Вкладка 1

## **1. Однозв’язні списки**

struct Node {

int data;

Node\* next;

};

**// Додати елемент в кінець**

void pushBack (Node\*& head, int value){

Node\* newNode = new Node;

newNode->data = value;

newNode->next = nullptr;

if(head==nullptr){

head = newNode;

} else{

Node\* temp = head;

while(temp->next !=nullptr){

temp = temp->next;

}

temp->next = newNode;

}

}

**// Вивести список**

void outputList (Node\*& head){

Node\* temp = head;

while (temp!=nullptr){

cout << temp->data << " ";

temp = temp->next;

}

cout << endl;

}

**// Знайти елемент**

void findElement (Node\* head, int key){

Node\* temp = head;

while (temp != nullptr){

if(temp->data == key){

cout << "Element founded." << endl;

return;

}

temp = temp->next;

}

cout << "Element not founded." << endl;

}

**// Видалити елемент**

void deleteValue (Node\*& head, int key2){

if (head==nullptr) return;

if (head->data == key2){

Node\* toDelete = head;

head = head->next;

delete toDelete;

return;

}

Node\* temp= head;

while (temp->next != nullptr && temp->next->data != key2){

temp = temp->next;

}

if (temp->next == nullptr){

cout << "Element not founded." << endl;

} else {

Node\* toDelete = temp->next;

temp->next = temp->next->next;

delete toDelete;

}

}

**// Вивести навпаки**

void printReverse(Node\* head) {

if (head==nullptr) return;

printReverse(head->next);

cout << head->data << " ";

}

**// Видалити список (очищення пам’яті)**

void deleteList(Node\*& head){

while (head != nullptr){

Node\* next = head->next;

delete head;

head = next;

}

}

**// Видалення від’ємних елементів**

void deleteNegatives(Node\*& head) {

while (head && head->data < 0) {

Node\* d = head; head = head->next; delete d;

}

Node\* t = head;

while (t && t->next) {

if (t->next->data < 0) {

Node\* d = t->next;

t->next = d->next;

delete d;

} else t = t->next;

}

}

**// Перемістити max у кінець**

void moveMaxToEnd(Node\*& head) {

if (!head || !head->next) return;

Node\* prevMax=nullptr,\*maxNode=head;

Node\* t=head;

while(t->next){

if(t->next->data > maxNode->data){ prevMax=t; maxNode=t->next; }

t=t->next;

}

if(maxNode->next==nullptr) return;

if(prevMax) prevMax->next=maxNode->next;

else head=head->next;

t->next=maxNode;

maxNode->next=nullptr;

}

## 

## 

## 

## 

## 

## **2. Двозв’язні списки**

struct DNode {

int data;

DNode\* prev;

DNode\* next;

};

**// Додавання в кінець**

void pushBack(DNode\*& head, int x) {

DNode\* n = new DNode{x,nullptr,nullptr};

if (!head) { head = n; return; }

DNode\* t = head;

while (t->next) t = t->next;

t->next = n;

n->prev = t;

}

**// Вивід**

void print(DNode\* head) {

while (head) { cout << head->data << " "; head = head->next; }

cout << endl;

**// Впорядкування спаданням**

void sortDesc(DNode\*& head) {

if (!head) return;

for (DNode\* i=head;i;i=i->next)

for (DNode\* j=i->next;j;j=j->next)

if (i->data < j->data) swap(i->data,j->data);

}

**// Розбиття на парні/непарні позиції**

void reorderOddEven(DNode\*& head){

if(!head) return;

DNode\* odd=nullptr,\*even=nullptr,\*to=nullptr;

int pos=1;

for(DNode\* t=head;t;t=t->next,pos++){

DNode\* n=new DNode{t->data,nullptr,nullptr};

if(pos%2){ if(!odd) odd=to=n; else {to->next=n;n->prev=to;to=n;} }

else { if(!even) even=to=n; else {to->next=n;n->prev=to;to=n;} }

}

if(!odd){ head=even; return; }

DNode\* o=odd; while(o->next)o=o->next;

o->next=even; if(even) even->prev=o;

head=odd;

}

## **3. Стек (масив)**

const int MAX = 100;

struct Stack {

int a[MAX];

int top=-1;

};

bool empty(Stack &s){ return s.top==-1; }

bool full(Stack &s){ return s.top==MAX-1; }

void push(Stack &s,int x){ if(!full(s)) s.a[++s.top]=x; }

int pop(Stack &s){ return empty(s)?-1:s.a[s.top--]; }

## **4. Черга (циклічна, масив)**

const int N=100;

struct Queue {

int a[N], front=0, rear=-1, cnt=0;

};

bool empty(Queue&q){ return q.cnt==0; }

bool full(Queue&q){ return q.cnt==N; }

void enqueue(Queue&q,int x){

if(full(q)) return;

q.rear=(q.rear+1)%N; q.a[q.rear]=x; q.cnt++;

}

void dequeue(Queue&q){

if(empty(q)) return;

q.front=(q.front+1)%N; q.cnt--;

}

**5. Бінарні дерева пошуку (BST)**

struct NodeT{

int key;

NodeT\* l;

NodeT\* r;

};

**// Вставка**

NodeT\* insert(NodeT\* root,int x){

if(!root) return new NodeT{x,nullptr,nullptr};

if(x<root->key) root->l=insert(root->l,x);

else root->r=insert(root->r,x);

return root;

}

**// Пошук**

bool find(NodeT\* root,int x){

if(!root) return false;

if(root->key==x) return true;

return x<root->key?find(root->left,x):find(root->right,x);

**// Кількість >v**

int countGreater(NodeT\* root,int v){

if(!root) return 0;

int c=(root->key>v);

return c+countGreater(root->l,v)+countGreater(root->r,v);

}

**// Мінімальний >v**

int minGreater(NodeT\* root,int v){

int res=-1;

while(root){

if(root->key>v){ res=root->key; root=root->l; }

else root=root->r;

}

return res;

}

**// Видалити min і max**

NodeT\* deleteMin(NodeT\* root){

if(!root->l){ NodeT\* r=root->r; delete root; return r; }

root->l=deleteMin(root->l); return root;

}

NodeT\* deleteMax(NodeT\* root){

if(!root->r){ NodeT\* l=root->l; delete root; return l; }

root->r=deleteMax(root->r); return root;

}

## 

## 

## **6. Графи (матриця + список)**

const int N=100;

int g[N][N];

**// Перевірка зв’язності BFS**

bool connected(int n){

bool vis[N]={0}; Queue q; enqueue(q,0); vis[0]=1; int cnt=1;

while(q.c>0){

int u=dequeue(q);

for(int v=0;v<n;v++) if(g[u][v]&&!vis[v]){ vis[v]=1;enqueue(q,v);cnt++; }

}

return cnt==n;

}

**// Кістякове дерево (BFS)**

void spanningTree(int n){

bool vis[N]={0}; Queue q; enqueue(q,0); vis[0]=1;

while(q.c>0){

int u=dequeue(q);

for(int v=0;v<n;v++) if(g[u][v]&&!vis[v]){

cout<<u<<"-"<<v<<endl;

vis[v]=1; enqueue(q,v);

}

}

}

**// Ступінь вершини**

int degree(int v,int n){ int d=0;for(int i=0;i<n;i++)d+=g[v][i];return d; }

**// матриця суміжності**

const int V=100;

int g[V][V]; // 0/1

**// Перевірка суміжності u,v**

bool adjacent(int u,int v){ return g[u][v]==1; }

**// Пошук у ширину (BFS)**

void BFS(int start,int n){

bool vis[V]={0};

Queue q; enqueue(q,start); vis[start]=1;

while(!empty(q)){

int u=q.a[q.front]; dequeue(q);

cout<<u<<" ";

for(int v=0;v<n;v++) if(g[u][v]&&!vis[v]){

enqueue(q,v); vis[v]=1;

}

}

}

## **7. Поліноми**

struct Poly { int coef,exp; Poly\* next; };

**// Похідна**

Poly\* derivative(Poly\* p){

Poly\* h=nullptr,\*t=nullptr;

while(p){

if(p->exp>0){

Poly\* n=new Poly{p->coef\*p->exp,p->exp-1,nullptr};

if(!h) h=t=n; else {t->next=n;t=n;}

}

p=p->next;

}

return h;

}

**// Сума двох**

Poly\* add(Poly\* a,Poly\* b){

Poly\* h=nullptr,\*t=nullptr;

while(a||b){

int ce,ex;

if(b==nullptr||(a&&a->exp<b->exp)){ ce=a->coef;ex=a->exp;a=a->next;}

else if(a==nullptr||(b&&b->exp<a->exp)){ ce=b->coef;ex=b->exp;b=b->next;}

else{ ce=a->coef+b->coef;ex=a->exp;a=a->next;b=b->next;}

if(ce!=0){

Poly\* n=new Poly{ce,ex,nullptr};

if(!h) h=t=n; else {t->next=n;t=n;}

}

}

return h;

}