```c
/* BTreeTest.c */
#include <stdio.h>
#include <string.h>
#include <malloc.h>
#include <stdlib.h>
#include "BTree.h"

static int IntCmp(const void *keyAddr, const void *dataAddr)
{
    int *p1 = (int *)keyAddr;
    int *p2 = (int *)dataAddr;
    return (*p1 - *p2);
}

static void IntTraverse(void *keyAddr, void *outData)
{
    int *p = (int *)keyAddr;
    printf("%d\n", *p);
}

static int StringCmp(const void *keyAddr, const void *dataAddr)
{
    char *p1 = *(char **)keyAddr;
    char *p2 = *(char **)dataAddr;
    return strcmp(p1, p2);
}

static void StringTraverse(void *keyAddr, void *outData)
{
    char *p = *(char **)keyAddr;
    printf("%s\n", p);
}

static void StringFree(void *keyAddr)
{
    char *p = *(char **)keyAddr;
    free(p);
}

int main()
{
    BTREE intBTree;
    BTreeNew(&intBTree, 3, sizeof(int), IntCmp, NULL);
    printf("order of intBTree = %d\n", BTreeOrder(&intBTree));
    int i = 0;
    for (; i < 10; i++)
    {
        BTreeInsert(&intBTree, &i);
    }
    if (!BTreeEmpty(&intBTree))
    {
        printf("size of intBtree = %d\n", BTreeSize(&intBTree));
        BTreeTravLevel(&intBTree, IntTraverse, NULL);
    }
    int intRemove = 1;
    if (0 == BTreeRemove(&intBTree, &intRemove))
    {
        printf("intBTree remove key %d success\n", intRemove);
    }
    else
    {
        printf("intBTree remove key %d fail\n", intRemove);
    }
    if (0 == BTreeRemove(&intBTree, &intRemove))
    {
        printf("intBTree remove key %d success\n", intRemove);
    }
    else
    {
        printf("intBTree remove key %d fail\n", intRemove);
    }
    if (!BTreeEmpty(&intBTree))
    {
```

```
74          printf("size of intBtree = %d\n", BTreeSize(&intBTree));
75          BTreeTravLevel(&intBTree, IntTraverse, NULL);
76      }
77      int intSearch = 2;
78      if (NULL != BTreeSearch(&intBTree, &intSearch))
79      {
80          printf("data %d is in intBTree\n", intSearch);
81      }
82      else
83      {
84          printf("data %d is not in intBTree\n", intSearch);
85      }
86      intSearch = 11;
87      if (NULL != BTreeSearch(&intBTree, &intSearch))
88      {
89          printf("data %d is in intBTree\n", intSearch);
90      }
91      else
92      {
93          printf("data %d is not in intBTree\n", intSearch);
94      }
95      BTreeDispose(&intBTree);
96      printf("\n\n");
97      BTREE stringBTree;
98      BTreeNew(&stringBTree, 3, sizeof(char *), StringCmp, StringFree);
99      char *name1 = strdup("pc");
100     char *name2 = strdup("pcwl513");
101     char *name3 = strdup("pcpc");
102     char *name4 = strdup("jerry");
103     char *name5 = strdup("jerry.peng");
104     char *name6 = strdup("yanglupu");
105     char *name7 = strdup("zhanglei");
106     char *name8 = strdup("lishanke");
107     BTreeInsert(&stringBTree, &name1);
108     BTreeInsert(&stringBTree, &name2);
109     BTreeInsert(&stringBTree, &name3);
110     BTreeInsert(&stringBTree, &name4);
111     BTreeInsert(&stringBTree, &name5);
112     BTreeInsert(&stringBTree, &name6);
113     BTreeInsert(&stringBTree, &name7);
114     BTreeInsert(&stringBTree, &name8);
115     if (!BTreeEmpty(&stringBTree))
116     {
117         printf("size of stringBTree = %d\n", BTreeSize(&stringBTree));
118         BTreeTravLevel(&stringBTree, StringTraverse, NULL);
119     }
120     char *strRemove = "pcpc";
121     if (0 == BTreeRemove(&stringBTree, &strRemove))
122     {
123         printf("stringBTree remove key %s success\n", strRemove);
124     }
125     else
126     {
127         printf("stringBTree remove key %s fail\n", strRemove);
128     }
129     if (0 == BTreeRemove(&stringBTree, &strRemove))
130     {
131         printf("stringBTree remove key %s success\n", strRemove);
132     }
133     else
134     {
135         printf("stringBTree remove key %s fail\n", strRemove);
136     }
137     if (!BTreeEmpty(&stringBTree))
138     {
139         printf("size of stringBTree = %d\n", BTreeSize(&stringBTree));
140         BTreeTravLevel(&stringBTree, StringTraverse, NULL);
141     }
142     char *strSearch = "yanglupu";
143     if (NULL != BTreeSearch(&stringBTree, &strSearch))
144     {
145         printf("data %s is in stringBTree\n", strSearch);
146     }
```

```
147         else
148         {
149             printf("data %s is not in stringBTree\n", strSearch);
150         }
151         strSearch = "123";
152         if (NULL != BTreeSearch(&stringBTree, &strSearch))
153         {
154             printf("data %s is in stringBTree\n", strSearch);
155         }
156         else
157         {
158             printf("data %s is not in stringBTree\n", strSearch);
159         }
160         BTreeDispose(&stringBTree);
161         return 0;
162     }
```