Санкт-Петербургский государственный политехнический университет
Кафедра “Прикладная математика”

Отчет по лабораторной лаботе 4

“Алгоритмы и сруктуры данных”

Студент группы № 5030102/20002

ФИО Двас Павел Григорьевич

Выполнил (дата) 01.12.2023

Оглавление

Постановка задачи

Написать самообучающуюся экспертную систему

Описание алгоритма

Для описания алгоритма требуется пояснить как работает бинарное дерево

Бинарное дерево представляет собой иерархическую структуру данных, состоящую из узлов, каждый из которых имеет не более двух потомков: левого и правого. Узлы содержат значения или ключи, а структура дерева обеспечивает эффективный поиск, вставку и удаление данных, основанный на их отношениях относительно корневого узла.

Представим дерево в качестве структуры

```
typedef struct tagTREE TREE;
struct tagTREE
{
  char Text[MAX_STR];
  TREE *No, *Yes;
};
```

Где Text – значение элемента, *No, *Yes – потомки

Добавление элемента

Для добавления нового узла необходимо вставить новую ветку в правильного ребенка

Очистка дерева

Для очистки дерева необходимо рекурсивно очистить всех детей, начиная с корня

Текст программы

main.c:

```
#include <stdio.h>
#include <conio.h>

#include "tree.h"

void GetStr( char *Str, int MaxLen )
{
  int i = 0;
  char ch;

  while ((ch = getchar()) != '\n')
```

```
    if (Str != NULL && i < MaxLen - 1)
      Str[i++] = ch;
  if (Str != NULL && i < MaxLen)
    Str[i] = 0;
}

void main( void )
{
  char FileName[MAX_STR];
  TREE *ExSys;

  SetDbgMemHooks();

  ExSys = Init("pig", NULL, NULL);

  system("chcp 1251");

  while (1)
    switch (_getch())
    {
    case '0':
      ClearTree(&ExSys);
      exit(30);
      break;
    case '1':
      Session(&ExSys);
      break;
    case '2':
      printf("Enter file name: ");
      GetStr(FileName, MAX_STR);
      SaveTree(FileName, ExSys);
      break;
    case '3':
      printf("Enter file name: ");
      GetStr(FileName, MAX_STR);
      LoadTree(FileName, &ExSys);
      break;
    case '4':
      DrawTree(ExSys);
      printf("\n");
      break;
    }
  ClearTree(&ExSys);
}
```

tree.c:

```
#include <stdio.h>
#include <string.h>
#include <conio.h>

#include "tree.h"

TREE * Init( char *Text, TREE *N, TREE *Y )
{
  TREE *T = malloc(sizeof(TREE));

  if (T == NULL)
    return NULL;

  strncpy(T->Text, Text, MAX_STR - 1);
  T->No = N;
  T->Yes = Y;
  return T;
}

void DrawTree( TREE *T )
{
  if (T == NULL)
  {
    printf("*");
    return;
  }
  printf("%s(", T->Text);
```

```
  DrawTree(T->Yes);
  printf(",");
  DrawTree(T->No);
  printf(")");
}

void Session( TREE **T )
{
  char key, Qst[MAX_STR], Ans[MAX_STR];

  if (*T == NULL)
    return;

  while (1)
  {
    printf("%s?\n", (*T)->Text);
    key = _getch();
    printf(key == 'Y' || key == 'y' ? "Yes\n" : "No\n");
    if ((key == 'Y' || key == 'y') && ((*T)->Yes == NULL))
    {
      printf("Your answer is: %s\n", (*T)->Text);
      return;
    }
    if ((*T)->Yes != NULL)
      key == 'Y' || key == 'y' ? (T = &(*T)->Yes) : (T = &(*T)->No);
    else
    {
      printf("Then enter a new definition:");
      GetStr(Qst, MAX_STR);
      printf("Enter the correct answer:");
      GetStr(Ans, MAX_STR);
      *T = Init(Qst, *T, Init(Ans, NULL, NULL));
      return;
    }
  }
}

void SaveTreeRec( FILE *F, TREE *T )
{
  if (T == NULL)
    return;
  fwrite(T, sizeof(TREE), 1, F);
  SaveTreeRec(F, T->No);
  SaveTreeRec(F, T->Yes);
}

int SaveTree( char *FileName, TREE *T )
{
  FILE *F;

  if ((F = fopen(FileName, "wb")) == NULL)
    return 0;

  SaveTreeRec(F, T);
  fclose(F);
  return 1;
}

void LoadTreeRec( FILE *F, TREE **T )
{
  TREE tr;

  if (fread(&tr, sizeof(TREE), 1, F) != 1)
    return;

  *T = Init(tr.Text, NULL, NULL);
  if (tr.No != NULL)
    LoadTreeRec(F, &(*T)->No);
```

```
  if (tr.Yes != NULL)
    LoadTreeRec(F, &(*T)->Yes);
}

int LoadTree( char *FileName, TREE **T )
{
  FILE *F;

  if ((F = fopen(FileName, "rb")) == NULL)
    return 0;

  LoadTreeRec(F, T);
  fclose(F);
  return 1;
}

void ClearTree( TREE **T )
{
  if (*T != NULL)
  {
    ClearTree(&(*T)->Yes);
    ClearTree(&(*T)->No);
    free(*T);
    *T = NULL;
  }
}
```

tree.h:

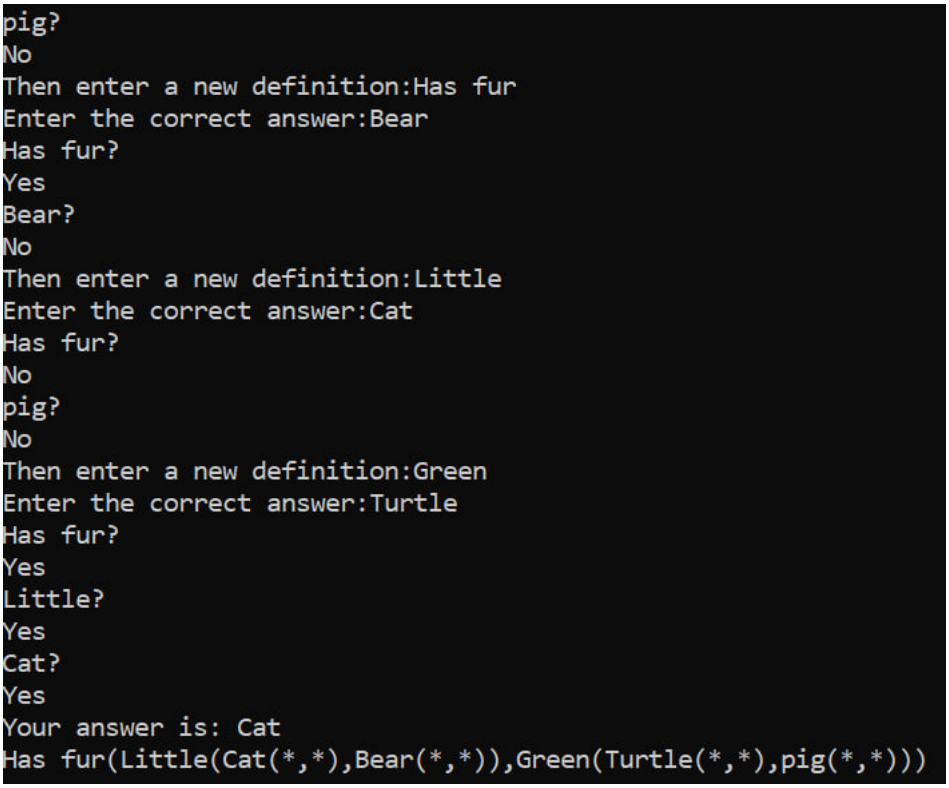
```
#define MAX_STR 102

typedef struct tagTREE TREE;
struct tagTREE
{
  char Text[MAX_STR];
  TREE *No, *Yes;
};

void GetStr( char *Str, int MaxLen );

void Session( TREE **T );

TREE * Init( char *Text, TREE *N, TREE *Y );

int SaveTree( char *FileName, TREE *T );
int LoadTree( char *FileName, TREE **T );

void DrawTree( TREE *T );
void ClearTree( TREE **T );
```

Описание тестирования

Для тестирования данной программы производился ее неоднократный запуск с вводом конкретных данных, при котором проверялось:

1) Стабильность работы программы при одинаковых входных данных;
2) Отсутствие “падений” и “зависаний”;
3) Корректное выполнение всех заявленных процедур;
4) Корректное завершение программы;

Пример тестирования:



При проведении тестирования такого рода никаких проблем обнаружено не было, что позволяет судить о корректности работы программы в целом.