

Contents

1	Basic	
1.1	ascii	
1.2	limits	
1.3	graph	
2	Section2	
2.1	thm	

1 Basic

1.1 ascii

int	char	int	char	int	char
32		64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	\$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(72	H	104	h
41)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[123	{
60	<	92	\	124	
61	=	93]	125	}
62	>	94	^	126	~
63	?	95	_		

1.2 limits

[Type]	[size]	[range]
char	1	127 to -128
signed char	1	127 to -128
unsigned char	1	0 to 255
short	2	32767 to -32768
int	4	2147483647 to -2147483648
unsigned int	4	0 to 4294967295
long	4	2147483647 to -2147483648
unsigned long	4	0 to 18446744073709551615
long long	8	
	9223372036854775807	to -9223372036854775808
double	8	
	1.79769e+308	to 2.22507e-308
long double	16	
	1.18973e+4932	to 3.3621e-4932
float	4	3.40282e+38 to 1.17549e-38
unsigned long long	8	18446744073709551615
string	32	

1.3 graph

```

1
2 #include<bits/stdc++.h>
3 using namespace std;
4
5 class Node {
6 public:
7     int val;
8     vector<Node*> children;
9
10    Node() {}
11
12    Node(int _val) {
13        val = _val;
14    }
15
16    Node(int _val, vector<Node*> _children) {
17        val = _val;
18        children = _children;
19    }
20 };
21
22 struct ListNode {
23     int val;
24     ListNode *next;
25     ListNode() : val(0), next(nullptr) {}
26     ListNode(int x) : val(x), next(nullptr) {}
27     ListNode(int x, ListNode *next) : val(x),
28         next(next) {}
29 };
30
31 struct TreeNode {
32     int val;
33     TreeNode *left;
34     TreeNode *right;
35     TreeNode() : val(0), left(nullptr),
36         right(nullptr) {}
37     TreeNode(int x) : val(x), left(nullptr),
38         right(nullptr) {}
39     TreeNode(int x, TreeNode *left, TreeNode *right)
40         : val(x), left(left), right(right) {}
41 };
42
43 class ListProblem {
44     vector<int> nums={};
45 public:
46     void solve() {
47         return;
48     }
49
50     ListNode* buildList(int idx) {
51         if(idx == nums.size()) return NULL;
52         ListNode *current=new
53             ListNode(nums[idx++],current->next);
54         return current;
55     }
56
57     void deleteList(ListNode* root) {
58         if(root == NULL) return;
59         deleteList(root->next);
60         delete root;
61         return;
62     }
63 };
64
65 class TreeProblem {
66     int null = INT_MIN;
67     vector<int> nums = {}, result;
68 public:
69     void solve() {
70         return;
71     }
72
73     TreeNode* buildBinaryTreeUsingDFS(int left, int
74         right) {

```

```

70     if((left > right) || (nums[(left+right)/2] == 140         inorderTraversal(root->right);
71         null)) return NULL; 141         return;
72     int mid = (left+right)/2; 142     }
73     TreeNode* current = new TreeNode( 143 };
74         nums[mid], 144
75         buildBinaryTreeUsingDFS(left,mid-1), 145 int main() {
76         buildBinaryTreeUsingDFS(mid+1,right)); 146
77     return current; 147     return 0;
78 } 148 }

79     TreeNode* buildBinaryTreeUsingBFS() {
80     int idx = 0;
81     TreeNode* root = new TreeNode(nums[idx++]);
82     queue<TreeNode*> q;
83     q.push(root);
84     while(idx < nums.size()) {
85         if(nums[idx] != null) {
86             TreeNode* left = new
87                 TreeNode(nums[idx]);
88             q.front()->left = left;
89             q.push(left);
90         }
91         idx++;
92         if((idx < nums.size()) && (nums[idx] !=
93             null)) {
94             TreeNode* right = new
95                 TreeNode(nums[idx]);
96             q.front()->right = right;
97             q.push(right);
98         }
99         idx++;
100         q.pop();
101     }
102     return root;
103 }

104 Node* buildNAryTree() {
105     int idx = 2;
106     Node *root = new Node(nums.front());
107     queue<Node*> q;
108     q.push(root);
109     while(idx < nums.size()) {
110         while((idx < nums.size()) && (nums[idx]
111             != null)) {
112             Node *current = new Node(nums[idx++]);
113             q.front()->children.push_back(current);
114             q.push(current);
115         }
116         idx++;
117         q.pop();
118     }
119     return root;
120 }

121 void deleteBinaryTree(TreeNode* root) {
122     if(root->left != NULL)
123         deleteBinaryTree(root->left);
124     if(root->right != NULL)
125         deleteBinaryTree(root->right);
126     delete root;
127     return;
128 }

129 void deleteNAryTree(Node* root) {
130     if(root == NULL) return;
131     for(int i=0; i<root->children.size(); i++) {
132         deleteNAryTree(root->children[i]);
133         delete root->children[i];
134     }
135     delete root;
136     return;
137 }

138 void inorderTraversal(TreeNode* root) {
139     if(root == NULL) return;
140     inorderTraversal(root->left);
141     cout<<root->val<< ' ';

```

2 Section2

2.1 thm

• 中文測試

$$\cdot \sum_{i=1}^n i^2 = \frac{n(n+1)(2n+1)}{6}$$