Cấu trúc dữ liệu và giải thuật

1/Cho mảng gồm các phần tử { 2, 3, 4, 10, 40 } nhập vào từ bàn phím. Viết chương trình tìm kiếm giá trị 10. Sử dụng cài đặt tìm kiếm nhị phân để tìm kiếm.

**Giải:#include <iostream>**

**using namespace std;**

**int binarySearch(int arr[], int l, int r, int x) {**

**while (l <= r) {**

**int mid = l + (r - l) / 2;**

**if (arr[mid] == x)**

**return mid;**

**if (arr[mid] < x)**

**l = mid + 1;**

**else**

**r = mid - 1;**

**}**

**return -1;**

**}**

**int main() {**

**int arr[] = {2, 3, 4, 10, 40};**

**int n = sizeof(arr) / sizeof(arr[0]);**

**int x = 10;**

**int result = binarySearch(arr, 0, n - 1, x);**

**if (result == -1)**

**cout << "Khong tim thay gia tri " << x << endl;**

**else**

**cout << "Tim thay tai vi tri " << result << endl;**

**return 0;**

**}**

**2/**Cho mảng gồm các phần tử { 2, 3, 4, 10, 40 } nhập vào từ bàn phím. Viết chương trình tìm kiếm giá trị 10. Sử dụng cài đặt tìm kiếm tuyến tính để tìm kiếm.

**Giải:#include <iostream>**

**using namespace std;**

**int main() {**

**int arr[] = {2, 3, 4, 10, 40};**

**int n = sizeof(arr) / sizeof(arr[0]);**

**int x = 10;**

**for (int i = 0; i < n; i++) {**

**if (arr[i] == x) {**

**cout << "Tim thay tai vi tri " << i << endl;**

**return 0;**

**}**

**}**

**cout << "Khong tim thay gia tri " << x << endl;**

**return 0;**

**}**

**3/**Cho mảng gồm các phần tử { 41, 23, 4, 14, 56, 1 } nhập vào từ bàn phím. Viết chương trình để sắp xếp. Sử dụng phương pháp sắp xếp đổi chỗ trực tiếp để sắp xếp**.**

**Giải:#include <iostream>**

**using namespace std;**

**void bubbleSort(int arr[], int n) {**

**for (int i = 0; i < n - 1; i++) {**

**for (int j = 0; j < n - i - 1; j++) {**

**if (arr[j] > arr[j + 1]) {**

**int temp = arr[j];**

**arr[j] = arr[j + 1];**

**arr[j + 1] = temp;**

**}**

**}**

**}**

**}**

**int main() {**

**int arr[] = {41, 23, 4, 14, 56, 1};**

**int n = sizeof(arr) / sizeof(arr[0]);**

**bubbleSort(arr, n);**

**cout << "Mang sau khi sap xep: ";**

**for (int i = 0; i < n; i++)**

**cout << arr[i] << " ";**

**cout << endl;**

**return 0;**

**}**

**4/**Cho mảng gồm các phần tử { 41, 23, 4, 14, 56, 1 } nhập vào từ bàn phím. Viết chương trình để sắp xếp. Sử dụng phương pháp sắp xếp chọn trực tiếp để sắp xếp.

**Giải**:#include <stdio.h>

void swap(int \*xp, int \*yp)

{

int temp = \*xp;

\*xp = \*yp;

\*yp = temp;

}

void selectionSort(int arr[], int n)

{

int i, j, min\_idx;

for (i = 0; i < n-1; i++)

{

min\_idx = i;

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

{

if (arr[j] < arr[min\_idx])

min\_idx = j;

}

swap(&arr[min\_idx], &arr[i]);

}

}

int main()

{

int arr[] = {41, 23, 4, 14, 56, 1};

int n = sizeof(arr)/sizeof(arr[0]);

int i;

printf("Mang truoc khi sap xep: \n");

for (i=0; i < n; i++)

printf("%d ", arr[i]);

selectionSort(arr, n);

printf("\nMang sau khi sap xep: \n");

for (i=0; i < n; i++)

printf("%d ", arr[i]);

printf("\n");

return 0;

}

**5/**Cho mảng gồm các phần tử { 41, 23, 4, 14, 56, 1 } nhập vào từ bàn phím. Viết chương trình để sắp xếp. Sử dụng phương pháp sắp xếp chèn trực tiếp để sắp xếp

Giải:#include <stdio.h>

void insertionSort(int arr[], int n)

{

int i, key, j;

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

{

key = arr[i];

j = i - 1;

while (j >= 0 && arr[j] > key)

{

arr[j + 1] = arr[j];

j = j - 1;

}

arr[j + 1] = key;

}

}

int main()

{

int arr[] = {41, 23, 4, 14, 56, 1};

int n = sizeof(arr)/sizeof(arr[0]);

int i;

printf("Mang truoc khi sap xep: \n");

for (i=0; i < n; i++)

printf("%d ", arr[i]);

insertionSort(arr, n);

printf("\nMang sau khi sap xep: \n");

for (i=0; i < n; i++)

printf("%d ", arr[i]);

printf("\n");

return 0;

}

6/Cho mảng gồm các phần tử { 41, 23, 4, 14, 56, 1 } nhập vào từ bàn phím. Viết chương trình để sắp xếp. Sử dụng phương pháp sắp xếp nhanh để sắp xếp.

**Giải:#include <iostream>**

**using namespace std;**

**void quickSort(int arr[], int low, int high) {**

**if (low < high) {**

**int pivot = arr[high];**

**int i = low - 1;**

**for (int j = low; j < high; j++) {**

**if (arr[j] <= pivot) {**

**i++;**

**swap(arr[i], arr[j]);**

**}**

**}**

**swap(arr[i + 1], arr[high]);**

**int pi = i + 1;**

**quickSort(arr, low, pi - 1);**

**quickSort(arr, pi + 1, high);**

**}**

**}**

**int main() {**

**int arr[] = {41, 23, 4, 14, 56, 1};**

**int n = sizeof(arr) / sizeof(arr[0]);**

**quickSort(arr, 0, n - 1);**

**cout << "Mang sau khi sap xep: ";**

**for (int i = 0; i < n; i++)**

**cout << arr[i] << " ";**

**cout << endl;**

**return 0;**

**}**

**7/**Cho mảng gồm các phần tử { 41, 23, 4, 14, 56, 1 } nhập vào từ bàn phím. Viết chương trình để sắp xếp. Sử dụng phương pháp sắp xếp cây để sắp xếp.

**Giải:**#include <iostream>

using namespace std;

void heapify(int arr[], int n, int i) {

int largest = i;

int l = 2 \* i + 1;

int r = 2 \* i + 2;

if (l < n && arr[l] > arr[largest])

largest = l;

if (r < n && arr[r] > arr[largest])

largest = r;

if (largest != i) {

swap(arr[i], arr[largest]);

heapify(arr, n, largest);

}

}

void heapSort(int arr[], int n) {

for (int i = n / 2 - 1; i >= 0; i--)

heapify(arr, n, i);

for (int i = n - 1; i >= 0; i--) {

swap(arr[0], arr[i]);

heapify(arr, i, 0);

}

}

int main() {

int arr[] = {41, 23, 4, 14, 56, 1};

int n = sizeof(arr) / sizeof(arr[0]);

heapSort(arr, n);

cout << "Mang sau khi sap xep: ";

for (int i = 0; i < n; i++)

cout << arr[i] << " ";

cout << endl;

return 0;

}

8/Viết chương trình cài đặt stack bằng mảng. Nhập vào stack 6 phần tử { 41, 23, 4, 14, 56, 1 }. Sau đó xuất giá trị top ra khỏi stack và in các phần tử trong stack còn lại.

Giải:#include <iostream>

using namespace std;

#define MAX\_SIZE 100

class Stack {

private:

int top;

int arr[MAX\_SIZE];

public:

Stack() {

top = -1;

}

bool isEmpty() {

return top == -1;

}

bool isFull() {

return top == MAX\_SIZE - 1;

}

void push(int val) {

if (isFull()) {

cout << "Stack is full. Cannot push value " << val << endl;

return;

}

arr[++top] = val;

cout << "Pushed value " << val << " into stack" << endl;

}

int pop() {

if (isEmpty()) {

cout << "Stack is empty. Cannot pop." << endl;

return -1;

}

int val = arr[top--];

cout << "Popped value " << val << " from stack" << endl;

return val;

}

int peek() {

if (isEmpty()) {

cout << "Stack is empty. Cannot peek." << endl;

return -1;

}

return arr[top];

}

void printStack() {

if (isEmpty()) {

cout << "Stack is empty." << endl;

return;

}

cout << "Stack elements: ";

for (int i = top; i >= 0; i--) {

cout << arr[i] << " ";

}

cout << endl;

}

};

int main() {

Stack s;

int arr[6] = {41, 23, 4, 14, 56, 1};

for (int i = 0; i < 6; i++) {

s.push(arr[i]);

}

int poppedValue = s.pop();

cout << "Top element of stack: " << s.peek() << endl;

s.printStack();

return 0;

}

9/ Viết chương trình cài đặt hàng đợi bằng mảng. Nhập vào queue 6 phần tử { 41, 23, 4, 14, 56, 1 }. Sau đó thêm vào phần tử {55} và loại bỏ phần {23}. In hàng đợi sau khi thêm và bỏ.

**Giải**:#include <iostream>

using namespace std;

#define MAX 100 // kích thước của hàng đợi

class Queue {

private:

int front, rear; // chỉ số phía trước và phía sau của hàng đợi

int arr[MAX]; // mảng lưu trữ các phần tử hàng đợi

public:

Queue() { // constructor, khởi tạo hàng đợi rỗng

front = -1;

rear = -1;

}

bool isFull() { // kiểm tra hàng đợi đã đầy chưa

return rear == MAX - 1;

}

bool isEmpty() { // kiểm tra hàng đợi có rỗng không

return front == -1 || front > rear;

}

void enqueue(int x) { // thêm phần tử vào cuối hàng đợi

if (isFull()) {

cout << "Hang doi da day!";

return;

}

if (front == -1) {

front = 0;

}

arr[++rear] = x;

cout << "Da them phan tu " << x << " vao hang doi." << endl;

}

void dequeue() { // loại bỏ phần tử ở đầu hàng đợi

if (isEmpty()) {

cout << "Hang doi rong!";

return;

}

int x = arr[front++];

cout << "Da xoa phan tu " << x << " khoi hang doi." << endl;

}

void display() { // hiển thị các phần tử trong hàng đợi

if (isEmpty()) {

cout << "Hang doi rong!";

return;

}

cout << "Cac phan tu trong hang doi la: ";

for (int i = front; i <= rear; i++) {

cout << arr[i] << " ";

}

cout << endl;

}

};

int main() {

Queue q;

int arr[6] = {41, 23, 4, 14, 56, 1};

for (int i = 0; i < 6; i++) {

q.enqueue(arr[i]);

}

q.enqueue(55); // thêm phần tử 55 vào hàng đợi

q.dequeue(); // loại bỏ phần tử đầu tiên (23) trong hàng đợi

q.display(); // hiển thị các phần tử còn lại trong hàng đợi

return 0;

}

10/Cho mảng 1 chiều các số nguyên. Viết hàm tính tổng các số chẵn trong mảng bằng phương pháp đệ quy.

**Giải:#include <iostream>**

**using namespace std;**

**int sumEvenRecursive(int arr[], int n) {**

**// Base case**

**if (n == 1) {**

**return (arr[0] % 2 == 0) ? arr[0] : 0;**

**}**

**// Recursive case**

**int sum = sumEvenRecursive(arr + 1, n - 1);**

**return (arr[0] % 2 == 0) ? sum + arr[0] : sum;**

**}**

**int main() {**

**int arr[] = {1, 2, 3, 4, 5, 6};**

**int n = sizeof(arr) / sizeof(arr[0]);**

**int sum = sumEvenRecursive(arr, n);**

**cout << "Tong cac so chan trong mang la: " << sum << endl;**

**return 0;**

**}**

**11/** Cho mảng 1 chiều các số thực. Viết hàm đếm số lượng giá trị dương trong mảng bằng phương pháp đệ quy.

**Giải**:#include <iostream>

using namespace std;

int countPositive(float arr[], int n) {

if (n == 0) {

return 0;

} else {

int count = countPositive(arr, n-1);

if (arr[n-1] > 0) {

count++;

}

return count;

}

}

int main() {

float arr[] = { 3.14, -2.5, 6.7, -1.0, 0.0, 9.8 };

int n = sizeof(arr) / sizeof(arr[0]);

int count = countPositive(arr, n);

cout << "Number of positive elements in the array: " << count << endl;

return 0;

}

12/Cho mảng 1 chiều các số nguyên. Viết hàm đệ quy xuất mảng.

**Giải**:#include <iostream>

using namespace std;

// Hàm đệ quy xuất mảng

void xuatMang(int arr[], int n, int index) {

// Nếu chỉ số phần tử hiện tại lớn hơn hoặc bằng số lượng phần tử trong mảng, dừng đệ quy

if (index >= n) {

return;

}

// Xuất giá trị phần tử hiện tại

cout << arr[index] << " ";

// Gọi đệ quy để xuất phần tử tiếp theo

xuatMang(arr, n, index + 1);

}

int main() {

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

int n = 5;

cout << "Mang: ";

xuatMang(arr, n, 0); // Gọi hàm đệ quy để xuất mảng

return 0;

}

13/Cho mảng 1 chiều các số thực. Hãy viết hàm đệ quy đếm số lượng giá trị dương có trong mảng.

Giải:#include <iostream>

using namespace std;

int countPositive(float arr[], int n) {

// base case

if (n == 0) {

return 0;

}

// recursive case

int count = countPositive(arr, n-1); // đệ quy đến phần tử trước

if (arr[n-1] > 0) { // nếu phần tử cuối cùng là số dương

count++; // tăng số lượng lên 1

}

return count;

}

int main() {

float arr[] = {1.5, -2.3, 4.6, -0.9, 5.0, -3.7};

int n = sizeof(arr)/sizeof(arr[0]);

cout << "So luong gia tri duong trong mang la: " << countPositive(arr, n);

return 0;

}

14/Hãy khai báo cấu trúc dữ liệu của cây nhị phân các số nguyên

Giải:

|  |
| --- |
| struct node |
|  | { |
|  | int info; |
|  | struct node \*pNext; |
|  | struct node \*pPrev; |
|  | }; |
|  |  |
|  |  |
|  | typedef NODE \*TREE; |

15/Hãy khai báo cấu trúc dữ liệu của danh sách liên kết kép các số nguyên

Giải:

|  |
| --- |
| struct node |
|  | { |
|  | int info; |
|  | struct node \*pNext; |
|  | struct node \*pPrev; |
|  | }; |
|  | typedef struct node NODE; |
|  |  |
|  | struct list |
|  | { |
|  | NODE \*pHead; |
|  | NODE \*pPrev; |
|  | }; |
|  | typedef struct list LIST; |

16/Cho mảng 1 chiều các số thực. Hãy viết hàm đệ quy tính tổng các giá trị dương có trong mảng

Giải:#include <stdio.h>

float sumPositive(float arr[], int size) {

if (size == 0) { // trường hợp cơ sở: kích thước mảng là 0

return 0;

}

else {

float sum = sumPositive(arr, size-1); // gọi đệ quy với mảng có kích thước nhỏ hơn

if (arr[size-1] > 0) { // nếu phần tử hiện tại lớn hơn 0, thì cộng vào tổng

sum += arr[size-1];

}

return sum;

}

}

int main() {

float arr[] = {1.5, -2.0, 3.2, 4.0, -5.5, 6.1, 7.0};

int size = sizeof(arr)/sizeof(arr[0]);

float sum = sumPositive(arr, size);

printf("Tong cac gia tri duong trong mang la: %f", sum);

return 0;

}

17/Cho mảng 1 chiều các số thực. Hãy viết hàm đệ quy tính tổng các giá trị có trong mảng.

Giải:#include <iostream>

using namespace std;

// Hàm tính tổng các giá trị trong mảng

float sum(float arr[], int size) {

if (size == 0) { // trường hợp cơ sở: kích thước mảng là 0

return 0;

}

else {

return arr[size-1] + sum(arr, size-1); // gọi đệ quy với mảng có kích thước nhỏ hơn

}

}

int main() {

float arr[] = {1.5, -2.0, 3.2, 4.0, -5.5, 6.1, 7.0};

int size = sizeof(arr)/sizeof(arr[0]);

float s = sum(arr, size);

cout << "Tong cac gia tri trong mang la: " << s;

return 0;

}

18/Cho mảng 1 chiều các số thực. Hãy viết hàm đệ quy kiểm tra mảng có thỏa mảng tính chất “toàn giá trị âm”

Giải:#include <iostream>

using namespace std;

// Hàm kiểm tra tính chất "toàn giá trị âm" của mảng

bool isAllNegative(float arr[], int size) {

if (size == 0) { // trường hợp cơ sở: kích thước mảng là 0

return true;

}

else if (arr[size-1] >= 0) { // nếu phần tử cuối cùng của mảng không âm, không thỏa tính chất "toàn giá trị âm"

return false;

}

else {

return isAllNegative(arr, size-1); // gọi đệ quy với mảng có kích thước nhỏ hơn

}

}

int main() {

float arr1[] = {-1.5, -2.0, -3.2, -4.0, -5.5, -6.1, -7.0};

int size1 = sizeof(arr1)/sizeof(arr1[0]);

float arr2[] = {-1.5, -2.0, 3.2, -4.0, -5.5, -6.1, -7.0};

int size2 = sizeof(arr2)/sizeof(arr2[0]);

if (isAllNegative(arr1, size1)) {

cout << "Mang 1 thoa man tinh chat 'toan gia tri am'";

}

else {

cout << "Mang 1 khong thoa man tinh chat 'toan gia tri am'";

}

if (isAllNegative(arr2, size2)) {

cout << "Mang 2 thoa man tinh chat 'toan gia tri am'";

}

else {

cout << "Mang 2 khong thoa man tinh chat 'toan gia tri am'";

}

return 0;

}

19/Cho mảng 1 chiều các số thực. Hãy viết hàm đệ quy tìm giá trị lớn nhất có trong mảng

Giải:#include <iostream>

using namespace std;

// Hàm tìm giá trị lớn nhất trong mảng

float findMax(float arr[], int size) {

if (size == 1) { // trường hợp cơ sở: mảng chỉ có một phần tử

return arr[0];

}

else {

float max = findMax(arr, size-1); // gọi đệ quy với mảng có kích thước nhỏ hơn

if (arr[size-1] > max) { // so sánh phần tử cuối cùng của mảng với giá trị lớn nhất đã tìm được

return arr[size-1];

}

else {

return max;

}

}

}

int main() {

float arr[] = {3.4, 5.7, -2.8, 4.1, 2.6, 8.9};

int size = sizeof(arr)/sizeof(arr[0]);

float max = findMax(arr, size);

cout << "Gia tri lon nhat trong mang la: " << max;

return 0;

}

20/Cho mảng 1 chiều các số thực. Hãy viết hàm đệ quy sắp xếp các giá trị trong mảng tăng dần

Giải:#include <iostream>

using namespace std;

// Hàm hoán đổi giá trị của hai biến

void swap(float &a, float &b) {

float temp = a;

a = b;

b = temp;

}

// Hàm đệ quy sắp xếp mảng tăng dần

void sortArray(float arr[], int size) {

if (size <= 1) { // trường hợp cơ sở: mảng chỉ có một phần tử hoặc không có phần tử nào

return;

}

else {

sortArray(arr, size-1); // sắp xếp đệ quy mảng trừ phần tử cuối cùng

if (arr[size-1] < arr[size-2]) { // so sánh phần tử cuối cùng với phần tử liền trước

swap(arr[size-1], arr[size-2]); // nếu phần tử cuối cùng nhỏ hơn phần tử liền trước, hoán đổi chúng

sortArray(arr, size-1); // sau khi hoán đổi, tiếp tục sắp xếp đệ quy mảng trừ phần tử cuối cùng

}

}

}

int main() {

float arr[] = {3.4, 5.7, -2.8, 4.1, 2.6, 8.9};

int size = sizeof(arr)/sizeof(arr[0]);

cout << "Mang truoc khi sap xep: ";

for (int i = 0; i < size; i++) {

cout << arr[i] << " ";

}

sortArray(arr, size);

cout << "\nMang sau khi sap xep: ";

for (int i = 0; i < size; i++) {

cout << arr[i] << " ";

}

return 0;

}

21/Tính T(n) = 1 x 2 x 3 x … x n

Giải:#include <iostream>

using namespace std;

int T(int n) {

if(n == 1) { // n = 1, trả về 1

return 1;

} else { // n > 1, tính T(n) = n x T(n-1)

return n \* T(n-1);

}

}

int main() {

int n;

cout << "Nhap n: ";

cin >> n;

cout << "T(" << n << ") = " << T(n) << endl;

return 0;

}