:5,6.5**