62

18 string

Contents

```
1 Basic
1.1 limits

2 Basic
2.1 limits

3 Basic
3.1 graph

4 Section2
4.1 thm

...
```

1 Basic

1.1 limits

1	int	char	int	char	int	char
2	32		64	@	96	•
3	33	!	65	Α	97	a
4	34	"	66	В	98	b
5	35	#	67	С	99	C
6	36	\$	68	D	100	d
7	37	%	69	E	101	e
8	38	&	70	F	102	f
9	39	,	71	G	103	g
10	40	(72	Н	104	h
11	41)	73	I	105	i
12	42	*	74	J	106	j
13	43	+	75	K	107	k
14	44	,	76	L	108	1
15	45	-	77	М	109	m
16	46		78	N	110	n
17	47	/	79	0	111	0
18	48	0	80	P	112	p
19	49	1	81	Q	113	q
20	50	2	82	R	114	r
21	51	3	83	S	115	S
22	52	4	84	T	116	t
23	53	5	85	U	117	и
24	54	6	86	V	118	V
25	55	7	87	W	119	W
26	56	8	88	X	120	X
27	57	9	89	Y	121	y
28	58	:	90	Z	122	Z
29	59	;	91	Γ	123	{
30	60	<	92	1	124	1
31	61	=	93	J	125	}
32	62	>	94	٨	126	~
33	63	?	95	_		

2 Basic

2.1 limits

```
1 [Type]
                       [size]
                                 [range]
2 char
                                 127 to -128
3 signed char
                        1
                                 127 to -128
4 unsigned char
                        1
                                 0 to 255
5 short
                        2
                                 32767 to -32768
                                 2147483647 to -2147483648
6 int
7 unsigned int
                                 0 to 4294967295
                                 2147483647 to -2147483648
8 long
9
  unsigned long
                        4
                                 0 to 18446744073709551615
10 long long
                       8
11
              9223372036854775807 to -9223372036854775808
12
  double
                        8
                              1.79769e+308 to 2.22507e-308
13
14
  long double
15
                             1.18973e+4932 to 3.3621e-4932
16 float
                                3.40282e+38 to 1.17549e-38
```

```
3 Basic
```

17 unsigned long long

8

32

18446744073709551615

3.1 graph

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

vector<int> nums = {}, result;

134

135

136

139

141

142

143

144

146

};

```
63
   public:
        void solve() {
                                                                 133
64
65
66
            return:
67
       }
68
                                                                 137
        TreeNode* buildBinaryTreeUsingDFS(int left, int
69
                                                                 138
            right) {
            if((left > right) || (nums[(left+right)/2] ==
70
                                                                 140
                 null)) return NULL;
71
            int mid = (left+right)/2;
            TreeNode* current = new TreeNode(
72
73
                 nums[mid],
                 buildBinaryTreeUsingDFS(left,mid-1),
                                                                 145
74
75
                 buildBinaryTreeUsingDFS(mid+1, right));
                                                                 147
76
            return current;
77
       }
                                                                 148 }
78
        TreeNode* buildBinaryTreeUsingBFS() {
79
80
            int idx = 0;
            TreeNode* root = new TreeNode(nums[idx++]);
81
82
            queue < TreeNode *> q;
83
            q.push(root);
            while(idx < nums.size()) {</pre>
84
85
                 if(nums[idx] != null) {
                     TreeNode* left = new
86
                          TreeNode(nums[idx]);
87
                     q.front()->left = left;
                     q.push(left);
88
89
                 }
                 idx++:
90
91
                 if((idx < nums.size()) && (nums[idx] !=</pre>
                     null)) {
92
                     TreeNode* right = new
                          TreeNode(nums[idx]);
93
                     q.front()->right = right;
94
                     q.push(right);
                 }
95
                 idx++;
96
97
                 q.pop();
            }
98
99
            return root;
       }
100
101
        Node* buildNAryTree() {
102
            int idx = 2;
103
104
            Node *root = new Node(nums.front());
            queue < Node *> q;
105
106
            q.push(root);
            while(idx < nums.size()) {</pre>
107
108
                 while((idx < nums.size()) && (nums[idx]</pre>
                     != null)) {
109
                     Node *current = new Node(nums[idx++]);
110
                     q.front()->children.push_back(current);
                     q.push(current);
111
                 }
112
                 idx++;
113
                 q.pop();
114
115
            }
116
            return root;
117
        }
118
        void deleteBinaryTree(TreeNode* root) {
119
            if(root->left != NULL)
120
                 deleteBinaryTree(root->left);
121
            if(root->right != NULL)
                 deleteBinaryTree(root->right);
            delete root;
122
123
            return;
       }
124
125
126
        void deleteNAryTree(Node* root) {
127
            if(root == NULL) return;
            for(int i=0; i<root->children.size(); i++) {
128
129
                 deleteNAryTree(root->children[i]);
130
                 delete root->children[i];
            }
131
```

Section2

return 0;

int main() {

delete root;

void inorderTraversal(TreeNode* root) {

inorderTraversal(root->left);

inorderTraversal(root->right);

if(root == NULL) return;

cout << root -> val << ' ';

return;

return;

}

4.1 thm

中文測試

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