

**课 程 实 验 报 告**

**课程名称： 数据结构实验**

**专业班级： 计算机科学与技术1601班**

**学 号： U201614531**

**姓 名： 刘本嵩**

**指导教师： 周时阳**

**报告日期： 2018年 1月 6日**

**计算机科学与技术学院**

**目 录**

[1 基于顺序存储结构的线性表实现 4](#_Toc1363093817)

[1.1 问题描述 4](#_Toc2084263998)

[1.2 系统设计 5](#_Toc2144917531)

[1.2.1 系统总体设计 5](#_Toc1024438400)

[1.2.2 有关常量和类型定义 6](#_Toc1970817169)

[1.2.3 算法设计 7](#_Toc1730265926)

[1.3 顺序表演示系统实现与测试 9](#_Toc851316344)

[1.3.1 系统实现 9](#_Toc1883946786)

[1.3.2 系统测试 23](#_Toc607075315)

[1.4 实验小结 34](#_Toc1948272577)

[2基于链式存储结构的线性表实现 34](#_Toc1388606817)

[2.1 实验目的 35](#_Toc1527835141)

[2.2 线性表基本运算定义 35](#_Toc1542264332)

[2.2 系统设计 36](#_Toc364765655)

[2.2.1 系统总体设计 36](#_Toc1874943628)

[2.2.2 算法设计 37](#_Toc1587004216)

[2.3 顺序表演示系统实现与测试 40](#_Toc58639531)

[2.3.1 系统实现 40](#_Toc151850257)

[2.3.2 算法测试 63](#_Toc1855857490)

[2.3.3 界面测试 64](#_Toc200838680)

[2.4 实验小结 64](#_Toc1194504771)

[3基于二叉链表的二叉树实现 64](#_Toc523828699)

[3.1 实验目的 65](#_Toc2053117396)

[3.2 系统设计 65](#_Toc1596482584)

[3.2.1 系统总体设计 65](#_Toc75446518)

[3.2.2 算法设计 66](#_Toc1251593857)

[3.3 二叉树演示系统实现与测试 68](#_Toc1397545139)

[3.3.1 系统实现 69](#_Toc1753430851)

[3.3.2 算法测试 88](#_Toc912648935)

[3.3.3 界面测试 90](#_Toc167181173)

[3.4 实验小结 90](#_Toc2138554204)

[4基于邻接表的图实现 91](#_Toc128259104)

[4.1 实验目的 91](#_Toc103961523)

[4.2 系统设计 91](#_Toc2135988088)

[4.2.1 系统总体设计 92](#_Toc1152697504)

[4.2.2 算法设计 93](#_Toc2074778692)

[4.3 图演示系统实现与测试 95](#_Toc1718770366)

[4.3.1 系统实现 95](#_Toc2004013848)

[4.3.2 实验中多次使用的rlib库在此时的最终版本 123](#_Toc1811241830)

[4.3.3 算法测试 171](#_Toc178362033)

[4.3.4 界面测试 174](#_Toc1804802778)

[4.4 实验小结 174](#_Toc1052365000)

[参考文献 175](#_Toc1706197174)

**1 基于顺序存储结构的线性表实现**

**1.1 问题描述**

线性表在物理内存中可以以顺序表的方式实现，即物理上存储位置相邻的两个元素是线性表中的相邻元素，且数据元素的前后关系不变。

实验要完成的顺序表算法：

⑴初始化表：函数名称是InitaList(L)；初始条件是线性表L不存在已存在；操作结果是构造一个空的线性表。

⑵销毁表：函数名称是DestroyList(L)；初始条件是线性表L已存在；操作结果是销毁线性表L。

⑶清空表：函数名称是ClearList(L)；初始条件是线性表L已存在；操作结果是将L重置为空表。

⑷判定空表：函数名称是ListEmpty(L)；初始条件是线性表L已存在；操作结果是若L为空表则返回TRUE,否则返回FALSE。

⑸求表长：函数名称是ListLength(L)；初始条件是线性表已存在；操作结果是返回L中数据元素的个数。

⑹获得元素：函数名称是GetElem(L,i,e)；初始条件是线性表已存在，1≤i≤ListLength(L)；操作结果是用e返回L中第i个数据元素的值。

⑺查找元素：函数名称是LocateElem(L,e,compare())；初始条件是线性表已存在；操作结果是返回L中第1个与e满足关系compare（）关系的数据元素的位序，若这样的数据元素不存在，则返回值为0。

⑻获得前驱：函数名称是PriorElem(L,cur\_e,pre\_e)；初始条件是线性表L已存在；操作结果是若cur\_e是L的数据元素，且不是第一个，则用pre\_e返回它的前驱，否则操作失败，pre\_e无定义。

⑼获得后继：函数名称是NextElem(L,cur\_e,next\_e)；初始条件是线性表L已存在；操作结果是若cur\_e是L的数据元素，且不是最后一个，则用next\_e返回它的后继，否则操作失败，next\_e无定义。

⑽插入元素：函数名称是ListInsert(L,i,e)；初始条件是线性表L已存在且非空，1≤i≤ListLength(L)+1；操作结果是在L的第i个位置之前插入新的数据元素e。

⑾删除元素：函数名称是ListDelete(L,i,e)；初始条件是线性表L已存在且非空，1≤i≤ListLength(L)；操作结果：删除L的第i个数据元素，用e返回其值。

⑿遍历表：函数名称是ListTraverse(L,visit())，初始条件是线性表L已存在；操作结果是依次对L的每个数据元素调用函数visit()。

实验目标：

通过实验达到⑴加深对线性表的概念、基本运算的理解；⑵熟练掌握线性表的逻辑结构与物理结构的关系；⑶物理结构采用顺序表,熟练掌握线性表的基本运算的实现。

**1.2 系统设计**

1.2.1 系统总体设计

本系统采用顺序表作为线性表的物理结构，实现线性表的基本运算。

系统具有一个功能菜单。在主程序中完成函数调用所需实参值的准备和函数执行结果的现实，并给出适当的操作提示显示。

系统通过定义一个SqLists类型的含有线性表指针数组和当前线性表数量的结构体，并声明一个此类型的全局结构变量Lists，每当创建或者删除一个线性表，则修改此变量，实现对多个线性表的管理。

系统开始运行时调用函数读取文件中的数据，并提供数据保存功能以实现线性表的文件形式保存。

该演示系统提供的操作有：表的初始化、销毁、清空、判空，求表长、获取数据元素、查找数据元素、获得前驱、获得后继、插入数据元素、删除数据元素、表的遍历、表的选择、数据保存。

在程序中实现消息处理和操作提示，包括数据的输入和输出，错误操作提示、程序的退出。

1.2.2 有关常量和类型定义

#define TRUE 1

#define FALSE 0

#define OK 1

#define ERROR 0

#define INFEASTABLE -1

#define OVERFLOW -2

#define LIST\_INIT\_SIZE 100

#define LISTINCREMENT 10

#define MAX\_SQLIST\_NUM 10 //可创建的线性表最大数量

typedef int status;

typedef int ElemType; //数据元素类型定义

typedef struct{ //线性表（顺序结构）的定义

ElemType \* elem;

int length;

int listsize;

}SqList;

typedef struct{ //线性表信息表结构定义（存储当前所有线性表基址及数量）

SqList \* pSqList[MAX\_SQLIST\_NUM];

int length;

}SqLists;

SqLists Lists; //线性表信息表

1.2.3 算法设计

（1）InitList(SqList \* L)

设计：分配存储空间，并初始化表长为0，表容量为LIST\_INIT\_SIZE。每次创建表时新建的表位序为最大。例如当前有6个表，执行此函数后创建的新表为表7

操作结果：构造一个空的线性表。

（2）DestroyList(SqList \* L)

设计：释放存储空间，每次操作当前线性表，销毁后当前线性表之后的线性表左移一个位序。例如当前操作表2，销毁表2后原表3左移成为表2，以此类推

操作结果：销毁线性表L。

（3）ClearList(SqList \* L)

设计：线性表L的长度赋值为0

操作结果：将L重置为空表。

（4）ListEmpty(SqList L)

设计：根据表长判断表是否为空

操作结果：若L为空表，则返回TRUE,否则返回FALSE。

（5）ListLength(SqList L)

设计：返回表长

操作结果：返回L中数据元素的个数。

（6）GetElem(SqList L, int i, ElemType \* e)

设计：根据位序找到第i个元素的地址并将其值赋值给指针e指向的元素

操作结果：用指针e指向的元素返回L中第i个数据元素的值。

（7）LocateElem(SqList L, ElemType e)

设计：遍历线性表找到第一个和元素e的相等的元素

操作结果：返回L中第1个与e相等的的数据元素的位序，若这样的数据元素不存在，则返回值为0。

（8）PriorElem(SqList L, ElemType cur, ElemType \* pre\_e)

设计：遍历线性表找到第一个和元素cur的相等的元素，如果其有前驱，用pre\_e返回，函数返回TRUE；否则函数返回FALSE，pre\_e无意义

操作结果：若cur是L的数据元素，且不是第一个，则用pre\_e返回它的 前驱，否则操作失败，pre\_e无定义。

（9）NextElem（L，cur\_e，&next\_e）

设计：遍历线性表找到第一个和元素cur的相等的元素，如果其有后继，用next\_e返回，函数返回TRUE；否则函数返回FALSE，next\_e无意义

操作结果：若cur是L的数据元素，且不是最后一个，则用next\_e返回它 的后继，否则操作失败，next\_e无定义。

（10）ListInsert(SqList \* L, int i, ElemType e)

设计：如果线性表已满，重新分配存储空间。将线性表指针L指向的线性表第i个元素之后的元素都右移一个位序，之后将e插入第i个位序

操作结果：在L的第i个位置之前插入新的数据元素e，L的长度加1

（11）ListDelete(SqList \* L, int i, ElemType \* e)

设计：将第i个位序的值赋给指针e指向的变量，之后第i个位序之后的元素全部左移一个位序

操作结果：删除L的第i个数据元素，用e返回其值，L的长度减1.

（12）ListTraverse(SqList L)

设计：遍历并输出表L中的每个元素值，返回表长

操作结果：依次输出表L中的每个变量的值

（13）LoadDate(void)

设计：调用CreatList函数读取文件信息并输出信息

（14）CreatList(void)

设计：读取文件信息并创建线性表

操作结果：在内存中重建物理结构代表的线性表数据

**1.3 顺序表演示系统实现与测试**

1.3.1 系统实现

编译环境：Windows下使用mingw-gcc 6.3.1编译，不开启扩展，程序清单如下：

>> File: exp.h

#ifndef EXPERIMENT1\_H

#define EXPERIMENT1\_H

#include <malloc.h>

#include <stdio.h>

#include <stdlib.h>

#define TRUE 1

#define FALSE 0

#define OK 1

#define ERROR 0

#define INFEASTABLE -1

#define OVERFLOW -2

typedef int status;

typedef int ElemType;

#define LIST\_INIT\_SIZE 100

#define LISTINCREMENT 10

#define MAX\_SQLIST\_NUM 10

typedef struct {

ElemType \*elem;

int length;

int listsize;

} SqList;

typedef struct {

SqList \*pSqList[MAX\_SQLIST\_NUM];

int length;

} SqLists;

SqLists Lists;

void LoadDate(void);

status CreatList(void);

status SaveDate(void);

extern char \*gp\_sqlists\_filename;

extern char \*gp\_sqlistdate\_filename;

extern char \*gp\_sqlistelem\_filename;

status InitList(SqList \*L);

SqList \*ChooseList(int \*);

status DestroyList(SqList \*L);

status ClearList(SqList \*L);

status ListEmpty(SqList L);

int ListLength(SqList L);

status GetElem(SqList L, int i, ElemType \*e);

status LocateElem(SqList L, ElemType e);

status PriorElem(SqList L, ElemType cur, ElemType \*pre\_e);

status NextElem(SqList L, ElemType cur, ElemType \*next\_e);

status ListInsert(SqList \*L, int i, ElemType e);

status ListDelete(SqList \*L, int i, ElemType \*e);

status ListTraverse(SqList L);

#endif /\*\*< EXP\_H\*/

file: main.c

#include "exp.h"

char \*gp\_sqlists\_filename = "./sqlists.dat";

char \*gp\_sqlistdate\_filename = "./sqlistdate.dat";

char \*gp\_sqlistelem\_filename = "./sqlistelem.dat";

int main() {

SqLists \*pLists = &Lists;

SqList \*pList = NULL;

size\_t opListNum = 0;

size\_t op = 1;

size\_t i;

size\_t elemNum;

ElemType elem, elem2;

char choise;

LoadDate();

while (op) {

#ifdef WIN32

system("cls");

#else

system("clear");

#endif

printf("\n\t\t\t线性表的顺序实现\n");

printf("\*\*当前线性表表数：%d\n", pLists->length);

printf("\*\*最大线性表数：%d\n", MAX\_SQLIST\_NUM);

printf("\*\*当前操作线性表(编号从1开始)：");

if (opListNum <= 0) {

printf("无，2-12操作前请先选择要操作的线性表或创建表\n");

} else {

printf("线性表%d\n", opListNum);

}

printf(" Menu for Linear Table On Sequence Structure \n");

printf("-------------------------------------------------\n");

printf("      1. InitList 8. PriorElem\n");

printf("      2. DestroyList 9. NextElem\n");

printf("      3. ClearList 10. ListInsert \n");

printf("      4. ListEmpty 11. ListDelete\n");

printf("      5. ListLength 12. ListTraverse\n");

printf("      6. GetElem 13. ChooseList\n");

printf("      7. LocateElem 14. SaveDate\n");

printf("      0. Exit\n");

printf("-------------------------------------------------\n");

printf(" 请选择你的操作[0~14]:");

scanf("%d", &op);

getchar();

switch (op) {

case 1:

if (pLists->length >= MAX\_SQLIST\_NUM) {

printf("线性表数达到最大，不允许创建新表!\n");

getchar();

break;

}

if (!(pLists->pSqList[pLists->length++] =

(SqList \*)malloc(sizeof(SqList))))

exit(OVERFLOW); //??? Shit copied from text book

if (InitList(pLists->pSqList[pLists->length - 1]) == OK)

printf("线性表创建成功！\n\n回车以继续进行下一步操作\n");

else {

printf("线性表创建失败！\n\n回车以继续进行下一步操作\n");

free(pLists->pSqList[pLists->length - 1]);

}

getchar();

break;

case 2:

if (opListNum < 1) {

printf("请先选择需要操作的线性表！\n");

getchar();

break;

}

if (DestroyList(pList))

printf("线性表销毁成功！\n\n回车以继续进行下一步操作\n");

if (opListNum < pLists->length) {

for (i = opListNum; i < pLists->length - 1; i++)

pLists->pSqList[i - 1] = pLists->pSqList[i];

}

pLists->length--;

opListNum = 0;

getchar();

break;

case 3:

if (opListNum < 1) {

printf("请先选择需要操作的线性表！\n");

getchar();

break;

}

if (ClearList(pList))

printf("线性表置空成功！\n\n回车以继续进行下一步操作\n");

getchar();

break;

case 4:

if (opListNum < 1) {

printf("请先选择需要操作的线性表！\n");

getchar();

break;

}

if (ListEmpty(\*pList))

printf("线性表为空\n\n回车以继续进行下一步操作\n");

else

printf("线性表非空\n\n回车以继续进行下一步操作\n");

getchar();

break;

case 5:

if (opListNum < 1) {

printf("请先选择需要操作的线性表！\n");

getchar();

break;

}

printf("线性表表长：%d\n\n回车以继续进行下一步操作\n",

ListLength(\*pList));

getchar();

break;

case 6:

if (opListNum < 1) {

printf("请先选择需要操作的线性表！\n");

getchar();

break;

}

printf("请输入要查找的元素序号:");

scanf("%d", &elemNum);

getchar();

if (elemNum >= 1 && elemNum <= pList->length) {

GetElem(\*pList, elemNum, &elem);

printf("元素值：%d\n\n回车以继续进行下一步操作\n", elem);

} else

printf("元素不存在\n\n回车以继续进行下一步操作\n");

getchar();

break;

case 7:

if (opListNum < 1) {

printf("请先选择需要操作的线性表！\n");

getchar();

break;

}

printf("请输入要查找的元素值：");

scanf("%d", &elem);

getchar();

if ((elemNum = LocateElem(\*pList, elem)))

printf("位序:%d\n\n回车以继续进行下一步操作\n", elemNum);

else

printf("元素不存在\n\n回车以继续进行下一步操作\n");

getchar();

break;

case 8:

if (opListNum < 1) {

printf("请先选择需要操作的线性表！\n");

getchar();

break;

}

printf("请输入要查找的元素值：");

scanf("%d", &elem);

getchar();

if (PriorElem(\*pList, elem, &elem2))

printf("前驱结点元素值:%d\n\n回车以继续进行下一步操作\n", elem2);

else

printf("前驱不存在!\n\n回车以继续进行下一步操作\n");

getchar();

break;

case 9:

if (opListNum < 1) {

printf("请先选择需要操作的线性表！\n");

getchar();

break;

}

printf("请输入要查找的元素值：");

scanf("%d", &elem);

getchar();

if (NextElem(\*pList, elem, &elem2))

printf("后继结点元素值:%d\n\n回车以继续进行下一步操作\n", elem2);

else

printf("后继不存在!\n\n回车以继续进行下一步操作\n");

getchar();

break;

case 10:

if (opListNum < 1) {

printf("请先选择需要操作的线性表！\n");

getchar();

break;

}

printf("请输入要插入的元素值：");

scanf("%d", &elem);

getchar();

printf("请输入要插入位序：");

scanf("%d", &elemNum);

getchar();

if (ListInsert(pList, elemNum, elem))

printf("插入成功！\n\n回车以继续进行下一步操作\n");

else

printf("插入失败！\n\n回车以继续进行下一步操作\n");

getchar();

break;

case 11:

if (opListNum < 1) {

printf("请先选择需要操作的线性表！\n");

getchar();

break;

}

printf("请输入要删除的位序：");

scanf("%d", &elemNum);

getchar();

if (ListDelete(pList, elemNum, &elem)) {

printf("\n删除的元素值：%d\n", elem);

printf("删除成功！\n\n回车以继续进行下一步操作\n");

} else

printf("删除失败！\n\n回车以继续进行下一步操作\n");

getchar();

break;

case 12:

if (opListNum < 1) {

printf("请先选择需要操作的线性表！\n");

getchar();

break;

}

if (!ListTraverse(\*pList))

printf("线性表是空表！\n\n回车以继续进行下一步操作\n");

getchar();

break;

case 13:

pList = ChooseList(&opListNum);

printf("当前线性表：表%d\n\n回车以继续进行下一步操作\n", opListNum);

getchar();

break;

case 14:

if (SaveDate()) {

printf("数据保存成功！\n\n回车以继续进行下一步操作\n");

} else {

printf("数据保存失败！\n\n回车以继续进行下一步操作\n");

}

getchar();

}

}

printf("保存数据?(Y/N)\n");

scanf("%1s", &choise);

getchar();

if (choise == 'Y' || choise == 'y') {

if (SaveDate())

printf("数据保存成功！\n");

else

printf("数据保存失败！\n");

}

printf("欢迎下次再使用本系统！\n");

getchar();

return 0;

}

>> File: impl.c

#include "experiment1.h"

SqList \*ChooseList(size\_t \*popListNum) {

size\_t choice;

printf("\n输入编号(1-");

printf("%d):", Lists.length);

scanf("%d", &choice);

getchar();

if (choice < 1 || choice > Lists.length) {

printf("\n表不存在\n");

return NULL;

}

\*popListNum = choice;

printf("成功\n");

return Lists.pSqList[choice - 1];

}

status ClearList(SqList \*L) {

L->length = 0;

return OK;

}

status CreatList(void) {

FILE \*pFILE;

SqLists \*pLists = &Lists;

SqList \*p\_sqList;

size\_t re = 0;

size\_t ListsLength = 0;

size\_t i;

if ((pFILE = fopen(gp\_sqlists\_filename, "rb+")) == NULL) {

fopen(gp\_sqlists\_filename, "wb+");

if ((pFILE = fopen(gp\_sqlists\_filename, "rb+")) == NULL) {

printf("信息文件打开失败！\n");

return re;

}

}

printf("信息文件打开成功！\n");

if (fread(pLists, sizeof(SqLists), 1, pFILE)) {

re += 4;

printf("信息文件加载成功！\n");

ListsLength = pLists->length;

}

for (i = 0; i < MAX\_SQLIST\_NUM; i++) {

pLists->pSqList[i] = NULL;

}

pLists->length = 0;

fclose(pFILE);

if ((pFILE = fopen(gp\_sqlistdate\_filename, "rb+")) == NULL) {

fopen(gp\_sqlistdate\_filename, "wb+");

if ((pFILE = fopen(gp\_sqlistdate\_filename, "rb+")) == NULL) {

printf("数据文件打开失败!\n");

return re;

}

}

printf("数据文件打开成功!\n");

for (i = 0; i < ListsLength; i++) {

pLists->pSqList[i] = (SqList \*)malloc(sizeof(SqList));

p\_sqList = pLists->pSqList[i];

if ((fread(p\_sqList, sizeof(SqList), 1, pFILE)) == 0) {

printf("数据文件不完整！\n");

free(p\_sqList);

return re;

}

p\_sqList->elem = (ElemType \*)malloc(LIST\_INIT\_SIZE \* sizeof(ElemType));

pLists->length++;

}

re += 8;

printf("数据文件加载成功！\n");

fclose(pFILE);

if ((pFILE = fopen(gp\_sqlistelem\_filename, "rb+")) == NULL) {

fopen(gp\_sqlistelem\_filename, "wb+");

if ((pFILE = fopen(gp\_sqlistdate\_filename, "rb+")) == NULL) {

printf("elem数据文件打开失败!\n");

return re;

}

}

printf("elem数据文件打开成功!\n");

for (i = 0; i < ListsLength; i++) {

if ((fread(pLists->pSqList[i]->elem, sizeof(ElemType),

pLists->pSqList[i]->length, pFILE)) == 0) {

printf("elem数据文件不完整！\n");

return re;

}

}

re += 16;

printf("elem数据文件加载成功！\n");

fclose(pFILE);

return re;

}

status DestroyList(SqList \*L) {

free(L->elem);

free(L);

return OK;

}

status GetElem(SqList L, size\_t i, ElemType \*e) {

\*e = L.elem[i - 1];

return OK;

}

status InitList(SqList \*L) {

L->elem = (ElemType \*)malloc(LIST\_INIT\_SIZE \* sizeof(ElemType));

if (!L->elem)

exit(OVERFLOW);

L->length = 0;

L->listsize = LIST\_INIT\_SIZE;

return OK;

}

status ListDelete(SqList \*L, size\_t i, ElemType \*e) {

ElemType \*p, \*q;

if (i < 1 || i > L->length)

return ERROR;

p = &L->elem[i - 1];

\*e = \*p;

for (q = &L->elem[L->length - 1]; p < q; p++) {

\*p = \*(p + 1);

}

L->length--;

return OK;

}

status ListEmpty(SqList L) { return L.length == 0; }

status ListInsert(SqList \*L, size\_t i, ElemType e) {

ElemType \*newbase, \*p, \*q;

if (i < 1 || i > L->length + 1)

return ERROR;

if (L->length >= L->listsize) {

newbase = (ElemType \*)realloc(L->elem, (L->listsize + LISTINCREMENT) \*

sizeof(ElemType));

if (!newbase)

exit(OVERFLOW);

L->elem = newbase;

L->listsize += LISTINCREMENT;

}

p = &L->elem[i - 1];

for (q = &L->elem[L->length - 1]; p <= q; q--) {

\*(q + 1) = \*q;

}

\*p = e;

L->length++;

return OK;

}

size\_t ListLength(SqList L) { return L.length; }

status

ListTraverse(SqList L) {

printf("\n-----------all elements -----------------------\n");

for (size\_t i = 0; i < L.length; i++)

printf("%d ", L.elem[i]);

printf("\n------------------ end ------------------------\n");

return L.length;

}

void LoadDate(void) {

size\_t Re = CreatList();

if (Re < 24) {

/\*数据加载提示信息\*/

printf("\n系统数据不完整!\n");

}

printf("\n按回车键继续...\n");

getchar();

return;

}

status LocateElem(SqList L, ElemType e) {

size\_t i;

for (i = 0; i < L.length; i++) {

if (L.elem[i] == e)

return i + 1;

}

return 0;

}

status NextElem(SqList L, ElemType cur, ElemType \*next\_e) {

size\_t i;

for (i = 0; i < L.length - 1; i++) {

if (L.elem[i] == cur) {

\*next\_e = L.elem[i + 1];

return OK;

}

}

return ERROR;

}

status PriorElem(SqList L, ElemType cur, ElemType \*pre\_e) {

size\_t i;

for (i = 1; i < L.length; i++) {

if (L.elem[i] == cur) {

\*pre\_e = L.elem[i - 1];

return OK;

}

}

return ERROR;

}

status SaveDate(void) {

FILE \*pFILE1, \*pFILE2;

SqLists \*pLists = &Lists;

size\_t i;

pFILE1 = fopen(gp\_sqlists\_filename, "wb");

fwrite(pLists, sizeof(SqLists), 1, pFILE1);

fclose(pFILE1);

pFILE1 = fopen(gp\_sqlistdate\_filename, "wb");

pFILE2 = fopen(gp\_sqlistelem\_filename, "wb");

for (i = 0; i < pLists->length; i++) {

fwrite(pLists->pSqList[i], sizeof(SqList), 1, pFILE1);

fwrite(pLists->pSqList[i]->elem, sizeof(ElemType),

pLists->pSqList[i]->length, pFILE2);

}

fclose(pFILE1);

fclose(pFILE2);

return OK;

}

1.3.2 系统测试

测试数据

表1：

1 2 3 4 5

表2

5 4 3 2 1

表3

8 8 8 8 8 8

表4

0 0 0 0 0 0

表5

5 6 2 45 3 25 4 85 69

表6

5 1 2 23

测试用例及其结果如下（各函数测试为独立测试，测试初始数据相同，不受上个函数测试影响）：

1. 测试函数：ChooseList

测试步骤及结果如表1-1所示

表1-1 ChooseList函数测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试步骤 | 测试输入 | 理论结果 | 运行结果 |
| 1 | 1.主界面输入13进入函数  2.按提示输入要操作的线性表序号，输入2 | 输出“操作成功！当前线性表：表2”按回车后当前操作线性表更新为2 | 输出“操作成功！当前线性表：表2”按回车后当前操作线性表更新为2 |
| 2 | 主界面输入12进入函数 | 输出“5 4 3 2 1” | 输出“5 4 3 2 1” |

1. 测试函数：DestroyList

测试步骤及结果如表1-2所示

表1-2 DestroyList函数测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试步骤 | 测试输入 | 理论结果 | 运行结果 |
| 1 | 1.主界面输入13进入函数  2.按提示输入要操作的线性表序号，输入2 | 输出“操作成功！当前线性表：表2” 按回车后当前操作线性表更新为2 | 输出“操作成功！当前线性表：表2”按回车后当前操作线性表更新为2 |
| 2 | 主界面输入12进入函数 | 输出“5 4 3 2 1” | 输出“5 4 3 2 1” |
| 3 | 主界面输入2进入函数 | 输出“线性表销毁成功！”按回车后当前线性表数更新为5，当前操作线性表更新为无 | 输出“线性表销毁成功！”按回车后当前线性表数更新为5，当前操作线性表更新为无 |
| 4 | 1.主界面输入13进入函数  2.按提示输入要操作的线性表序号，输入2 | 输出“操作成功！当前线性表：表2” 按回车后当前操作线性表更新为2 | 输出“操作成功！当前线性表：表2”按回车后当前操作线性表更新为2 |
| 5 | 主界面输入12进入函数 | 原表3变为表2，输出“8 8 8 8 8 8” | 输出“8 8 8 8 8 8” |

1. 测试函数：ClearList

测试步骤及结果如表1-3所示

表1-3 ClearList函数测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试步骤 | 测试输入 | 理论结果 | 运行结果 |
| 1 | 1.主界面输入13进入函数  2.按提示输入要操作的线性表序号，输入2 | 输出“操作成功！当前线性表：表2” 按回车后当前操作线性表更新为2 | 输出“操作成功！当前线性表：表2”按回车后当前操作线性表更新为2 |
| 2 | 主界面输入4进入函数 | 输出“线性表非空” | 输出“线性表非空” |
| 3 | 主界面输入3进入函数 | 输出“线性表置空成功！” | 输出“线性表置空成功！” |
| 4 | 主界面输入4进入函数 | 输出“线性表为空” | 输出“线性表为空” |

1. 测试函数：ListEmpty

测试步骤及结果如表1-4所示

表1-4 ListEmpty函数测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试步骤 | 测试输入 | 理论结果 | 运行结果 |
| 1 | 1.主界面输入13进入函数  2.按提示输入要操作的线性表序号，输入2 | 输出“操作成功！当前线性表：表2” 按回车后当前操作线性表更新为2 | 输出“操作成功！当前线性表：表2”按回车后当前操作线性表更新为2 |
| 2 | 主界面输入12进入函数 | 输出“5 4 3 2 1” | 输出“5 4 3 2 1” |
| 3 | 主界面输入4进入函数 | 输出“线性表非空” | 输出“线性表非空” |
| 4 | 主界面输入3进入函数 | 输出“线性表置空成功！” | 输出“线性表置空成功！” |
| 5 | 主界面输入12进入函数 | 输出“线性表是空表！” | 输出“线性表是空表！” |
| 6 | 主界面输入4进入函数 | 输出“线性表为空” | 输出“线性表为空” |

1. 测试函数：ListLength

测试步骤及结果如表1-5所示

表1-5 ListLength函数测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试步骤 | 测试输入 | 理论结果 | 运行结果 |
| 1 | 1.主界面输入13进入函数  2.按提示输入要操作的线性表序号，输入3 | 输出“操作成功！当前线性表：表3” 按回车后当前线性表表数更新为3 | 输出“操作成功！当前线性表：表3” 按回车后当前线性表表数更新为3 |
| 2 | 主界面输入12进入函数 | 输出“8 8 8 8 8 8” | 输出“8 8 8 8 8 8” |
| 3 | 主界面输入5进入函数 | 输出“线性表表长：6” | 输出“线性表表长：6” |

1. 测试函数：GetElem

测试步骤及结果如表1-6所示

表1-6 GetElem函数测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试步骤 | 测试输入 | 理论结果 | 运行结果 |
| 1 | 1.主界面输入13进入函数  2.按提示输入要操作的线性表序号，输入2 | 输出“操作成功！当前线性表：表2” 按回车后当前线性表表数更新为2 | 输出“操作成功！当前线性表：表2” 按回车后当前线性表表数更新为2 |
| 2 | 主界面输入12进入函数 | 输出“5 4 3 2 1” | 输出“5 4 3 2 1” |
| 3 | 1.主界面输入7进入函数  2.按提示输入要查找的元素值，输入4 | 输出“位序:2” | 输出“位序:2” |

1. 测试函数：PriorElem

测试步骤及结果如表1-7所示

表1-7 PriorElem函数测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试步骤 | 测试输入 | 理论结果 | 运行结果 |
| 1 | 1.主界面输入13进入函数  2.按提示输入要操作的线性表序号，输入2 | 输出“操作成功！当前线性表：表2” 按回车后当前线性表表数更新为2 | 输出“操作成功！当前线性表：表2” 按回车后当前线性表表数更新为2 |
| 2 | 主界面输入12进入函数 | 输出“5 4 3 2 1” | 输出“5 4 3 2 1” |
| 3 | 1.主界面输入8进入函数  2.按提示输入要查找的元素值，输入4 | 输出“前驱结点元素值:5” | 输出“前驱结点元素值:5” |
| 4 | 1.主界面输入8进入函数  2.按提示输入要查找的元素值，输入5 | 输出“前驱不存在!” | 输出“前驱不存在!” |

1. 测试函数：NextElem

测试步骤及结果如表1-8所示

表1-8 NextElem函数测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试步骤 | 测试输入 | 理论结果 | 运行结果 |
| 1 | 1.主界面输入13进入函数  2.按提示输入要操作的线性表序号，输入2 | 输出“操作成功！当前线性表：表2” 按回车后当前线性表表数更新为2 | 输出“操作成功！当前线性表：表2” 按回车后当前线性表表数更新为2 |
| 2 | 主界面输入12进入函数 | 输出“5 4 3 2 1” | 输出“5 4 3 2 1” |
| 3 | 1.主界面输入9进入函数  2.按提示输入要查找的元素值，输入4 | 输出“后继结点元素值:3” | 输出“后继结点元素值:3” |
| 4 | 1.主界面输入9进入函数  2.按提示输入要查找的元素值，输入1 | 输出“后继不存在!” | 输出“后继不存在!” |

1. 测试函数：ListInsert

测试步骤及结果如表1-9所示

表1-9 ListInsert函数测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试步骤 | 测试输入 | 理论结果 | 运行结果 |
| 1 | 1.主界面输入13进入函数  2.按提示输入要操作的线性表序号，输入2 | 输出“操作成功！当前线性表：表2” 按回车后当前线性表表数更新为2 | 输出“操作成功！当前线性表：表2” 按回车后当前线性表表数更新为2 |
| 2 | 主界面输入12进入函数 | 输出“5 4 3 2 1” | 输出“5 4 3 2 1” |
| 3 | 1.主界面输入10进入函数  2.按提示输入要插入的元素值，输入25  3. 按提示输入要插入的元素值，输入1 | 输出“插入成功！” | 输出“插入成功！” |
| 4 | 主界面输入12进入函数 | 输出“25 5 4 3 2 1” | 输出“25 5 4 3 2 1” |
| 5 | 1.主界面输入10进入函数  2.按提示输入要插入的元素值，输入7  3. 按提示输入要插入的元素值，输入7 | 输出“插入成功！” | 输出“插入成功！” |
| 6 | 主界面输入12进入函数 | 输出“25 5 4 3 2 1 7” | 输出“25 5 4 3 2 1 7” |

1. 测试函数：ListDelete

测试步骤及结果如表1-10所示

表1-10 ListDelete函数测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试步骤 | 测试输入 | 理论结果 | 运行结果 |
| 1 | 1.主界面输入13进入函数2.按提示输入要操作的线性表序号，输入2 | 输出“操作成功！当前线性表：表2” 按回车后当前线性表表数更新为2 | 输出“操作成功！当前线性表：表2” 按回车后当前线性表表数更新为2 |
| 2 | 主界面输入12进入函数 | 输出“5 4 3 2 1” | 输出“5 4 3 2 1” |
| 3 | 1.主界面输入11进入函数  2.按提示输入要删除的位序，输入2 | 输出“删除的元素值：4  删除成功！” | 输出“删除的元素值：4  删除成功！” |
| 4 | 主界面输入12进入函数 | 输出“5 3 2 1” | 输出“5 3 2 1” |
| 5 | 1.主界面输入11进入函数  2.按提示输入要删除的位序，输入1 | 输出“删除的元素值：5  删除成功！” | 输出“删除的元素值：5  删除成功！” |
| 6 | 主界面输入12进入函数 | 输出“3 2 1” | 输出“3 2 1” |
| 7 | 1.主界面输入11进入函数  2.按提示输入要删除的位序，输入3 | 输出“删除的元素值：1  删除成功！” | 输出“删除的元素值：1  删除成功！” |
| 8 | 主界面输入12进入函数 | 输出“3 2” | 输出“3 2” |

1. 测试函数：ListTraverse

测试步骤及结果如表1-11所示

表1-11 ListTraverse函数测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试步骤 | 测试输入 | 理论结果 | 运行结果 |
| 1 | 1.主界面输入13进入函数  2.按提示输入要操作的线性表序号，输入2 | 输出“操作成功！当前线性表：表2” 按回车后当前线性表表数更新为2 | 输出“操作成功！当前线性表：表2” 按回车后当前线性表表数更新为2 |
| 2 | 主界面输入12进入函数 | 输出“5 4 3 2 1” | 输出“5 4 3 2 1” |
| 3 | 1.主界面输入13进入函数  2.按提示输入要操作的线性表序号，输入5 | 输出“操作成功！当前线性表：表5” 按回车后当前线性表表数更新为5 | 输出“操作成功！当前线性表：表5” 按回车后当前线性表表数更新为5 |
| 4 | 主界面输入12进入函数 | 输出“5 6 2 45 3 25 4 85 69” | 输出“5 6 2 45 3 25 4 85 69” |

1. 测试函数：ChooseList

测试步骤及结果如表1-12所示

表1-12 ChooseList函数测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试步骤 | 测试输入 | 理论结果 | 运行结果 |
| 1 | 1.主界面输入13进入函数  2.按提示输入要操作的线性表序号，输入2 | 输出“操作成功！当前线性表：表2” 按回车后当前线性表表数更新为2 | 输出“操作成功！当前线性表：表2” 按回车后当前线性表表数更新为2 |
| 2 | 主界面输入12进入函数 | 输出“5 4 3 2 1” | 输出“5 4 3 2 1” |
| 3 | 1.主界面输入13进入函数  2.按提示输入要操作的线性表序号，输入5 | 输出“操作成功！当前线性表：表5” 按回车后当前线性表表数更新为5 | 输出“操作成功！当前线性表：表5” 按回车后当前线性表表数更新为5 |
| 4 | 主界面输入12进入函数 | 输出“5 6 2 45 3 25 4 85 69” | 输出“5 6 2 45 3 25 4 85 69” |

1. 测试函数：SaveDate

测试步骤及结果如表1-13所示

表1-13 SaveDate函数测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试步骤 | 测试输入 | 理论结果 | 运行结果 |
| 1 | 1.主界面输入13进入函数2.按提示输入要操作的线性表序号，输入2 | 输出“操作成功！当前线性表：表2” 按回车后当前线性表表数更新为2 | 输出“操作成功！当前线性表：表2” 按回车后当前线性表表数更新为2 |
| 2 | 主界面输入12进入函数 | 输出“5 4 3 2 1” | 输出“5 4 3 2 1” |
| 3 | 1.主界面输入11进入函数  2.按提示输入要删除的位序，输入2 | 输出“删除的元素值：4  删除成功！” | 输出“删除的元素值：4  删除成功！” |
| 4 | 主界面输入12进入函数 | 输出“5 3 2 1” | 输出“5 3 2 1” |
| 5 | 主界面输入14进入函数 | 输出“数据保存成功！” | 输出“数据保存成功！” |
| 6 | 1.主界面输入0进入函数  2.按提示输入N，数据已经保存，不需在此处保存数据 | 输出“保存数据?(Y/N)”  输出“欢迎下次再使用本系统！” | 输出“保存数据?(Y/N)”  输出“欢迎下次再使用本系统！” |
| 7 | 重新运行目标程序 |  |  |
| 8 | 1.主界面输入13进入函数2.按提示输入要操作的线性表序号，输入2 | 输出“操作成功！当前线性表：表2” 按回车后当前线性表表数更新为2 | 输出“操作成功！当前线性表：表2” 按回车后当前线性表表数更新为2 |
| 9 | 主界面输入12进入函数 | 输出“5 3 2 1” | 输出“5 3 2 1” |

1. 测试函数：Exit

测试步骤及结果如表1-14所示

表1-14 Exit函数测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试步骤 | 测试输入 | 理论结果 | 运行结果 |
| 1 | 1.主界面输入13进入函数2.按提示输入要操作的线性表序号，输入2 | 输出“操作成功！当前线性表：表2” 按回车后当前线性表表数更新为2 | 输出“操作成功！当前线性表：表2” 按回车后当前线性表表数更新为2 |
| 2 | 主界面输入12进入函数 | 输出“5 4 3 2 1” | 输出“5 4 3 2 1” |
| 3 | 1.主界面输入11进入函数  2.按提示输入要删除的位序，输入2 | 输出“删除的元素值：4  删除成功！” | 输出“删除的元素值：4  删除成功！” |
| 4 | 主界面输入12进入函数 | 输出“5 3 2 1” | 输出“5 3 2 1” |
| 5 | 1.主界面输入0进入函数  2.按提示输入N，不在此处保存数据 | 1.输出“保存数据?(Y/N)”  2.输出“欢迎下次再使用本系统！”  3.程序关闭 | 1.输出“保存数据?(Y/N)”  2.输出“欢迎下次再使用本系统！”  3.程序关闭 |
| 6 | 重新运行目标程序 |  |  |
| 7 | 1.主界面输入13进入函数  2.按提示输入要操作的线性表序号，输入2 | 输出“操作成功！当前线性表：表2” 按回车后当前线性表表数更新为2 | 输出“操作成功！当前线性表：表2” 按回车后当前线性表表数更新为2 |
| 8 | 主界面输入12进入函数 | 输出“5 4 3 2 1” | 输出“5 4 3 2 1” |
| 9 | 1.主界面输入13进入函数  2.按提示输入要操作的线性表序号，输入2 | 输出“操作成功！当前线性表：表2” 按回车后当前线性表表数更新为2 | 输出“操作成功！当前线性表：表2” 按回车后当前线性表表数更新为2 |
| 10 | 主界面输入12进入函数 | 输出“5 4 3 2 1” | 输出“5 4 3 2 1” |
| 11 | 1.主界面输入11进入函数  2.按提示输入要删除的位序，输入2 | 输出“删除的元素值：4  删除成功！” | 输出“删除的元素值：4  删除成功！” |
| 12 | 主界面输入12进入函数 | 输出“5 3 2 1” | 输出“5 3 2 1” |
| 13 | 1.主界面输入0进入函数  2.按提示输入Y，在此处保存数据 | 1.输出“保存数据?(Y/N)”  2.输出“欢迎下次再使用本系统！”  3.程序关闭 | 1.输出“保存数据?(Y/N)”  2.输出“欢迎下次再使用本系统！”  3.程序关闭 |
| 14 | 重新运行目标程序 |  |  |
| 15 | 1.主界面输入13进入函数2.按提示输入要操作的线性表序号，输入2 | 输出“操作成功！当前线性表：表2” 按回车后当前线性表表数更新为2 | 输出“操作成功！当前线性表：表2” 按回车后当前线性表表数更新为2 |
| 16 | 主界面输入12进入函数 | 输出“5 3 2 1” | 输出“5 3 2 1” |

**1.4 实验小结**

本次实验加深了对线性表的概念、基本运算的理解，掌握了线性表的基本预算的实现。熟练了线性表的逻辑结构和物理结构之间的关系。今后的学习过程中应当多从数据结构的角度分析如何进行数据的处理、存储以方便问题的解决，并要勤加练习达到熟能生巧的地步。

# 2基于链式存储结构的线性表实现

## 2.1 实验目的

通过实验达到⑴加深对线性表的概念、基本运算的理解；⑵熟练掌握线性表的逻辑结构与物理结构的关系；⑶物理结构采用单链表,熟练掌握线性表的基本运算的实现。

## 2.2 线性表基本运算定义

依据最小完备性和常用性相结合的原则，以函数形式定义了线性表的初始化表、销毁表、清空表、判定空表、求表长和获得元素等12种基本运算，具体运算功能定义如下。

⑴初始化表：函数名称是InitaList(L)；初始条件是线性表L不存在已存在；操作结果是构造一个空的线性表。

⑵销毁表：函数名称是DestroyList(L)；初始条件是线性表L已存在；操作结果是销毁线性表L。

⑶清空表：函数名称是ClearList(L)；初始条件是线性表L已存在；操作结果是将L重置为空表。

⑷判定空表：函数名称是ListEmpty(L)；初始条件是线性表L已存在；操作结果是若L为空表则返回TRUE,否则返回FALSE。

⑸求表长：函数名称是ListLength(L)；初始条件是线性表已存在；操作结果是返回L中数据元素的个数。

⑹获得元素：函数名称是GetElem(L,i,e)；初始条件是线性表已存在，1≤i≤ListLength(L)；操作结果是用e返回L中第i个数据元素的值。

⑺查找元素：函数名称是LocateElem(L,e,compare())；初始条件是线性表已存在；操作结果是返回L中第1个与e满足关系compare（）关系的数据元素的位序，若这样的数据元素不存在，则返回值为0。

⑻获得前驱：函数名称是PriorElem(L,cur\_e,pre\_e)；初始条件是线性表L已存在；操作结果是若cur\_e是L的数据元素，且不是第一个，则用pre\_e返回它的前驱，否则操作失败，pre\_e无定义。

⑼获得后继：函数名称是NextElem(L,cur\_e,next\_e)；初始条件是线性表L已存在；操作结果是若cur\_e是L的数据元素，且不是最后一个，则用next\_e返回它的后继，否则操作失败，next\_e无定义。

⑽插入元素：函数名称是ListInsert(L,i,e)；初始条件是线性表L已存在且非空，1≤i≤ListLength(L)+1；操作结果是在L的第i个位置之前插入新的数据元素e。

⑾删除元素：函数名称是ListDelete(L,i,e)；初始条件是线性表L已存在且非空，1≤i≤ListLength(L)；操作结果：删除L的第i个数据元素，用e返回其值。

⑿遍历表：函数名称是ListTraverse(L,visit())，初始条件是线性表L已存在；操作结果是依次对L的每个数据元素调用函数visit()。

**2.2 系统设计**

2.2.1 系统总体设计

本系统采用顺序表作为线性表的物理结构，实现线性表的基本运算。遵守C++14标准。

系统具有一个Terminal风格交互界面，称为rfaketerm，在general\_ui.hpp中实现。fake\_terminal::go会阻塞主线程，接收输入，简单parse之后通过callback函数进行处理。callback是一个更高级的parser，负责将输入翻译到C++函数地址并执行std::invoke，获取返回值，即时打印到stdout。在程序发生未定义行为时，会通过std::exception向自身发送*SIGABRT*信号，这有利于通用调试工具的应用。

在主程序中完成函数调用所需实参值的准备和函数执行结果的显示，并给出适当的操作提示显示。

系统定义一个reflection\_impl(作为本题要求的接口和容器库普遍承认的接口之间的wrapper)，其中含有一个核心链表对象Lab::list。如果需要实现对多个线性表的管理，只需使用std::deque<reflection\_impl>即可进行管理。

该演示系统提供的操作有：表的初始化、销毁、清空、判空，求表长、获取数据元素、查找数据元素、获得前驱、获得后继、插入数据元素、删除数据元素、表的遍历、表的选择。

在程序中实现消息处理和操作提示，包括数据的输入和输出，错误操作提示、程序的退出。

2.2.2 算法设计

（1）InitList(SqList \* L)

设计：分配存储空间，并初始化表长为0，表容量为LIST\_INIT\_SIZE。每次创建表时新建的表位序为最大。例如当前有6个表，执行此函数后创建的新表为表7

操作结果：构造一个空的线性表。

（2）DestroyList(SqList \* L)

设计：释放存储空间，每次操作当前线性表，销毁后当前线性表之后的线性表左移一个位序。例如当前操作表2，销毁表2后原表3左移成为表2，以此类推

操作结果：销毁线性表L。

（3）ClearList(SqList \* L)

设计：线性表L的长度赋值为0

操作结果：将L重置为空表。

（4）ListEmpty(SqList L)

设计：根据表长判断表是否为空

操作结果：若L为空表，则返回TRUE,否则返回FALSE。

（5）ListLength(SqList L)

设计：返回表长

操作结果：返回L中数据元素的个数。

（6）GetElem(SqList L, int i, ElemType \* e)

设计：根据位序找到第i个元素的地址并将其值赋值给指针e指向的元素

操作结果：用指针e指向的元素返回L中第i个数据元素的值。

（7）LocateElem(SqList L, ElemType e)

设计：遍历线性表找到第一个和元素e的相等的元素

操作结果：返回L中第1个与e相等的的数据元素的位序，若这样的数据元素不存在，则返回值为0。

（8）PriorElem(SqList L, ElemType cur, ElemType \* pre\_e)

设计：遍历线性表找到第一个和元素cur的相等的元素，如果其有前驱，用pre\_e返回，函数返回TRUE；否则函数返回FALSE，pre\_e无意义

操作结果：若cur是L的数据元素，且不是第一个，则用pre\_e返回它的 前驱，否则操作失败，pre\_e无定义。

（9）NextElem（L，cur\_e，&next\_e）

设计：遍历线性表找到第一个和元素cur的相等的元素，如果其有后继，用next\_e返回，函数返回TRUE；否则函数返回FALSE，next\_e无意义

操作结果：若cur是L的数据元素，且不是最后一个，则用next\_e返回它 的后继，否则操作失败，next\_e无定义。

（10）ListInsert(SqList \* L, int i, ElemType e)

设计：如果线性表已满，重新分配存储空间。将线性表指针L指向的线性表第i个元素之后的元素都右移一个位序，之后将e插入第i个位序

操作结果：在L的第i个位置之前插入新的数据元素e，L的长度加1

（11）ListDelete(SqList \* L, int i, ElemType \* e)

设计：将第i个位序的值赋给指针e指向的变量，之后第i个位序之后的元素全部左移一个位序

操作结果：删除L的第i个数据元素，用e返回其值，L的长度减1.

（12）ListTraverse(SqList L)

设计：遍历并输出表L中的每个元素值，返回表长

操作结果：依次输出表L中的每个变量的值

**2.3 顺序表演示系统实现与测试**

2.3.1 系统实现

编程环境：Linux x86\_64 ARCH gcc 8.0.0 cmake 3.10.0 GNU Make 4.2.1 GNU ld 2.29.1 GNU ar 2.29.1 kernel 4.13.12-1-ARCH 其他环境设定均在CMakeLists.txt进行了说明。

为Windows进行了交叉编译，使用cmake 3.10.0 mingw-gcc 6.3.1 nmake Windows 10 1709 (summer creator update) 静态编译使用mingw-gcc 6.3.1提供的libstdc++。

Windows版本缺失部分功能(界面美化)。

下面是src目录下的hpp/cc/CMakeLists.txt文件清单：依赖于rlib，此库被打包进源码目录，库内容均为原创。其中包含了测试所用代码。

//////////// FileName := CMakeLists.txt

cmake\_minimum\_required(VERSION 3.5)

project(hust\_shit)

set(CMAKE\_CXX\_STANDARD 14)

set(CMAKE\_C\_STANDARD 11)

set(CMAKE\_VERBOSE\_MAKEFILE ON)

set(CMAKE\_CXX\_FLAGS\_DEBUG "-g -DMALLOC\_CHECK\_=2")

set(CMAKE\_CXX\_FLAGS\_RELEASE "-O3")

set(CMAKE\_CXX\_FLAGS "${CMAKE\_CXX\_FLAGS} -msse4.2")

include\_directories("/usr/include")

include\_directories("/usr/local/include")

include\_directories(".")

set(BUILD\_SRC main.cc reflected\_impl.hpp lab\_list.hpp labafx.hpp general\_ui.hpp parser.hpp)

add\_executable(exp2 ${BUILD\_SRC})

//////////// FileName := general\_ui.hpp

#ifndef HUST\_SHIT\_GENERAL\_UI\_HPP\_

#define HUST\_SHIT\_GENERAL\_UI\_HPP\_

#include <functional>

#include <string>

#include <iostream>

#include <list>

#include <rlib/stdio.hpp>

#include <rlib/terminal.hpp>

#include <rlib/string/string.hpp>

#include <rlib/sys/os.hpp>

using namespace rlib::terminal;

using rlib::splitString;

class fake\_terminal {

public:

using callback\_t = std::function<void (std::list<std::string>)>;

[[noreturn]] static void go(const callback\_t &callback) {

while(true) {

prompt();

callback(splitString(rlib::io::scanln()));

}

}

private:

static void prompt() {

if constexpr(rlib::OSInfo::os == rlib::OSInfo::os\_t::WINDOWS) {

rlib::io::print(color\_t::green, "rfaketerm 0.0", clear, font\_t::bold, "~", clear);

}

else {

rlib::io::print(color\_t::green, "rfaketerm 0.0", clear, font\_t::bold, "~", clear);

}

}

};

#endif

//////////// FileName := labafx.hpp

#ifndef LAB\_AFX\_HPP\_

#define LAB\_AFX\_HPP\_

#include <cstddef>

#include <nmmintrin.h>

// typedef struct lab\_\_pair\_st Pair;

namespace LabUtils

{

template<typename ForwardIterator>

size\_t distance(ForwardIterator a, ForwardIterator b)

{

size\_t dist = 0;

for (; true; ++dist, ++a)

{

if (a == b) break;

}

return dist;

}

template<typename ForwardIterator>

ForwardIterator advance(ForwardIterator a, size\_t n)

{

for (size\_t cter = 0; cter < n; ++cter)

{

++a;

}

return a;

}

}

namespace Lab

{

////////////////////////////////////////////////

///////// SECTION TO IGNORE BEGINS /////////////

////////////////////////////////////////////////

constexpr unsigned INIT\_HASH\_VALUE = 0x01234567;

unsigned int naive\_hash(const void \*data, int size)

{

// work only for Plain Old Data (POD)

// stupid but efficient for random data

// unsafe for attack, but security is NOT required

auto crc = INIT\_HASH\_VALUE;

unsigned char \*data\_ = (unsigned char \*) data;

for (int i = 0; i < size; ++i)

{

crc = \_mm\_crc32\_u8(crc, data\_[i]);

}

return crc;

}

template<typename T>

unsigned int hast\_f(const T &s)

{

// optimised for base type

// faster than pure naive\_hash

return naive\_hash(&s, sizeof(s));

}

// the following is for speedups

template<>

unsigned int hast\_f(const unsigned long long &s)

{

return \_mm\_crc32\_u64(INIT\_HASH\_VALUE, s);

}

template<>

unsigned int hast\_f(const long long &s)

{

return \_mm\_crc32\_u64(INIT\_HASH\_VALUE, s);

}

template<>

unsigned int hast\_f(const double &s)

{

union

{

double f;

unsigned long long i;

} u;

u.f = s;

return \_mm\_crc32\_u64(INIT\_HASH\_VALUE, u.i);

}

template<>

unsigned int hast\_f(const float &s)

{

union

{

float f;

unsigned int i;

} u;

u.f = s;

return \_mm\_crc32\_u32(INIT\_HASH\_VALUE, u.i);

}

template<>

unsigned int hast\_f(const unsigned &s)

{

return \_mm\_crc32\_u32(INIT\_HASH\_VALUE, s);

}

template<>

unsigned int hast\_f(const int &s)

{

return \_mm\_crc32\_u32(INIT\_HASH\_VALUE, s);

}

template<>

unsigned int hast\_f(const unsigned short &s)

{

return \_mm\_crc32\_u16(INIT\_HASH\_VALUE, s);

}

template<>

unsigned int hast\_f(const short &s)

{

return \_mm\_crc32\_u16(INIT\_HASH\_VALUE, s);

}

template<>

unsigned int hast\_f(const signed char &s)

{

return \_mm\_crc32\_u8(INIT\_HASH\_VALUE, s);

}

template<>

unsigned int hast\_f(const unsigned char &s)

{

return \_mm\_crc32\_u8(INIT\_HASH\_VALUE, s);

}

template<>

unsigned int hast\_f(const char &s)

{

return \_mm\_crc32\_u8(INIT\_HASH\_VALUE, s);

}

////////////////////////////////////////////////

////////// SECTION TO IGNORE ENDS //////////////

////////////////////////////////////////////////

// work only for Plain Old Data (POD)

// otherwise correctness is not guaranteed

// stupid but efficient for random data

// unsafe for attack, since security is NOT required

unsigned int naive\_hash(const void \*data, int size);

// Lab::hash<T> simulates std::hash<T>

// usage: hash\_result = hash<T>()(item\_to\_hash);

template<typename T>

class hash

{

public:

unsigned int operator()(const T &s) { return hash\_f(s); }

};

template<typename T1, typename T2>

struct pair

{

T1 first;

T2 second;

};

// USE Lab::make\_pair LIKE std::make\_pair

template<typename T1, typename T2>

pair<T1, T1> make\_pair(const T1 &first, const T2 &second)

{

return pair<T1, T2>{first, second};

};

// usage:

// auto comp = less<T>();

// comp(a, b) == a < b;

// OR

// less<T>()(a, b) == a < b;

template<typename T>

class less

{

public:

bool operator()(const T &a, const T &b) { return a < b; }

};

}

#endif

//////////// FileName := lab\_list.hpp

#ifndef LAB\_LIST\_HPP\_

#define LAB\_LIST\_HPP\_

#include <cstddef>

#include <iterator>

namespace Lab

{

template<typename Type>

class list

{

private:

struct node

{

node() = default;

node(const Type &data, node \*pre, node \*next) : data(data), pre(pre), next(next) {}

Type data = 0;

node \*pre = nullptr;

node \*next = nullptr;

};

public:

class iterator :

public std::iterator<std::bidirectional\_iterator\_tag, Type, Type, const Type \*, Type &>

{

public:

iterator(node \*currentTmp) { current = currentTmp; }

Type &operator\*() { return current->data; }

const Type &operator\*() const { return current->data; }

iterator &operator++()

{

current = current->next;

return \*this;

}

iterator &operator--()

{

current = current->pre;

return \*this;

}

iterator &operator++(int)

{

auto restore = \*this;

current = current->next;

return restore;

}

iterator &operator--(int)

{

auto restore = \*this;

current = current->pre;

return restore;

}

bool operator!=(const iterator &another) const

{

return another.current != current;

}

bool operator==(const iterator &another) const

{

return another.current == current;

}

node \*current;

};

void push\_back(const Type &elem);

void push\_front(const Type &elem);

iterator begin();

iterator end();

size\_t size() const;

void pop\_front();

void pop\_back();

void insert(iterator iter, const Type &elem);

void erase(iterator iter);

void clear();

~list();

private:

node \*beg = nullptr;

node \*en = nullptr;

size\_t length = 0;

};

template<typename Type>

typename list<Type>::iterator list<Type>::begin()

{

return iterator(beg);

}

template<typename Type>

typename list<Type>::iterator list<Type>::end()

{

return iterator(en);

}

template<typename Type>

void list<Type>::push\_back(const Type &elem)

{

node \*newNode = new node;

newNode->data = elem;

newNode->pre = en;

newNode->next = nullptr;

if (en)

{

en->next = newNode;

}

en = newNode;

if (!length)

{

beg = newNode;

}

length++;

}

template<typename Type>

void list<Type>::push\_front(const Type &elem)

{

node \*newNode = new node;

newNode->data = elem;

newNode->next = beg;

newNode->pre = nullptr;

if (beg)

{

beg->pre = newNode;

}

beg = newNode;

if (!length)

en = newNode;

length++;

}

template<typename Type>

size\_t list<Type>::size() const { return length; }

template<typename Type>

void list<Type>::pop\_front()

{

node \*newNode = new node;

newNode = beg;

beg = beg->next;

if (beg)

beg->pre = nullptr;

length--;

delete newNode;

}

template<typename Type>

void list<Type>::pop\_back()

{

node \*newNode = new node;

length--;

newNode = en;

en = en->pre;

if (en)

en->next = nullptr;

delete newNode;

}

template<typename Type>

void list<Type>::insert(iterator iter, const Type &elem)

{

if(iter == this->end()) return this->push\_back(elem);

if(iter == this->begin()) return this->push\_front(elem);

node \*newNode = new node{elem, iter.current->pre, iter.current};

iter.current->pre->next = newNode;

iter.current->pre = newNode;

length++;

}

template<typename Type>

void list<Type>::erase(iterator iter)

{

// node \*newNode = iter.current->pre;

if (iter.current->pre)

iter.current->pre->next = iter.current->next;

if (iter.current->next)

iter.current->next->pre = iter.current->pre;

delete iter.current;

length--;

}

template<typename Type>

void list<Type>::clear()

{

while (beg != en)

{

node \*newNode = beg->next;

delete beg;

beg = newNode;

length--;

}

delete beg;

beg = en = nullptr;

length = 0;

}

template<typename Type>

list<Type>::~list()

{

this->clear();

}

} // namespace Lab

#endif

//////////// FileName := list\_test.cc

/\*\*

\* By recolic, Nov 10.

\*/

#include <chrono>

#include <iostream>

#include <random>

#include <functional>

#include "test\_utils.hpp"

using println = rlib::io::println;

std::default\_random\_engine rand\_eng(810);

std::uniform\_real\_distribution<double> distribution(0, 100);

double m\_rand() {return distribution(rand\_eng);}

template <class operation\_t, typename... args\_t>

void timed\_func(const std::string &info, std::function<operation\_t> f, args\_t... args)

{

println(info, "launched.");

auto begin = std::chrono::high\_resolution\_clock::now();

f(args ...);

auto end = std::chrono::high\_resolution\_clock::now();

println(info, "used", std::chrono::duration<double>(end - begin).count(), "s");

}

template <class operation\_t, typename... args\_t>

void repeat(size\_t count, std::function<operation\_t> f, args\_t... args)

{

for(size\_t cter = 0; cter < count; ++cter)

f(args ...);

}

int main()

{

using data\_t = double;

Lab::list<data\_t> lsa;

std::list<data\_t> lsb;

using op\_arg1\_t = Lab::list<data\_t> &;

using op\_arg2\_t = std::list<data\_t> &;

#define op\_args\_t op\_arg1\_t, op\_arg2\_t

using operation\_t = void(op\_args\_t);

auto co\_push\_back = [](auto &bufa, auto &bufb){

auto val = m\_rand();

bufa.push\_back(val);

bufb.push\_back(val);

};

auto co\_push\_front = [](auto &bufa, auto &bufb){

auto val = m\_rand();

bufa.push\_front(val);

bufb.push\_front(val);

};

auto co\_pop\_front = [](auto &bufa, auto &bufb){

bufa.pop\_front();

bufb.pop\_front();

};

auto co\_pop\_back = [](auto &bufa, auto &bufb){

bufa.pop\_back();

bufb.pop\_back();

};

auto co\_erase = [](auto &bufa, auto &bufb){

bufa.erase(++bufa.begin());

bufb.erase(++bufb.begin());

};

auto co\_clear = [](auto &bufa, auto &bufb){

bufa.clear();

bufb.clear();

};

using namespace std::placeholders;

#define TEST(count, operation, desc) LIST\_ASSERT\_EQUIVALENCE(lsa, lsb, std::function<operation\_t>( \

std::bind(timed\_func<operation\_t, op\_args\_t>, desc, \

std::function<operation\_t>(std::bind(repeat<operation\_t, op\_args\_t>, count, operation, \_1, \_2)), \

\_1, \_2)))

TEST(1000, co\_push\_back, "push1");

TEST(10000000, co\_push\_back, "push2");

TEST(9999000, co\_pop\_back, "pop1");

TEST(54320, co\_push\_back, "push3");

TEST(123, co\_pop\_back, "pop2");

TEST(1, co\_erase, "erase1");

TEST(66, co\_push\_back, "push4");

TEST(543, co\_erase, "erase2");

TEST(2, co\_clear, "clear1");

TEST(3456, co\_push\_back, "push5");

println("s/back/front/g and retest...");

TEST(1000, co\_push\_front, "push1");

TEST(10000000, co\_push\_front, "push2");

TEST(9999000, co\_pop\_front, "pop1");

TEST(54320, co\_push\_front, "push3");

TEST(123, co\_pop\_front, "pop2");

TEST(1, co\_erase, "erase1");

TEST(66, co\_push\_front, "push4");

TEST(543, co\_erase, "erase2");

TEST(2, co\_clear, "clear1");

TEST(3456, co\_push\_front, "push5");

println("All tests done.");

return 0;

}

//////////// FileName := main.cc

#include <general\_ui.hpp>

#include <parser.hpp>

reflected\_impl impl;

int main() {

fake\_terminal::go(parser::parse);

}

//////////// FileName := parser.hpp

#ifndef \_HUST\_SHIT\_PARSER\_HPP

#define \_HUST\_SHIT\_PARSER\_HPP 1

#include <reflected\_impl.hpp>

#include <list>

#include <string>

#include <iomanip>

#include <rlib/stdio.hpp>

#include <rlib/terminal.hpp>

using namespace rlib::terminal;

class parser {

private:

static std::string getArg(const std::list<std::string> &ls, size\_t n) {

auto iter = ls.cbegin();

for(size\_t cter = 0; cter < n; ++cter) {

++iter;

}

return std::move(\*iter);

}

static void help\_msg() {

std::string msg = R"\_STR\_(

rfaketerm 0.0 shit specially edition

Usage: <Command> [args ...]

Command List:

help : Show this message.

exit : exit politely.

InitList

DestroyList

ClearList

ListEmpty

ListLength

GetElem <size\_t positionPlusOne>

LocateElem <data\_t elemValue>

PriorElem <data\_t elemValue>

NextElem <data\_t elemValue>

ListInsert <size\_t positionPlusOne> <data\_t elemValue>

ListDelete <size\_t positionPlusOne>

ListTraverse

)\_STR\_";

rlib::io::println(msg);

}

public:

static void parse(const std::list<std::string> &to\_parse) {

if(to\_parse.empty())

return;

rlib::io::print(std::boolalpha);

#define IFCMD(str) if(\*to\_parse.begin() == str)

#define WANT\_ARG(n) if(to\_parse.size() != n+1) {rlib::io::println(color\_t::red, font\_t::bold, "Error:", clear, color\_t::lightgray, n, "arguments wanted but", to\_parse.size()-1, "provided.", clear); return;}

#define SIZE\_ARG(n) std::stoul(getArg(to\_parse, n))

#define DATA\_ARG(n) std::stoi(getArg(to\_parse, n))

#define HAVE\_RETURN\_VALUE auto ret =

#define PRINT\_RETURN\_VALUE rlib::io::println(ret);

IFCMD("InitList") {

WANT\_ARG(0)

impl.InitList();

}

IFCMD("DestroyList") {

WANT\_ARG(0)

impl.DestroyList();

}

IFCMD("ClearList") {

WANT\_ARG(0)

impl.ClearList();

}

IFCMD("ListEmpty") {

WANT\_ARG(0)

HAVE\_RETURN\_VALUE

impl.ListEmpty();

PRINT\_RETURN\_VALUE

}

IFCMD("ListLength") {

WANT\_ARG(0)

HAVE\_RETURN\_VALUE

impl.ListLength();

PRINT\_RETURN\_VALUE

}

IFCMD("GetElem") {

WANT\_ARG(1)

HAVE\_RETURN\_VALUE

impl.GetElem(SIZE\_ARG(1));

PRINT\_RETURN\_VALUE

}

IFCMD("LocateElem") {

WANT\_ARG(1)

HAVE\_RETURN\_VALUE

impl.LocateElem(DATA\_ARG(1));

PRINT\_RETURN\_VALUE

}

IFCMD("PriorElem") {

WANT\_ARG(1)

HAVE\_RETURN\_VALUE

impl.PriorElem(DATA\_ARG(1));

PRINT\_RETURN\_VALUE

}

IFCMD("NextElem") {

WANT\_ARG(1)

HAVE\_RETURN\_VALUE

impl.NextElem(DATA\_ARG(1));

PRINT\_RETURN\_VALUE

}

IFCMD("ListInsert") {

WANT\_ARG(2)

impl.ListInsert(SIZE\_ARG(1), DATA\_ARG(2));

}

IFCMD("ListDelete") {

WANT\_ARG(1)

HAVE\_RETURN\_VALUE

impl.ListDelete(SIZE\_ARG(1));

PRINT\_RETURN\_VALUE

}

IFCMD("ListTraverse") {

WANT\_ARG(0)

impl.ListTraverse();

}

// Shits done.

IFCMD("exit") {

rlib::io::println("bye~");

::std::exit(0);

}

IFCMD("help") {

help\_msg();

}

//impl.debug();

}

};

#endif //\_HUST\_SHIT\_PARSER\_HPP

//////////// FileName := reflected\_impl.hpp

#ifndef HUST\_SHIT\_REFLECTED\_IMPL\_HPP\_

#define HUST\_SHIT\_REFLECTED\_IMPL\_HPP\_

/\*

\* You should NEVER use this code in ANY consequence,

\* as these code is just to make hust happy.

\*/

#include <utility>

#include <functional>

#include <algorithm>

#include "lab\_list.hpp"

#include "labafx.hpp"

#include <rlib/stdio.hpp>

class reflected\_impl {

public:

using data\_t = int;

using BooleanAsserter = std::function<bool(const data\_t &)>;

using OperationVisiter = std::function<void(const data\_t &)>;

void InitList() const {}

void DestroyList() {container.clear();}

void ClearList() {container.clear();}

bool ListEmpty() const {return container.size() == 0;}

size\_t ListLength() const {return container.size();}

data\_t GetElem(size\_t \_shit\_IndexPlusOne) {

auto index = \_shit\_IndexPlusOne - 1;

auto iter = container.begin();

for(size\_t cter = 0; cter < index; ++cter) {

++iter;

}

return std::move(\*iter);

}

size\_t \_LocateElem(const BooleanAsserter &comparer) {

auto iter = std::find\_if(container.begin(), container.end(), comparer);

if(iter == container.end()) {

return 0;

}

return LabUtils::distance(container.begin(), iter);

}

size\_t LocateElem(data\_t val) {

auto comparer = BooleanAsserter([v=val](const data\_t &dat){

return dat == v;

});

return \_LocateElem(comparer);

}

data\_t PriorElem(data\_t tofind) {

auto pos = std::find(container.begin(), container.end(), tofind);

if(pos == container.end() || pos == container.begin()) {

throw std::runtime\_error("ElemError: You told me that it's undefined, so I do it.");

}

return \*(--pos);

}

data\_t NextElem(data\_t tofind) {

auto pos = std::find(container.begin(), container.end(), tofind);

if(pos == container.end() || pos == --container.end()) {

throw std::runtime\_error("ElemError: You told me that it's undefined, so I do it.");

}

return \*(++pos);

}

void ListInsert(size\_t \_shit\_IndexPlusOne, data\_t elem) {

auto index = \_shit\_IndexPlusOne - 1;

auto iter = LabUtils::advance(container.begin(), index);

container.insert(iter, elem);

}

data\_t ListDelete(size\_t \_shit\_IndexPlusOne) {

auto index = \_shit\_IndexPlusOne - 1;

auto iter = LabUtils::advance(container.begin(), index);

auto to\_return = \*iter;

container.erase(iter);

return std::move(to\_return);

}

void \_ListTraverse(const OperationVisiter &visiter) {

std::for\_each(container.begin(), container.end(), visiter);

}

void ListTraverse() {

\_ListTraverse(OperationVisiter([](const auto &val){rlib::io::print(val, " ");}));

rlib::io::println("");

}

void debug() {

rlib::io::println\_iter(container);

rlib::io::println(container.size());

}

private:

Lab::list<data\_t> container;

};

extern reflected\_impl impl;

#endif

//////////// FileName := rlib

cat: rlib: 是一个目录

//////////// FileName := test\_utils.hpp

#include <cstdlib>

#include <rlib/stdio.hpp>

#include <rlib/traits.hpp>

#define dynamic\_assert(cond, message) do { \

if(!cond) { \

rlib::io::println("dynamic assertion failed:", message); \

std::exit(2); \

} \

} while(false)

// -- operation must be a templated callable object, usually templated lambda.

// NEW: operation must fuck two buf at same time.

#define ASSERT\_EQUIVALENCE(bufA, bufB, operation, equal\_checker) \

do { \

static\_assert(std::is\_same<rlib::is\_callable<decltype(equal\_checker<double>)>::type, \

std::true\_type>::value, \

"equal\_checker is not callable"); \

dynamic\_assert(equal\_checker(bufA, bufB), "given buf is not equal."); \

operation(bufA, bufB); \

dynamic\_assert(equal\_checker(bufA, bufB), "operation failed."); \

} while(false)

/\*

//vector

#include "lab\_vector.hpp"

#include <vector>

template<typename data\_t>

bool vector\_equal(const Lab::vector<data\_t> &vcta, const std::vector<data\_t> &vctb)

{

if(vcta.size() != vctb.size()) return false;

Lab::vector<data\_t> &fake\_vcta = const\_cast<Lab::vector<data\_t> &>(vcta);

for(auto ia = fake\_vcta.begin(), ib = vctb.begin();

ia != fake\_vcta.end() && ib != vctb.end();

++ia, ++ib)

{

if(\*ia != \*ib) return false;

}

return true;

}

#define VECTOR\_ASSERT\_EQUIVALENCE(bufA, bufB, operation) ASSERT\_EQUIVALENCE(bufA, bufB, operation, vector\_equal)

\*/

//list

#include "lab\_list.hpp"

#include <list>

template<typename data\_t>

bool list\_equal(const Lab::list<data\_t> &bufa, const std::list<data\_t> &bufb)

{

if(bufa.size() != bufb.size()) return false;

Lab::list<data\_t> &fake\_bufa = const\_cast<Lab::list<data\_t> &>(bufa);

for(auto ia = fake\_bufa.begin(), ib = bufb.begin();

ia != fake\_bufa.end() && ib != bufb.end();

++ia, ++ib)

{

if(\*ia != \*ib) return false;

}

return true;

}

#define LIST\_ASSERT\_EQUIVALENCE(bufA, bufB, operation) ASSERT\_EQUIVALENCE(bufA, bufB, operation, list\_equal)

/\*

//set

#include "lab\_set.hpp"

#include <set>

template<typename data\_t>

bool set\_equal(const Lab::set<data\_t> &bufa, const std::set<data\_t> &bufb)

{

if(bufa.size() != bufb.size()) return false;

Lab::set<data\_t> &fake\_bufa = const\_cast<Lab::set<data\_t> &>(bufa);

for(auto ia = fake\_bufa.begin(), ib = bufb.begin();

ia != fake\_bufa.end() && ib != bufb.end();

++ia, ++ib)

{

if(\*ia != \*ib) return false;

}

return true;

}

#define SET\_ASSERT\_EQUIVALENCE(bufA, bufB, operation) ASSERT\_EQUIVALENCE(bufA, bufB, operation, set\_equal)

//priority\_queue

#include "lab\_priority\_queue.hpp"

#include <queue>

template<typename data\_t>

bool priority\_queue\_equal(const Lab::priority\_queue<data\_t> &bufa, const std::priority\_queue<data\_t> &bufb)

{

return true;

}

#define PRIORITY\_QUEUE\_ASSERT\_EQUIVALENCE(bufA, bufB, operation) ASSERT\_EQUIVALENCE(bufA, bufB, operation, priority\_queue\_equal)

template<typename data\_t>

bool priority\_queue\_destroy\_and\_check(Lab::priority\_queue<data\_t> &bufa, std::priority\_queue<data\_t> &bufb)

{

if(bufa.size() != bufb.size()) return false;

while(bufb.size())

{

if(bufa.top() != bufb.top()) return false;

bufa.pop();

bufb.pop();

}

return true;

}

//unordered\_map

#include "lab\_unordered\_map.hpp"

#include <unordered\_map>

template<typename key\_t, typename data\_t>

bool unordered\_map\_equal(const Lab::unordered\_map<key\_t, data\_t> &bufa, const std::unordered\_map<key\_t, data\_t> &bufb)

{

if(bufa.size() != bufb.size()) return false;

Lab::unordered\_map<key\_t, data\_t> &fake\_bufa = const\_cast<Lab::unordered\_map<key\_t, data\_t> &>(bufa);

for(auto ia = fake\_bufa.begin(), ib = bufb.begin();

ia != fake\_bufa.end() && ib != bufb.end();

++ia, ++ib)

{

if(\*ia != \*ib) return false;

if(fake\_bufa.find((\*ib).first) != ia) return false;

}

return true;

}

template<typename key\_data\_t>

bool \_unordered\_map\_equal(const Lab::unordered\_map<key\_data\_t, key\_data\_t> &bufa, const std::unordered\_map<key\_data\_t, key\_data\_t> &bufb)

{

return unordered\_map\_equal(bufa, bufb);

}

#define UNORDERED\_MAP\_ASSERT\_EQUIVALENCE(bufA, bufB, operation) ASSERT\_EQUIVALENCE(bufA, bufB, operation, \_unordered\_map\_equal)

\*/

2.3.2 算法测试

直接通过测试程序对算法部分可靠性进行测试。

插入/删除测试各20000000次，其他测试分必要性共几千次，测试结果完全正确(和std::list进行严格的表现比较)。

push1 launched.

push1 used 0.000312113 s

push2 launched.

push2 used 2.07433 s

pop1 launched.

pop1 used 0.89098 s

push3 launched.

push3 used 0.00973691 s

pop2 launched.

pop2 used 2.5136e-05 s

erase1 launched.

erase1 used 3.297e-06 s

push4 launched.

push4 used 4.5379e-05 s

erase2 launched.

erase2 used 0.00011576 s

clear1 launched.

clear1 used 0.00816342 s

push5 launched.

push5 used 0.0015594 s

s/back/front/g and retest...

push1 launched.

push1 used 0.000389672 s

push2 launched.

push2 used 1.93912 s

pop1 launched.

pop1 used 0.983706 s

push3 launched.

push3 used 0.01539 s

pop2 launched.

pop2 used 7.5143e-05 s

erase1 launched.

erase1 used 3.21e-06 s

push4 launched.

push4 used 0.000109478 s

erase2 launched.

erase2 used 0.000324628 s

clear1 launched.

clear1 used 0.0244477 s

push5 launched.

push5 used 0.000904992 s

All tests done.

2.3.3 界面测试

简单的测试表明，界面的正确性没有问题。

**2.4 实验小结**

本次实验加深了对线性表的概念、基本运算的理解，掌握了线性表的基本预算的实现。熟练了线性表的逻辑结构和物理结构之间的关系。今后的学习过程中应当多从数据结构的角度分析如何进行数据的处理、存储以方便问题的解决，并要勤加练习达到熟能生巧的地步。

# 3基于二叉链表的二叉树实现

**3.1 实验目的**

通过实验达到⑴加深对二叉树的概念、基本运算的理解；⑵熟练掌握二叉树的逻辑结构与物理结构的关系；⑶以二叉链表作为物理结构，熟练掌握二叉树基本运算的实现。

**3.2 系统设计**

3.2.1 系统总体设计

本系统采用顺序表作为线性表的物理结构，实现线性表的基本运算。遵守C++14标准。

系统具有一个Terminal风格交互界面，称为rfaketerm，在general\_ui.hpp中实现。fake\_terminal::go会阻塞主线程，接收输入，简单parse之后通过callback函数进行处理。callback是一个由ccgen.py生成代码的parser(即reflection,C++20标准库提供了原生功能)，负责将输入翻译到下一层即relected\_impl。它将请求进一步解释，并与后端数据结构进行交互，获取返回值，被rfaketerm打印到stdout。在程序发生未定义行为时，会通过std::exception向自身发送*SIGABRT*信号，这有利于通用调试工具的应用。为了美观，rfaketerm默认情况下会把所有异常抓下并打印错误信息到stdout。

User Manual在 rfaketerm中使用help命令即可获得。为了便于GUI下的使用，rfaketerm启动时会自动模拟执行help命令。

系统定义一个reflection\_impl(作为本题要求的接口和容器库普遍承认的接口之间的wrapper)，其负责管理数据结构对象hust\_xxxx::unordered\_btree。为了实现对多个线性表的管理，只需使用std::vector<btree>即可。

该演示系统提供的操作有：初始化二叉树、销毁二叉树、创建二叉树、清空二叉树、判定空二叉树和求二叉树深度等20种基本运算和Select, List等用于在多个树间切换的操作，详见help。

在程序中实现消息处理和操作提示，包括数据的输入和输出，错误操作提示、程序的退出。

3.2.2 算法设计

依据最小完备性和常用性相结合的原则，以函数形式定义了二叉树的初始化二叉树、销毁二叉树、创建二叉树、清空二叉树、判定空二叉树和求二叉树深度等20种基本运算，具体运算功能定义如下。

⑴初始化二叉树：函数名称是InitBiTree(T)；初始条件是二叉树T不存在；操作结果是构造空二叉树T。

⑵销毁二叉树：树函数名称是DestroyBiTree(T)；初始条件是二叉树T已存在；操作结果是销毁二叉树T。

⑶创建二叉树：函数名称是CreateBiTree(T,definition)；初始条件是definition 给出二叉树T的定义；操作结果是按definition构造二叉树T。

⑷清空二叉树：函数名称是ClearBiTree (T)；初始条件是二叉树T存在； 操作结果是将二叉树T清空。

⑸判定空二叉树：函数名称是BiTreeEmpty(T)；初始条件是二叉树T存在；操作结果是若T为空二叉树则返回TRUE，否则返回FALSE。

⑹求二叉树深度：函数名称是BiTreeDepth(T)；初始条件是二叉树T存在；操作结果是返回T的深度。

⑺获得根结点：函数名称是Root(T)；初始条件是二叉树T已存在；操作结果是返回T的根。

⑻获得结点：函数名称是Value(T,e)；初始条件是二叉树T已存在，e是T中的某个结点；操作结果是返回e的值。

⑼结点赋值：函数名称是Assign(T,&e,value)；初始条件是二叉树T已存在，e是T中的某个结点；操作结果是结点e赋值为value。

⑽获得双亲结点：函数名称是Parent(T,e) ；初始条件是二叉树T已存在，e是T中的某个结点；操作结果是若e是T的非根结点，则返回它的双亲结点指针，否则返回NULL。

⑾获得左孩子结点：函数名称是LeftChild(T,e)；初始条件是二叉树T存在，e是T中某个节点；操作结果是返回e的左孩子结点指针。若e无左孩子，则返回NULL。

⑿获得右孩子结点：函数名称是RightChild(T,e)；初始条件是二叉树T已存在，e是T中某个结点；操作结果是返回e的右孩子结点指针。若e无右孩子，则返回NULL。

⒀获得左兄弟结点：函数名称是LeftSibling(T,e)；初始条件是二叉树T存在，e是T中某个结点；操作结果是返回e的左兄弟结点指针。若e是T的左孩子或者无左兄弟，则返回NULL。

⒁获得右兄弟结点：函数名称是RightSibling(T,e)；初始条件是二叉树T已存在，e是T中某个结点；操作结果是返回e的右兄弟结点指针。若e是T的右孩子或者无有兄弟，则返回NULL。

⒂插入子树：函数名称是InsertChild(T,p,LR,c)；初始条件是二叉树T存在，p指向T中的某个结点，LR为0或1，,非空二叉树c与T不相交且右子树为空；操作结果是根据LR为0或者1，插入c为T中p所指结点的左或右子树，p 所指结点的原有左子树或右子树则为c的右子树

⒃删除子树：函数名称是DeleteChild(T.p.LR)；初始条件是二叉树T存在，p指向T中的某个结点，LR为0或1。 操作结果是根据LR为0或者1，删除c为T中p所指结点的左或右子树。

⒄前序遍历：函数名称是PreOrderTraverse(T,Visit())；初始条件是二叉树T存在，Visit是对结点操作的应用函数；操作结果：先序遍历t，对每个结点调用函数Visit一次且一次，一旦调用失败，则操作失败。

⒅中序遍历：函数名称是InOrderTraverse(T,Visit))；初始条件是二叉树T存在，Visit是对结点操作的应用函数；操作结果是中序遍历t，对每个结点调用函数Visit一次且一次，一旦调用失败，则操作失败。

⒆后序遍历：函数名称是PostOrderTraverse(T,Visit))；初始条件是二叉树T存在，Visit是对结点操作的应用函数；操作结果是后序遍历t，对每个结点调用函数Visit一次且一次，一旦调用失败，则操作失败。

⒇按层遍历：函数名称是LevelOrderTraverse(T,Visit))；初始条件是二叉树T存在，Visit是对结点操作的应用函数；操作结果是层序遍历t，对每个结点调用函数Visit一次且一次，一旦调用失败，则操作失败。

**3.3 二叉树演示系统实现与测试**

3.3.1 系统实现

编程环境：Linux x86\_64 ARCH gcc 8.0.0 cmake 3.10.1 GNU Make 4.2.1 GNU ld 2.29.1 GNU ar 2.29.1 kernel 4.14.5-1-ARCH 其他环境设定均在CMakeLists.txt进行了说明。

为Windows进行了交叉编译，使用cmake 3.10.0 mingw-gcc 6.3.1 nmake Windows 10 1709 (summer creator update) 静态编译使用mingw-gcc 6.3.1提供的libstdc++。Windows版本缺失部分功能(界面美化)。

使用了gc库。

下面是src目录下的hpp/cc/CMakeLists.txt文件清单：依赖于rlib，此库被打包进源码目录，库内容均为原创。其中包含了测试所用代码。

//FileName := btree.hpp

#ifndef HUST\_BTREE\_HPP\_

#define HUST\_BTREE\_HPP\_

//#include <gc.h> //You cannot compile it and it doesn't matter.

#include <rlib/require/cxx11>

#include <stdexcept>

#include <exception>

#include <functional>

#include <rlib/string/string.hpp>

#include <rlib/stdio.hpp>

namespace hust\_xxxx {

enum class foreach\_rule {LEFT\_MIDDLE\_RIGHT, LEFT\_RIGHT\_MIDDLE, MIDDLE\_LEFT\_RIGHT};

template<typename data\_t>

class [[deprecated/\*, "fatal memory bug, invalid algo, extremely bad design."\*/]] unordered\_btree {

struct node {

data\_t payload;

node \*left = nullptr;

node \*right = nullptr;

node \*parent = nullptr;

size\_t depth = 0; //Root

node() = delete;

node(const data\_t &payload, node \*parent) : payload(payload), parent(parent), depth(parent?parent->depth+1:0) {}

void for\_each(foreach\_rule rule, std::function<void(node &)> func) {

if(rule == foreach\_rule::MIDDLE\_LEFT\_RIGHT) func(\*this);

if(left) left->for\_each(rule, func);

if(rule == foreach\_rule::LEFT\_MIDDLE\_RIGHT) func(\*this);

if(right) right->for\_each(rule, func);

if(rule == foreach\_rule::LEFT\_RIGHT\_MIDDLE) func(\*this);

}

};

public:

using nlang = std::string;

unordered\_btree() {}

bool empty() const {

return root == nullptr;

}

bool clear() {

root = nullptr;

}

size\_t depth() {

size\_t max\_depth = 0;

this->for\_each([&max\_depth](node &n){

max\_depth = n.depth>max\_depth ? n.depth : max\_depth;

});

return max\_depth;

}

nlang \_root() {

return std::move(nlang(""));

}

data\_t get(const nlang &pos) {

auto n = nlang\_translate(pos);

if(!n)

throw std::runtime\_error("Trying to access an empty node.");

return std::move(n->payload);

}

void set(const nlang &pos, const data\_t &payload) {

auto iter = nlang\_translate(pos);

if(iter)

iter->payload = payload;

else

nlang\_translate(pos, true, payload);

}

nlang parent(nlang pos) {

rlib::replaceSubString(pos, " ", "");

return pos.empty() ? pos : pos.substr(0, pos.size()-1);

}

nlang lchild(const nlang &pos) {

return pos + 'L';

}

nlang rchild(const nlang &pos) {

return pos + 'R';

}

void for\_each(std::function<void(node &)> func, typename hust\_xxxx::foreach\_rule rule = foreach\_rule::LEFT\_MIDDLE\_RIGHT) {

if(root) root->for\_each(rule, func);

}

void level\_for\_each(std::function<void(node &)> func) {

size\_t curr\_depth = 0;

while(true) {

bool must\_break = true;

this->for\_each([&, \_curr\_depth=curr\_depth](node &n){

if(n.depth == \_curr\_depth) {

func(n);

must\_break = false;

}

});

if(must\_break) break;

++curr\_depth;

}

}

static void printer(node &n) {rlib::print(n.payload, "");}

void merge(unordered\_btree &another, const nlang &where, bool right) {

auto n = nlang\_translate(where);

if(right) n->right = another.root;

else n->left = another.root;

another.root = nullptr;

}

void drop(const nlang &where, bool right) {

auto n = nlang\_translate(where);

if(right) n->right = nullptr;

else n->left = nullptr;

}

private:

node \*nlang\_translate(const nlang &lang, bool newIfNull = false, const data\_t &newPayload = data\_t()) {

node \*curr = root;

for(auto act : lang) {

if(!curr)

throw std::runtime\_error("invalid nlang to this tree. Too many null in path.");

switch(act) {

case 'L':

if(!curr->left)

curr->left = new node(newPayload, curr);

curr = curr->left;

break;

case 'R':

if(!curr->right)

curr->right = new node(newPayload, curr);

curr = curr->right;

break;

case ' ':

break;

default:

throw std::runtime\_error("invalid nlang to this tree.");

}

}

if(!curr && newIfNull) //Create root.

root = new node(newPayload, nullptr);

return curr;

}

node \*root = nullptr;

};

}

#endif//FileName := ccgen.py

#!/usr/bin/python3

src = 'reflected\_impl.hpp'

mode = 'gen\_code'

#mode = 'gen\_help'

# DO NOT use macro in func\_name! It'll gen wrong code!

macro\_list = [

('nlangref','nlang'),

('nlang','NodeLanguage'),

('dataref\_t','data\_t'),

('void','null'),

]

size\_arg = ['size\_t']

int\_arg = ['int', 'data\_t']

string\_arg = ['NodeLanguage']

void\_ret = ['void', 'null']

def gen\_code(line):

line = line.replace('\t','').replace('\r', '').strip()

if len(line) == 0:

return

ret\_type = line.split(' ')[0]

funcAndArgs = line[len(ret\_type):].strip().split('(')

func\_name, args = funcAndArgs[0], funcAndArgs[1].split(')')[0]

print('//\_\_ccgen\_debug\_\_: `ret name(args)` is `{} {}({})`'.format(ret\_type, func\_name, args))

args\_string = []

for arg in args.split(','):

arg\_type = arg.strip().split(' ')[0].replace(' ','')

if len(arg\_type) == 0:

continue

if arg\_type in size\_arg:

args\_string.append('SIZE\_ARG({})'.format(len(args\_string)+1)) # start from one

elif arg\_type in int\_arg:

args\_string.append('INT\_ARG({})'.format(len(args\_string)+1)) # start from one

elif arg\_type in string\_arg:

args\_string.append('STRING\_ARG({})'.format(len(args\_string)+1)) # start from one

else:

raise RuntimeError('Unclassed arg left here. line={}|arg\_type={}'.format(line, arg\_type))

args\_size = len(args\_string)

args\_string = ', '.join(args\_string)

print(' IFCMD("{}") {{'.format(func\_name))

print(' WANT\_ARG({})'.format(args\_size))

if ret\_type not in void\_ret:

print(' HAVE\_RETURN\_VALUE')

print(' impl.{}({});'.format(func\_name, args\_string))

if ret\_type not in void\_ret:

print(' PRINT\_RETURN\_VALUE')

print(' }')

def gen\_help(line):

line = line.replace('\t','').replace('\r', '').strip()

if len(line) == 0:

return

ret\_type = line.split(' ')[0]

funcAndArgs = line[len(ret\_type):].strip().split('(')

func\_name, args = funcAndArgs[0], funcAndArgs[1].split(')')[0]

# print('//\_\_ccgen\_debug\_\_: `ret name(args)` is `{} {}({})`'.format(ret\_type, func\_name, args))

if len(args) == 0:

print('{} -> {}'.format(func\_name, ret\_type))

else:

print('{} [{}] -> {}'.format(func\_name, args, ret\_type))

if mode == 'gen\_code':

fuck\_a\_line = gen\_code

print('//Code generated by ccgen.py below. Do not edit them by hand.')

else:

fuck\_a\_line = gen\_help

print('FuncName [Argument ...] -> ReturnValue # Instructions')

with open(src) as fd:

cont = fd.read()

working = False

for line in cont.split('\n'):

if -1 != line.find('\_\_py\_ccgen\_begin\_\_'):

working = True

continue

if -1 != line.find('\_\_py\_ccgen\_end\_\_'):

working = False

continue

if working:

for \_from, \_to in macro\_list:

line = line.replace(\_from, \_to)

fuck\_a\_line(line)

if mode != 'gen\_code':

exit(0)

print('''

IFCMD("exit") {

rlib::println("bye~");

::std::exit(0);

}

IFCMD("help") {

help\_msg();

}

//impl.debug();

//Code generated by ccgen.py ahead. Do not edit them by hand.

''')//FileName := cmake\_clean.sh

#!/bin/bash

make clean

rm -rf cmake-build-debug/ cmake\_install.cmake Makefile CMakeFiles CMakeCache.txt

//FileName := CMakeLists.txt

cmake\_minimum\_required(VERSION 3.2)

project(hust\_\_)

set(CMAKE\_CXX\_STANDARD 14)

set(CMAKE\_C\_STANDARD 11)

set(CMAKE\_VERBOSE\_MAKEFILE ON)

set(CMAKE\_CXX\_FLAGS\_DEBUG "-g -DMALLOC\_CHECK\_=2")

set(CMAKE\_CXX\_FLAGS\_RELEASE "-O3")

include\_directories("/usr/include")

include\_directories("/usr/local/include")

include\_directories(".")

### create a custom target called build\_libr that is part of ALL

### and will run each time you type make

##add\_custom\_target(build\_libr ALL

## COMMAND make

## WORKING\_DIRECTORY rlib

## COMMENT "Calling rlib makefile to build libr.a")

add\_library(r STATIC rlib/libr.cc)

set(BUILD\_SRC main.cc reflected\_impl.hpp btree.hpp general\_ui.hpp parser.hpp)

add\_executable(exp3 ${BUILD\_SRC})

##add\_dependencies(exp3 build\_libr)

target\_link\_libraries(exp3 r)//FileName := general\_ui.hpp

#ifndef HUST\_\_\_GENERAL\_UI\_HPP\_

#define HUST\_\_\_GENERAL\_UI\_HPP\_

#include <functional>

#include <string>

#include <iostream>

#include <list>

#include <rlib/stdio.hpp>

#include <rlib/terminal.hpp>

#include <rlib/string/string.hpp>

#include <rlib/sys/os.hpp>

using namespace rlib::terminal;

using rlib::splitString;

class fake\_terminal {

public:

using callback\_t = std::function<void (std::vector<std::string>)>;

static void showError(const std::string &msg) {

rlib::printfln("{}{}Error{}{}: {}{}", color\_t::red, font\_t::bold, clear, color\_t::lightgray, msg, clear);

}

[[noreturn]] static void go(const callback\_t &callback) {

callback(splitString("help"));

while(true) {

prompt();

try {

callback(splitString(rlib::scanln()));

}

catch(std::exception &e) {

showError(e.what());

}

if(std::cin.eof())

std::exit(0);

}

}

private:

static void prompt() {

if constexpr(rlib::OSInfo::os == rlib::OSInfo::os\_t::WINDOWS) {

rlib::printf("rfaketerm 0.2 ~");

}

else {

rlib::printf("{}rfaketerm 0.2{} {}~{} ", color\_t::green, clear, font\_t::bold, clear);

}

}

};

#endif

//FileName := input

Assign 1

Assign L 4

Assign R 2

Assign LR 32

Assign LL 22

Assign RL 21

Assign LRL 324

......省略大约500行

InOrderTraverse

CreateBiTree

Select 1

Assign 10

Assign L 40

Assign R 20

Assign LR 320

Assign LL 220

Assign RL 210

Assign LRL 3240

......省略大约500行

Select 0

InsertChild LL 1 0

PreOrderTraverse

//FileName := main.cc

#include <general\_ui.hpp>

#include <parser.hpp>

reflected\_impl impl;

int main() {

fake\_terminal::go(parser::parse);

}//FileName := parser.hpp

#ifndef \_HUST\_\_\_PARSER\_HPP

#define \_HUST\_\_\_PARSER\_HPP 1

#include <reflected\_impl.hpp>

#include <list>

#include <string>

#include <iomanip>

#include <rlib/stdio.hpp>

#include <rlib/terminal.hpp>

using namespace rlib::terminal;

class parser {

private:

static void help\_msg() {

std::string msg = R"\_STR\_(

rfaketerm 0.2 HUST\_xxxx special edition

>>> Usage: <Command> [args ...]

>>> Command List:

CommandName [Arguments ...] -> ReturnValue # Instructions

help -> null # Show this message

exit -> null # exit politely

Select [int i] -> null # Select which btree to use (Select 0 by default, index starts from zero)

List -> null # List how many btree is working currently

InitBiTree -> null

DestroyBiTree -> null

CreateBiTree -> null

ClearBiTree -> null

BiTreeEmpty -> bool

BiTreeDepth -> int

Root -> NodeLanguage

Value [NodeLanguage n] -> data\_t

Assign [NodeLanguage n, data\_t val] -> null

Parent [NodeLanguage n] -> NodeLanguage

LeftChild [NodeLanguage n] -> NodeLanguage

RightChild [NodeLanguage n] -> NodeLanguage

LeftSibling [NodeLanguage n] -> NodeLanguage

RightSibling [NodeLanguage n] -> NodeLanguage

InsertChild [NodeLanguage n, int toInsert, int LR] -> null # toInsert is index of btree to insert, start from zero, in `List`

DeleteChild [NodeLanguage n, int LR] -> null

PreOrderTraverse -> null

InOrderTraverse -> null

PostOrderTraverse -> null

LevelOrderTraverse -> null

>>> What's NodeLanguage?

NodeLanguage is a string language, with which you can appoint a node in a tree easily and quickly.

Example: assume you have a tree like this now,

A

/ \

B C

/ \ \

F G H

/ /

J X

Then you can use NodeLanguage to represent every node:

A = ""

B = "L"

C = "R"

F = "LL"

G = "LR"

H = "RR"

J = "LRL"

X = "RRL"

Every 'L' and 'R' represents a step, and you can reach the node step by step.

You can also appoint a not existing node, sothat you can insert a node here. But all node in the path must exists, here're examples:

Assign(Y, "RRLR"); //Good

Assign(D, "RLL"); //Bad, "RL" not exist

Assign(M, "L"); //Valid, B is erased and M is assigned

Assign(N , " LR L L "); //Valid, extra spaces are allowed in NodeLanguage

So you can build a tree quickly in my terminal like this:

rfaketerm ~ Assign 1

rfaketerm ~ Assign L 3

rfaketerm ~ Assign R 22

rfaketerm ~ Assign LR 11

)\_STR\_";

rlib::println(msg);

}

public:

static void parse(const std::vector<std::string> &to\_parse) {

if(to\_parse.empty())

return;

rlib::print(std::boolalpha);

#define AREA\_BEGIN if(to\_parse.begin()->empty()) {}

#define IFCMD(str) else if(\*to\_parse.begin() == str)

#define AREA\_END else

#define WANT\_ARG(n) if(to\_parse.size() != n+1) {throw std::runtime\_error(rlib::format\_string("{} arguments wanted but {} provided.", n, to\_parse.size()-1));}

#define STRING\_ARG(n) to\_parse[n]

#define SIZE\_ARG(n) std::stoul(to\_parse[n])

#define INT\_ARG(n) std::stoi(to\_parse[n])

#define HAVE\_RETURN\_VALUE auto ret =

#define PRINT\_RETURN\_VALUE rlib::println(ret);

AREA\_BEGIN

//Code generated by ccgen.py below. Do not edit them by hand.

//\_\_ccgen\_debug\_\_: `ret name(args)` is `null Select(size\_t i)`

IFCMD("Select") {

WANT\_ARG(1)

impl.Select(SIZE\_ARG(1));

}

//\_\_ccgen\_debug\_\_: `ret name(args)` is `null List()`

IFCMD("List") {

WANT\_ARG(0)

impl.List();

}

//\_\_ccgen\_debug\_\_: `ret name(args)` is `null InitBiTree()`

IFCMD("InitBiTree") {

WANT\_ARG(0)

impl.InitBiTree();

}

//\_\_ccgen\_debug\_\_: `ret name(args)` is `null DestroyBiTree()`

IFCMD("DestroyBiTree") {

WANT\_ARG(0)

impl.DestroyBiTree();

}

//\_\_ccgen\_debug\_\_: `ret name(args)` is `null CreateBiTree()`

IFCMD("CreateBiTree") {

WANT\_ARG(0)

impl.CreateBiTree();

}

//\_\_ccgen\_debug\_\_: `ret name(args)` is `null ClearBiTree()`

IFCMD("ClearBiTree") {

WANT\_ARG(0)

impl.ClearBiTree();

}

//\_\_ccgen\_debug\_\_: `ret name(args)` is `bool BiTreeEmpty()`

IFCMD("BiTreeEmpty") {

WANT\_ARG(0)

HAVE\_RETURN\_VALUE

impl.BiTreeEmpty();

PRINT\_RETURN\_VALUE

}

//\_\_ccgen\_debug\_\_: `ret name(args)` is `size\_t BiTreeDepth()`

IFCMD("BiTreeDepth") {

WANT\_ARG(0)

HAVE\_RETURN\_VALUE

impl.BiTreeDepth();

PRINT\_RETURN\_VALUE

}

//\_\_ccgen\_debug\_\_: `ret name(args)` is `NodeLanguage Root()`

IFCMD("Root") {

WANT\_ARG(0)

HAVE\_RETURN\_VALUE

impl.Root();

PRINT\_RETURN\_VALUE

}

//\_\_ccgen\_debug\_\_: `ret name(args)` is `data\_t Value(NodeLanguage n)`

IFCMD("Value") {

WANT\_ARG(1)

HAVE\_RETURN\_VALUE

impl.Value(STRING\_ARG(1));

PRINT\_RETURN\_VALUE

}

//\_\_ccgen\_debug\_\_: `ret name(args)` is `null Assign(NodeLanguage n, data\_t val)`

IFCMD("Assign") {

WANT\_ARG(2)

impl.Assign(STRING\_ARG(1), INT\_ARG(2));

}

//\_\_ccgen\_debug\_\_: `ret name(args)` is `NodeLanguage Parent(NodeLanguage n)`

IFCMD("Parent") {

WANT\_ARG(1)

HAVE\_RETURN\_VALUE

impl.Parent(STRING\_ARG(1));

PRINT\_RETURN\_VALUE

}

//\_\_ccgen\_debug\_\_: `ret name(args)` is `NodeLanguage LeftChild(NodeLanguage n)`

IFCMD("LeftChild") {

WANT\_ARG(1)

HAVE\_RETURN\_VALUE

impl.LeftChild(STRING\_ARG(1));

PRINT\_RETURN\_VALUE

}

//\_\_ccgen\_debug\_\_: `ret name(args)` is `NodeLanguage RightChild(NodeLanguage n)`

IFCMD("RightChild") {

WANT\_ARG(1)

HAVE\_RETURN\_VALUE

impl.RightChild(STRING\_ARG(1));

PRINT\_RETURN\_VALUE

}

//\_\_ccgen\_debug\_\_: `ret name(args)` is `NodeLanguage LeftSibling(NodeLanguage n)`

IFCMD("LeftSibling") {

WANT\_ARG(1)

HAVE\_RETURN\_VALUE

impl.LeftSibling(STRING\_ARG(1));

PRINT\_RETURN\_VALUE

}

//\_\_ccgen\_debug\_\_: `ret name(args)` is `NodeLanguage RightSibling(NodeLanguage n)`

IFCMD("RightSibling") {

WANT\_ARG(1)

HAVE\_RETURN\_VALUE

impl.RightSibling(STRING\_ARG(1));

PRINT\_RETURN\_VALUE

}

//\_\_ccgen\_debug\_\_: `ret name(args)` is `null InsertChild(NodeLanguage n, size\_t toInsert, size\_t LR)`

IFCMD("InsertChild") {

WANT\_ARG(3)

impl.InsertChild(STRING\_ARG(1), SIZE\_ARG(2), SIZE\_ARG(3));

}

//\_\_ccgen\_debug\_\_: `ret name(args)` is `null DeleteChild(NodeLanguage n, size\_t LR)`

IFCMD("DeleteChild") {

WANT\_ARG(2)

impl.DeleteChild(STRING\_ARG(1), SIZE\_ARG(2));

}

//\_\_ccgen\_debug\_\_: `ret name(args)` is `null PreOrderTraverse()`

IFCMD("PreOrderTraverse") {

WANT\_ARG(0)

impl.PreOrderTraverse();

}

//\_\_ccgen\_debug\_\_: `ret name(args)` is `null InOrderTraverse()`

IFCMD("InOrderTraverse") {

WANT\_ARG(0)

impl.InOrderTraverse();

}

//\_\_ccgen\_debug\_\_: `ret name(args)` is `null PostOrderTraverse()`

IFCMD("PostOrderTraverse") {

WANT\_ARG(0)

impl.PostOrderTraverse();

}

//\_\_ccgen\_debug\_\_: `ret name(args)` is `null LevelOrderTraverse()`

IFCMD("LevelOrderTraverse") {

WANT\_ARG(0)

impl.LevelOrderTraverse();

}

IFCMD("exit") {

rlib::println("bye~");

::std::exit(0);

}

IFCMD("help") {

help\_msg();

}

//Code generated by ccgen.py ahead. Do not edit them by hand.

AREA\_END {

throw std::invalid\_argument("Invalid argument. Try to type `help` to get helped.");

}

}

};

#endif //\_HUST\_\_\_PARSER\_HPP

//FileName := reflected\_impl.hpp

#ifndef HUST\_\_\_REFLECTED\_IMPL\_HPP\_

#define HUST\_\_\_REFLECTED\_IMPL\_HPP\_

#include <utility>

#include <functional>

#include <algorithm>

#include <vector>

#include "btree.hpp"

#include <rlib/stdio.hpp>

//class reflected\_impl {

//public:

// using data\_t = int;

// using BooleanAsserter = std::function<bool(const data\_t &)>;

// using OperationVisiter = std::function<void(const data\_t &)>;

//

// void InitList() const {}

// void DestroyList() {container.clear();}

// void ClearList() {container.clear();}

// bool ListEmpty() const {return container.size() == 0;}

// size\_t ListLength() const {return container.size();}

// data\_t GetElem(size\_t \_\_\_IndexPlusOne) {

// auto index = \_\_\_IndexPlusOne - 1;

// auto iter = container.begin();

// for(size\_t cter = 0; cter < index; ++cter) {

// ++iter;

// }

// return std::move(\*iter);

// }

// size\_t \_LocateElem(const BooleanAsserter &comparer) {

// auto iter = std::find\_if(container.begin(), container.end(), comparer);

// if(iter == container.end()) {

// return 0;

// }

// return LabUtils::distance(container.begin(), iter);

// }

// size\_t LocateElem(data\_t val) {

// auto comparer = BooleanAsserter([v=val](const data\_t &dat){

// return dat == v;

// });

// return \_LocateElem(comparer);

// }

// data\_t PriorElem(data\_t tofind) {

// auto pos = std::find(container.begin(), container.end(), tofind);

// if(pos == container.end() || pos == container.begin()) {

// throw std::runtime\_error("ElemError: You told me that it's undefined, so I do it.");

// }

// return \*(--pos);

// }

// data\_t NextElem(data\_t tofind) {

// auto pos = std::find(container.begin(), container.end(), tofind);

// if(pos == container.end() || pos == --container.end()) {

// throw std::runtime\_error("ElemError: You told me that it's undefined, so I do it.");

// }

// return \*(++pos);

// }

// void ListInsert(size\_t \_\_\_IndexPlusOne, data\_t elem) {

// auto index = \_\_\_IndexPlusOne - 1;

// auto iter = LabUtils::advance(container.begin(), index);

// container.insert(iter, elem);

// }

// data\_t ListDelete(size\_t \_\_\_IndexPlusOne) {

// auto index = \_\_\_IndexPlusOne - 1;

// auto iter = LabUtils::advance(container.begin(), index);

// auto to\_return = \*iter;

// container.erase(iter);

// return std::move(to\_return);

// }

// void \_ListTraverse(const OperationVisiter &visiter) {

// std::for\_each(container.begin(), container.end(), visiter);

// }

// void ListTraverse() {

// \_ListTraverse(OperationVisiter([](const auto &val){rlib::io::print(val, " ");}));

// rlib::io::println("");

// }

//

// void debug() {

// rlib::io::println\_iter(container);

// rlib::io::println(container.size());

// }

//private:

// Lab::list<data\_t> container;

//};

using hust\_xxxx::unordered\_btree;

class reflected\_impl {

public:

using data\_t = int;

using dataref\_t = const data\_t &;

using nlang = std::string;

using nlangref = const nlang &;

reflected\_impl() : containers(1), current(containers.begin()) {}

//\_\_py\_ccgen\_begin\_\_

void Select(size\_t i) {current = containers.begin() + i;}

void List() {rlib::printfln("You have {} btree now, selecting {}.", containers.size(), current - containers.begin());}

void InitBiTree() {}

void DestroyBiTree() {containers.erase(current); current = containers.begin();}

void CreateBiTree() {containers.push\_back(unordered\_btree<data\_t>());}

void ClearBiTree() {current->clear();}

bool BiTreeEmpty() {return current->empty();}

size\_t BiTreeDepth() {return current->depth();}

nlang Root() {return current->\_root();}

data\_t Value(nlangref n) {return current->get(n);}

void Assign(nlangref n, dataref\_t val) {return current->set(n, val);}

nlang Parent(nlangref n) {return current->parent(n);}

nlang LeftChild(nlangref n) {return current->lchild(n);}

nlang RightChild(nlangref n) {return current->rchild(n);}

nlang LeftSibling(nlangref n) {return current->lchild(current->parent(n));}

nlang RightSibling(nlangref n) {return current->rchild(current->parent(n));}

void InsertChild(nlangref n, size\_t toInsert, size\_t LR) {return current->merge(containers[toInsert], n, LR==1);}

void DeleteChild(nlangref n, size\_t LR) {return current->drop(n, LR==1);}

void PreOrderTraverse() {current->for\_each(unordered\_btree<data\_t>::printer, hust\_xxxx::foreach\_rule::MIDDLE\_LEFT\_RIGHT);}

void InOrderTraverse() {current->for\_each(unordered\_btree<data\_t>::printer, hust\_xxxx::foreach\_rule::LEFT\_MIDDLE\_RIGHT);}

void PostOrderTraverse() {current->for\_each(unordered\_btree<data\_t>::printer, hust\_xxxx::foreach\_rule::LEFT\_RIGHT\_MIDDLE);}

void LevelOrderTraverse() {current->level\_for\_each(unordered\_btree<data\_t>::printer);}

//\_\_py\_ccgen\_end\_\_

private:

std::vector<unordered\_btree<data\_t>> containers;

decltype(containers.begin()) current;

};

extern reflected\_impl impl;

#endif

//FileName := rlib

cat: rlib: 是一个目录

3.3.2 算法测试

由于本次实验测试程序并不好写，也没有提前写好的测试程序(各种库的树实现当然都是平衡树)，只用复制粘贴的方法生成了1000多个测试样例，可以初步说明程序的鲁棒性。

`cmake . -DCMAKE\_BUILD\_TYPE=Release ; and make ; and ./exp3 < input`

其中input的内容为

Assign 1

Assign L 4

Assign R 2

Assign LR 32

Assign LL 22

Assign RL 21

Assign LRL 324

Assign L 1

Assign LL 4

Assign LR 2

Assign LLR 32

Assign LLL 22

Assign LRL 21

Assign LLRL 324

Assign RL 4

Assign RR 2

Assign RLR 32

Assign LRLRLL 22

Assign LRLRRL 21

Assign LRLRLRL 324

Assign LRL 1

Assign LRLL 4

Assign LRLR 2

Assign LRLLR 32

Assign LRLLL 22

Assign LRLRL 21

Assign LRLLRL 324

Assign LRLL 1

Assign LRLLL 4

Assign LRLLR 2

Assign LRLLLR 32

Assign LRLLLL 22

Assign LRLLRL 21

Assign LRLLLRL 324

Assign RLRLRLRL 4

Assign RLR

此处省略1100行 内容见原文件

LRLLRRLRRRL 210

Assign LRLRRLRLRLLRRLRRLRL 3240

Assign LRLRRLRLRLLRLLL 10

Assign LRLRRLRLRLLRLLLL 40

Assign LRLRRLRLRLLRLLLR 20

Assign LRLRRLRLRLLRLLLLR 320

Assign LRLRRLRLRLLRLLLLL 220

Assign LRLRRLRLRLLRLLLRL 210

Assign LRLRRLRLRLLRLLLLRL 3240

Assign LRLRRLRLRLLRLLLL 10

Assign LRLRRLRLRLLRLLLLL 40

Assign LRLRRLRLRLLRLLLLR 20

Assign LRLRRLRLRLLRLLLLLR 320

Assign LRLRRLRLRLLRLLLLLL 220

Assign LRLRRLRLRLLRLLLLRL 210

Assign LRLRRLRLRLLRLLLLLRL 3240

Assign LRLRRLRLRLLRLLLRL 40

Assign LRLRRLRLRLLRLLLRR 20

Assign LRLRRLRLRLLRLLLRLR 320

Assign LRLRRLRLRLLRLLLLRLRLL 220

Assign LRLRRLRLRLLRLLLLRLRRL 210

Assign LRLRRLRLRLLRLLLLRLRLRL 3240

Assign LRLRRLRLRLLRLLLLRL 10

Assign LRLRRLRLRLLRLLLLRLL 40

Assign LRLRRLRLRLLRLLLLRLR 20

Assign LRLRRLRLRLLRLLLLRLLR 320

Assign LRLRRLRLRLLRLLLLRLLL 220

Assign LRLRRLRLRLLRLLLLRLRL 210

Assign LRLRRLRLRLLRLLLLRLLRL 3240

Assign LRLRRLRLRLLRLLLLRLL 10

Assign LRLRRLRLRLLRLLLLRLLL 40

Assign LRLRRLRLRLLRLLLLRLLR 20

Select 0

InsertChild LL 1 0

PreOrderTraverse

程序进行这些操作并没有出现错误。由于输出过长，此处无法展示，请直接运行测试。

3.3.3 界面测试

rfaketerm和中间的每一层中间层都复用了过去的框架，采用了较好的实现方式和架构，同时使用了代码自动生成，其已经经历多次实验的考验。简单的测试表明，界面的正确性没有问题。

**3.4 实验小结**

此次实验相比上次做了以下更新：

rlib更新了stdio.hpp，加入了更接近python的fmt风格。进行了重构，由纯头库变为含部分静态库，解决了符号冲突的隐患。对所有子模块完善了对不同C++的检测，增加编译的鲁棒性。以及其他小的修复和接口更新。

实验程序框架方面，增加了简单的代码生成器和帮助信息生成器，可以增加开发速度和程序可靠性。完善了异常处理，增加了用户的异常时体验。完善了对信号和EOF的处理规则更新了，使得rfaketerm成为一个支持简单脚本的shell，极大的便利了从外部的自动化测试和自动化任务。其他大量的不完善细节。

本次实验加深了对二叉树的概念、基本运算的理解，掌握了二叉树的基本预算的实现。熟练了二元树的逻辑结构和物理结构之间的关系。今后的学习过程中应当多从数据结构的角度分析如何进行数据的处理、存储以方便问题的解决，并要勤加练习达到熟能生巧的地步。

# 4基于邻接表的图实现

## 4.1 实验目的

通过实验达到⑴加深对图的概念、基本运算的理解；⑵熟练掌握图的逻辑结构与物理结构的关系；⑶以邻接表作为物理结构，熟练掌握图基本运算的实现。

**4.2 系统设计**

4.2.1 系统总体设计

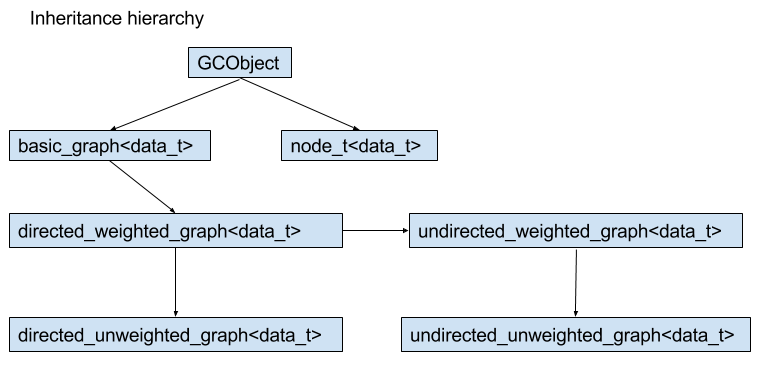
本系统实现图的基本运算。遵守C++14标准。

系统具有一个Terminal风格交互界面，称为rfaketerm，在general\_ui.hpp中实现。fake\_terminal::go会阻塞主线程，接收输入，简单parse之后通过callback函数进行处理。callback是一个由ccgen.py生成代码的parser(即reflection,C++20标准库提供了原生功能)，负责将输入翻译到下一层即relected\_impl。它将请求进一步解释，并与后端数据结构进行交互，获取返回值，被rfaketerm打印到stdout。在程序发生未定义行为时，会通过std::exception向自身发送*SIGABRT*信号，这有利于通用调试工具的应用。为了美观，rfaketerm默认情况下会把所有异常抓下并打印错误信息到stdout。

User Manual在 rfaketerm中使用help命令即可获得。为了便于GUI下的使用，rfaketerm启动时会自动模拟执行help命令。

系统定义一个reflection\_impl(作为本题要求的接口和容器库普遍承认的接口之间的wrapper)，其负责管理数据结构对象hust\_xxxx::basic\_graph。为了实现对多个graph的管理，只需使用std::vector<basic\_graph>即可。继承关系如下。

basic\_graph可以选择使用std::list或std::vector进行存储。为了低成本的保证地址的有效性，在使用std::list时性能较差，但支持所有操作中地址有效。使用std::vector时删除操作会使地址失效，因此此操作被禁用，但它的DFS和BFS比前者快一个Θ(n)因子。在4.3.3详细比较。



该演示系统提供的操作有：创建图、销毁图、查找顶点、获得顶点值和顶点赋值等13种基本运算和Select, List等用于在多个树间切换的操作，详见help。

在程序中实现消息处理和操作提示，包括数据的输入和输出，错误操作提示、程序的退出。

4.2.2 算法设计

依据最小完备性和常用性相结合的原则，以函数形式定义了二叉树的初始化二叉树、销毁二叉树、创建二叉树、清空二叉树、判定空二叉树和求二叉树深度等20种基本运算，具体运算功能定义如下。

⑴创建图：函数名称是CreateGraph(&G,V,VR)；初始条件是V是图的顶点集，VR是图的关系集；操作结果是按V和VR的定义构造图G。

⑵销毁图：树函数名称是DestroyGraph(T)；初始条件图G已存在；操作结果是销毁图G。

⑶查找顶点：函数名称是LocateVex(G,u)；初始条件是图G存在，u和G中的顶点具有相同特征；操作结果是若u在图G中存在，返回顶点u的位置信息，否则返回其它信息。

⑷获得顶点值：函数名称是GetVex (G,v)；初始条件是图G存在，v是G中的某个顶点；操作结果是返回v的值。

⑸顶点赋值：函数名称是PutVex (G,v,value)；初始条件是图G存在，v是G中的某个顶点；操作结果是对v赋值value。

⑹获得第一邻接点：函数名称是FirstAdjVex(&G, v)；初始条件是图G存在，v是G的一个顶点；操作结果是返回v的第一个邻接顶点，如果v没有邻接顶点，返回“空”。

⑺获得下一邻接点：函数名称是NextAdjVex(&G, v, w)；初始条件是图G存在，v是G的一个顶点,w是v的邻接顶点；操作结果是返回v的（相对于w）下一个邻接顶点，如果w是最后一个邻接顶点，返回“空”。

⑻插入顶点：函数名称是InsertVex(&G,v)；初始条件是图G存在，v和G中的顶点具有相同特征；操作结果是在图G中增加新顶点v。

⑼删除顶点：函数名称是DeleteVex(&G,v)；初始条件是图G存在，v是G的一个顶点；操作结果是在图G中删除顶点v和与v相关的弧。

⑽插入弧：函数名称是InsertArc(&G,v,w)；初始条件是图G存在，v、w是G的顶点；操作结果是在图G中增加弧<v,w>，如果图G是无向图，还需要增加<w,v>。

⑾删除弧：函数名称是DeleteArc(&G,v,w)；初始条件是图G存在，v、w是G的顶点；操作结果是在图G中删除弧<v,w>，如果图G是无向图，还需要删除<w,v>。

⑿深度优先搜索遍历：函数名称是DFSTraverse(G,visit())；初始条件是图G存在；操作结果是图G进行深度优先搜索遍历，依次对图中的每一个顶点使用函数visit访问一次，且仅访问一次。

⒀广深度优先搜索遍历：函数名称是BFSTraverse(G,visit())；初始条件是图G存在；操作结果是图G进行广度优先搜索遍历，依次对图中的每一个顶点使用函数visit访问一次，且仅访问一次。

**4.3 图演示系统实现与测试**

4.3.1 系统实现

编程环境：Linux x86\_64 ARCH gcc 8.0.0 cmake 3.10.1 GNU Make 4.2.1 GNU ld 2.29.1 GNU ar 2.29.1 kernel 4.14.11-1-ARCH 其他环境设定均在CMakeLists.txt进行了说明。

为Windows进行了交叉编译，使用cmake 3.10.0 mingw-gcc 6.3.1 nmake Windows 10 1709 (summer creator update) 静态编译使用mingw-gcc 6.3.1提供的libstdc++。Windows版本缺失部分功能(界面美化)。

使用了gc库。编译前请阅读README.md。使用testgen.py生成性能测试所用的测试样例。

下面是src目录下的hpp/cc/CMakeLists.txt文件清单：依赖于rlib，此库被打包进源码目录，库内容均为原创。其中包含了测试所用代码。

//ccgen.py

#!/usr/bin/env python3

import sys

if len(sys.argv) != 2:

print('Usage: `./ccgen.py code` or `./ccgen.py help`')

exit(1)

src = 'reflected\_impl.hpp'

mode = sys.argv[1]

# DO NOT use macro in func\_name! It'll gen wrong code!

macro\_list = [

('langref\_t','lang\_t'),

('lang\_t','Language'),

('dataref\_t','data\_t'),

('void','null'),

]

size\_arg = ['size\_t']

int\_arg = ['int', 'data\_t']

string\_arg = ['Language']

void\_ret = ['void', 'null']

def gen\_code(line):

line = line.replace('\t','').replace('\r', '').strip()

if len(line) == 0:

return

ret\_type = line.split(' ')[0]

funcAndArgs = line[len(ret\_type):].strip().split('(')

func\_name, args = funcAndArgs[0], funcAndArgs[1].split(')')[0]

print('//\_\_ccgen\_debug\_\_: `ret name(args)` is `{} {}({})`'.format(ret\_type, func\_name, args))

args\_string = []

for arg in args.split(','):

arg\_type = arg.strip().split(' ')[0].replace(' ','')

if len(arg\_type) == 0:

continue

if arg\_type in size\_arg:

args\_string.append('SIZE\_ARG({})'.format(len(args\_string)+1)) # start from one

elif arg\_type in int\_arg:

args\_string.append('INT\_ARG({})'.format(len(args\_string)+1)) # start from one

elif arg\_type in string\_arg:

args\_string.append('STRING\_ARG({})'.format(len(args\_string)+1)) # start from one

else:

raise RuntimeError('Unclassed arg left here. line={}|arg\_type={}'.format(line, arg\_type))

args\_size = len(args\_string)

args\_string = ', '.join(args\_string)

print(' IFCMD("{}") {{'.format(func\_name))

print(' WANT\_ARG({})'.format(args\_size))

if ret\_type not in void\_ret:

print(' HAVE\_RETURN\_VALUE')

print(' impl.{}({});'.format(func\_name, args\_string))

if ret\_type not in void\_ret:

print(' PRINT\_RETURN\_VALUE')

print(' }')

def gen\_help(line):

line = line.replace('\t','').replace('\r', '').strip()

if len(line) == 0:

return

ret\_type = line.split(' ')[0]

funcAndArgs = line[len(ret\_type):].strip().split('(')

func\_name, args = funcAndArgs[0], funcAndArgs[1].split(')')[0]

# print('//\_\_ccgen\_debug\_\_: `ret name(args)` is `{} {}({})`'.format(ret\_type, func\_name, args))

if len(args) == 0:

print('{} -> {}'.format(func\_name, ret\_type))

else:

print('{} [{}] -> {}'.format(func\_name, args, ret\_type))

if mode == 'code':

fuck\_a\_line = gen\_code

print('//Code generated by ccgen.py below. Do not edit them by hand.')

else:

fuck\_a\_line = gen\_help

print('FuncName [Argument ...] -> ReturnValue # Instructions')

with open(src) as fd:

cont = fd.read()

working = False

for line in cont.split('\n'):

if -1 != line.find('\_\_py\_ccgen\_begin\_\_'):

working = True

continue

if -1 != line.find('\_\_py\_ccgen\_end\_\_'):

working = False

continue

if working:

for \_from, \_to in macro\_list:

line = line.replace(\_from, \_to)

fuck\_a\_line(line)

if mode != 'code':

exit(0)

print('''

IFCMD("exit") {

rlib::println("bye~");

::std::exit(0);

}

IFCMD("help") {

help\_msg();

}

//impl.debug();

//Code generated by ccgen.py ahead. Do not edit them by hand.

''')

//cmake\_clean.sh

#!/bin/bash

make clean

rm -rf cmake-build-debug/ cmake\_install.cmake Makefile CMakeFiles CMakeCache.txt

//CMakeLists.txt

cmake\_minimum\_required(VERSION 3.2)

project(hust\_\_)

set(CMAKE\_CXX\_STANDARD 14)

set(CMAKE\_C\_STANDARD 11)

set(CMAKE\_VERBOSE\_MAKEFILE ON)

set(CMAKE\_CXX\_FLAGS\_DEBUG "-g -DMALLOC\_CHECK\_=2")

set(CMAKE\_CXX\_FLAGS\_RELEASE "-O3")

# Much higher performance, but not supports DeleteVex(removeNode).

# set(CMAKE\_CXX\_FLAGS "${CMAKE\_CXX\_FLAGS} -DCOMPILE\_NO\_ERASE -DNODES\_PRE\_ALLOC\_MEM=1000000")

set(THREADS\_PREFER\_PTHREAD\_FLAG ON)

find\_package(Threads REQUIRED)

include\_directories("/usr/include")

include\_directories("/usr/local/include")

include\_directories(".")

### create a custom target called build\_libr that is part of ALL

### and will run each time you type make

##add\_custom\_target(build\_libr ALL

## COMMAND make

## WORKING\_DIRECTORY rlib

## COMMENT "Calling rlib makefile to build libr.a")

add\_library(r STATIC rlib/libr.cc)

set(BUILD\_SRC main.cc reflected\_impl.hpp general\_ui.hpp parser.hpp graph.hpp gc/gc.cpp gc/gc.h indexed\_list.hpp)

add\_executable(exp4 ${BUILD\_SRC})

##add\_dependencies(exp4 build\_libr)

target\_link\_libraries(exp4 r)

target\_link\_libraries(exp4 Threads::Threads)

//gc

cat: gc: 是一个目录

//general\_ui.hpp

#ifndef HUST\_\_\_GENERAL\_UI\_HPP\_

#define HUST\_\_\_GENERAL\_UI\_HPP\_

#include <functional>

#include <string>

#include <iostream>

#include <list>

#include <rlib/stdio.hpp>

#include <rlib/terminal.hpp>

#include <rlib/string/string.hpp>

#include <rlib/sys/os.hpp>

using namespace rlib::terminal;

using rlib::splitString;

class fake\_terminal {

public:

using callback\_t = std::function<void (std::vector<std::string>)>;

static void showError(const std::string &msg) {

rlib::printfln("{}{}Error{}{}: {}{}", color\_t::red, font\_t::bold, clear, color\_t::lightgray, msg, clear);

}

[[noreturn]] static void go(const callback\_t &callback) {

callback(splitString("help"));

bool scripting = false;

while(true) {

if(!scripting)

prompt();

try {

auto cont = rlib::scanln();

if(cont.find("#!") != std::string::npos) { //Remove annoying prompt while scripting.

rlib::println();

scripting = true;

}

size\_t pos = cont.find('#');

if(pos != std::string::npos)

cont = cont.substr(0, pos); //Remove comments. Avoid rlib::splitString to make it faster.

callback(splitString(cont));

}

catch(std::exception &e) {

showError(e.what());

}

if(std::cin.eof())

std::exit(0);

}

}

private:

static void prompt() {

if(rlib::OSInfo::os == rlib::OSInfo::os\_t::WINDOWS) {

rlib::printf("rfaketerm 0.2 ~");

}

else {

rlib::printf("{}rfaketerm 0.2{} {}~{} ", color\_t::green, clear, font\_t::bold, clear);

}

}

};

#endif

//graph.hpp

#ifndef HUST\_XXXX\_GRAPH\_HPP\_

#define HUST\_XXXX\_GRAPH\_HPP\_

#include <gc/gc.h>

#include <rlib/string/string.hpp>

#include <list>

#include <vector>

#include <string>

#include <iomanip>

#include <sstream>

#include <cstdint>

#include <stdexcept>

#include <exception>

#include <algorithm>

#include <queue>

#include <stack>

#include <unordered\_map>

// Warning: namespace pollution

using namespace std::string\_literals;

namespace hust\_xxxx {

template <typename data\_t>

class basic\_graph : public GCObject {

public:

using weight\_t = uint32\_t;

struct node\_t;

using edge\_t = std::pair<node\_t \*, weight\_t>;

struct node\_t : public GCObject {

node\_t() = default;

explicit node\_t(const data\_t &dat) : dat(dat) {}

data\_t dat;

std::vector<edge\_t> neighbors;

};

#ifdef COMPILE\_NO\_ERASE

#ifndef NODES\_PRE\_ALLOC\_MEM

#define NODES\_PRE\_ALLOC\_MEM 10000000

#endif

basic\_graph() {nodes.reserve(NODES\_PRE\_ALLOC\_MEM);}

#else

basic\_graph() = default;

#endif

virtual ~basic\_graph() = default;

protected:

template <typename \_data\_t>

static \_data\_t stringToDataObj(const std::string &str) {

std::stringstream ss;

\_data\_t val;

ss << str;

ss >> val;

return std::move(val);

}

static std::string dataObjToString(const data\_t &dat) {

std::stringstream ss;

ss << dat;

return std::move(ss.str());

}

std::string deAlias(const std::string &addr) {

if(addr.empty())

return addr;

try {

return nodeAlias.at(addr);

}

catch (std::out\_of\_range &) {

return addr;

}

}

std::string toNodeLanguage(const node\_t &node) {

return std::move(rlib::format\_string("{}`{}{}{}", node.dat, std::hex, (uint64\_t)(&node), std::dec));

}

std::string toEdgeLanguage(const node\_t &from, const edge\_t &to) {

return std::move(rlib::format\_string("{}`{}{}`{}{}", to.second, std::hex, (uint64\_t)&from, (uint64\_t)&to.first, std::dec));

}

auto fromNodeLanguage(const std::string &lang, bool newIfInvalidAddr = false, bool assignIfNew = false, bool assignIfExist = false) {

auto datAndAddr = rlib::splitString(lang, '`');

if(datAndAddr.size() != 2)

throw std::invalid\_argument("fromNodeLanguage want a nodeLanguage with address, but got bad format.");

data\_t val = stringToDataObj<data\_t>(datAndAddr[0]);

static\_assert(std::is\_same<uint64\_t, unsigned long>::value, "unsigned long isn't uint64\_t");

try {

uint64\_t addr = std::stoul(deAlias(datAndAddr[1]), nullptr, 16);

auto target = nodes.end();

try {

target = nodePointerToIter(reinterpret\_cast<node\_t \*>(addr));

}

catch(std::invalid\_argument &) {}

if(target != nodes.end()) {

if(assignIfExist)

target->dat = val;

return target;

}

}

catch(std::invalid\_argument &) {

// invalid addr, continue to try append.

}

catch(std::out\_of\_range &e) {

// seems valid addr, but out of range.

throw std::out\_of\_range(rlib::format\_string("Address `{}` out\_of\_range, check it!(stoul says {})", datAndAddr[1], e.what()));

}

if(!newIfInvalidAddr)

throw std::invalid\_argument(rlib::format\_string("Can not find node\_t at {}{}", std::hex, datAndAddr[1]));

if(assignIfNew)

nodes.push\_back(node\_t(val));

else

nodes.push\_back(node\_t());

if(!datAndAddr[1].empty()) {

nodeAlias[datAndAddr[1]] = rlib::format\_string("{}{}{}", std::hex, (uint64\_t)&\*--nodes.end(), std::dec);//std::to\_string(reinterpret\_cast<uint64\_t>(&\*(--nodes.end())));

} //appointed alias

return --nodes.end();

}

auto fromEdgeLanguage(const std::string &lang, bool newIfInvalidAddr = false) {

auto arg = rlib::splitString(lang, '`');

if(arg.size() != 3)

throw std::invalid\_argument("bad edge language");

weight\_t val = stringToDataObj<weight\_t>(arg[0]);

static\_assert(std::is\_same<uint64\_t, unsigned long>::value, "unsigned long isn't uint64\_t");

node\_t \*addrFrom = reinterpret\_cast<node\_t \*>(std::stoul(deAlias(arg[1]), nullptr, 16));

node\_t \*addrTo = reinterpret\_cast<node\_t \*>(std::stoul(deAlias(arg[2]), nullptr, 16));

auto target = nodePointerToIter(addrFrom); //throws std::invalid\_argument

nodePointerToIter(addrTo); //Confirm that nodeTo do exists.

auto pos = std::find\_if(target->neighbors.begin(), target->neighbors.end(), [&](const edge\_t &e){

return (uint64\_t)addrTo == (uint64\_t)e.first;

});

if(pos != target->neighbors.end()) {

return pos;

}

else {

if(newIfInvalidAddr) {

target->neighbors.push\_back(std::make\_pair(addrTo, val));

return target->neighbors.end() - 1;

}

else

throw std::invalid\_argument("requested edge not exist");

}

}

// For std::vector<>, O(1) convert...

#ifdef COMPILE\_NO\_ERASE

size\_t nodePointerToIndex(const node\_t \*ptr) {

node\_t \*begin = nodes.data();

if(ptr - begin >= nodes.size() \* sizeof(node\_t) || ptr - begin < 0)

throw std::invalid\_argument("nodePointerToIter failed: not found.");

return ptr - begin;

}

auto nodePointerToIter(const node\_t \*ptr) {

return nodes.begin() + nodePointerToIndex(ptr);

}

#else

//Warning: O(n) is too slow!

size\_t nodePointerToIndex(const node\_t \*ptr) {

size\_t cter = 0;

for(auto iter = nodes.begin(); iter != nodes.end(); ++iter, ++cter) {

if(&\*iter == ptr)

return cter;

}

throw std::invalid\_argument("nodePointerToIndex failed: node not found.");

}

//Warning: O(n) is too slow!

auto nodePointerToIter(const node\_t \*ptr) {

for(auto iter = nodes.begin(); iter != nodes.end(); ++iter) {

if(&\*iter == ptr)

return iter;

}

throw std::invalid\_argument("nodePointerToIndex failed: node not found.");

}

#endif

public:

std::string findNode(const data\_t &val) {

for(auto &node : nodes) {

if(node.dat == val)

return toNodeLanguage(node);

}

return "`";

}

std::string getNodeValue(const std::string &lang) {

return toNodeLanguage(\*fromNodeLanguage(lang));

}

void setNodeValue(const std::string &lang) {

fromNodeLanguage(lang, true, true, true);

}

std::string findFirstNearNode(const std::string &lang) {

auto node = fromNodeLanguage(lang);

return node->neighbors.empty() ? "`" : toNodeLanguage(\*node->neighbors.begin()->first);

}

std::string findNextNearNode(const std::string &centerNd, const std::string &posNd) {

auto center = fromNodeLanguage(centerNd);

auto pos = fromNodeLanguage(posNd);

for(auto iter = center->neighbors.begin(); iter != center->neighbors.end(); ++iter) {

if (iter->first == &\*pos) {

++iter;

if(iter == center->neighbors.end())

return "`";

else

return toNodeLanguage(\*iter->first);

}

}

return "`";

}

void removeNode(const std::string &lang) {

#ifdef COMPILE\_NO\_ERASE

throw std::runtime\_error("This program is compiled as vector version, which gets much higher performance but without supporting removeNode.");

#else

nodes.erase(fromNodeLanguage(lang));

#endif

}

using node\_visiter = std::function<void(node\_t &)>;

const node\_visiter printer = [](node\_t &nd){rlib::printf("{}`{}{}{} ", nd.dat, std::hex, (uint64\_t)&nd, std::dec);};

void dfs(const node\_visiter &func) {

std::vector<bool> masks(nodes.size(), false);

dfs\_helper(func, masks, \*nodes.begin());

}

void dfs\_helper(const node\_visiter &func, std::vector<bool> &masks, node\_t &curr) {

masks[nodePointerToIndex(&curr)] = true;

func(curr);

for(auto &edge : curr.neighbors) {

node\_t &next = \*edge.first;

size\_t index = nodePointerToIndex(&next);

if(!masks[index])

dfs\_helper(func, masks, next);

}

}

void bfs(const node\_visiter &func) {

std::vector<bool> masks(nodes.size(), false);

masks[0] = true; //dfs method can't apply to bfs.

bfs\_helper(func, masks, std::list<node\_t \*>{&\*nodes.begin()});

}

void bfs\_helper(const node\_visiter &func, std::vector<bool> &masks, const std::list<node\_t \*> &curr) {

std::list<node\_t \*> next;

for(node\_t \*node : curr) {

for(auto &edge : node->neighbors) {

node\_t &nextNode = \*edge.first;

size\_t index = nodePointerToIndex(&nextNode);

if(masks[index])

continue;

else {

masks[index] = true;

next.push\_back(&nextNode);

}

}

}

std::for\_each(curr.begin(), curr.end(), [&func](node\_t \*p){func(\*p);});

if(!next.empty())

bfs\_helper(func, masks, next);

}

void simple\_foreach(const node\_visiter &func) {

std::for\_each(nodes.begin(), nodes.end(), func);

}

virtual void insertEdge(const std::string &lang) = 0;

virtual void removeEdge(const std::string &lang) = 0;

protected:

#ifdef COMPILE\_NO\_ERASE

std::vector<node\_t> nodes;

#else

std::list<node\_t> nodes;

#endif

std::unordered\_map<std::string, std::string> nodeAlias;

};

template <typename data\_t>

class directed\_weighted\_graph : public basic\_graph<data\_t> {

public:

using super = basic\_graph<data\_t>;

directed\_weighted\_graph() = default;

virtual ~directed\_weighted\_graph() = default;

virtual void insertEdge(const std::string &lang) override {

super::fromEdgeLanguage(lang, true);

}

virtual void removeEdge(const std::string &lang) override {

auto nodeLang = "`"s + rlib::splitString(lang, '`')[1];

super::fromNodeLanguage(nodeLang)->neighbors.erase(super::fromEdgeLanguage(lang));

}

};

template <typename data\_t>

class undirected\_weighted\_graph : public directed\_weighted\_graph<data\_t> {

public:

undirected\_weighted\_graph() = default;

virtual ~undirected\_weighted\_graph() = default;

virtual void insertEdge(const std::string &lang) override {

auto parts = rlib::splitString(lang, '`');

std::string reversedLang = rlib::joinString('`', std::array<std::string, 3>{parts[0], parts[2], parts[1]});

directed\_weighted\_graph<data\_t>::insertEdge(lang);

directed\_weighted\_graph<data\_t>::insertEdge(reversedLang);

}

virtual void removeEdge(const std::string &lang) override {

auto parts = rlib::splitString(lang, '`');

std::string reversedLang = rlib::joinString('`', std::array<std::string, 3>{parts[0], parts[2], parts[1]});

directed\_weighted\_graph<data\_t>::removeEdge(lang);

directed\_weighted\_graph<data\_t>::removeEdge(reversedLang);

}

};

template <typename data\_t>

class directed\_unweighted\_graph : public directed\_weighted\_graph<data\_t> {

public:

directed\_unweighted\_graph() = default;

virtual ~directed\_unweighted\_graph() = default;

virtual void insertEdge(const std::string &lang) override {

auto parts = rlib::splitString(lang, '`');

parts[0] = '1';

directed\_weighted\_graph<data\_t>::insertEdge(rlib::joinString('`', parts));

}

};

template <typename data\_t>

class undirected\_unweighted\_graph : public undirected\_weighted\_graph<data\_t> {

public:

undirected\_unweighted\_graph() = default;

virtual ~undirected\_unweighted\_graph() = default;

virtual void insertEdge(const std::string &lang) override {

auto parts = rlib::splitString(lang, '`');

parts[0] = '1';

undirected\_weighted\_graph<data\_t>::insertEdge(rlib::joinString('`', parts));

}

};

}

#endif

//indexed\_list.hpp

#ifndef \_HUST\_INDEXED\_LIST\_HPP

#define \_HUST\_INDEXED\_LIST\_HPP 1

#include <list>

class indexed\_list : public std::list {

public:

};

#endif //\_HUST\_INDEXED\_LIST\_HPP

//main.cc

#include <general\_ui.hpp>

#include <parser.hpp>

reflected\_impl impl;

//GCThread gc;

int main() {

fake\_terminal::go(parser::parse);

}//parser.hpp

#ifndef \_HUST\_\_\_PARSER\_HPP

#define \_HUST\_\_\_PARSER\_HPP 1

#include <reflected\_impl.hpp>

#include <list>

#include <string>

#include <iomanip>

#include <rlib/stdio.hpp>

#include <rlib/terminal.hpp>

using namespace rlib::terminal;

class parser

{

private:

static void help\_msg()

{

std::string msg = R"\_STR\_(

rfaketerm 0.2 HUST\_xxxx special edition

>>> Usage: <Command> [args ...]

>>> Command List:

CommandName [Arguments ...] -> ReturnValue # Instructions

# Commands useful to operate

help -> null # Show this message

exit -> null # exit politely

Select [int i] -> null # Select which graph to use (Select 0 by default, index starts from zero)

List -> null # List how many graph is working currently

QuickTraverse # Print all nodes information to stdout in current graph

# Commands required by Question Book

CreateGraph [string typeStr] -> null # typeStr must be one of: 'directed\_weighted\_graph' 'undirected\_weighted\_graph' 'directed\_unweighted\_graph' 'undirected\_unweighted\_graph'

DestroyGraph -> null

LocateVex [data\_t val] -> Language

GetVex [Language lang] -> Language

PutVex [Language lang] -> null # omit `address` to append a new node, otherwise to edit a existing node.

FirstAdjVex [Language lang] -> Language

NextAdjVex [Language lang1, Language lang2] -> Language

InsertVex [Language lang] -> null

DeleteVex [Language lang] -> null

InsertArc [Language lang] -> null

DeleteArc [Language lang] -> null

DFSTraverse -> null

BFSTraverse -> null

>>> What's Language? How should I use it?

Language includes NodeLanguage and EdgeLanguage.

NodeLanguage is a string language, with which you can describe a node in a graph.

It's a string with format: [value]`[address]

EdgeLanguage is a string language, with which you can describe an edge connected with two valid nodes.

It's a string with format: [weight]`<nodeAddressFrom>`[nodeAddressTo]

In addition, NodeAddress is guaranteed to be valid during the lifetime of the process, unless erased.

Usually, you needn't fill all areas in a "Language". For example:

rfaketerm ~ CreateGraph directed\_unweighted\_graph

rfaketerm ~ Select 0

rfaketerm ~ PutVex 200`

rfaketerm ~ LocateVex 100

100`FFFF04AE

rfaketerm ~ GetVex `FFFF04AE

100`FFFF04AE

rfaketerm ~ PutVex 200`FFFF04AE

rfaketerm ~ PutVex 2333`

rfaketerm ~ PutVex 666`

rfaketerm ~ QuickTraverse

200`FFFF04AE 2333`FFFF04BE 666`FFFF010A

rfaketerm ~ InsertVex `FFFF04AE`FFFF010A

rfaketerm ~ DFSTraverse

...

In order to simplify node address, you can set an "alias to address" while performing "PutVex".

Any given address will be checked if it've been registered as an alias.

For example:

rfaketerm ~ PutVex 200`node1

rfaketerm ~ PutVex 2333`node2

rfaketerm ~ PutVex 666`

rfaketerm ~ QuickTraverse

200`FFFF04AE 2333`FFFF04BE 666`FFFF010A

rfaketerm ~ GetVex `node2

2333`FFFF04BE

rfaketerm ~ GetVex `FFFF04BE

2333`FFFF04BE

)\_STR\_";

rlib::println(msg);

}

public:

static void parse(const std::vector<std::string> &to\_parse)

{

if (to\_parse.empty())

return;

rlib::print(std::boolalpha);

#define AREA\_BEGIN if(to\_parse.begin()->empty()) {}

#define IFCMD(str) else if(\*to\_parse.begin() == str)

#define AREA\_END else

#define WANT\_ARG(n) if(to\_parse.size() != n+1) {throw std::runtime\_error(rlib::format\_string("{} arguments wanted but {} provided.", n, to\_parse.size()-1));}

#define STRING\_ARG(n) to\_parse[n]

#define SIZE\_ARG(n) std::stoul(to\_parse[n])

#define INT\_ARG(n) std::stoi(to\_parse[n])

#define HAVE\_RETURN\_VALUE auto ret =

#define PRINT\_RETURN\_VALUE rlib::println(ret);

AREA\_BEGIN

//\_\_ccgen\_managed\_begin\_\_

//Code generated by ccgen.py below. Do not edit them by hand.

//\_\_ccgen\_debug\_\_: `ret name(args)` is `null Select(size\_t i)`

IFCMD("Select")

{

WANT\_ARG(1)

impl.Select(SIZE\_ARG(1));

}

//\_\_ccgen\_debug\_\_: `ret name(args)` is `null List()`

IFCMD("List")

{

WANT\_ARG(0)

impl.List();

}

//\_\_ccgen\_debug\_\_: `ret name(args)` is `null QuickTraverse()`

IFCMD("QuickTraverse")

{

WANT\_ARG(0)

impl.QuickTraverse();

}

//\_\_ccgen\_debug\_\_: `ret name(args)` is `null CreateGraph(Language typeStr)`

IFCMD("CreateGraph")

{

WANT\_ARG(1)

impl.CreateGraph(STRING\_ARG(1));

}

//\_\_ccgen\_debug\_\_: `ret name(args)` is `null DestroyGraph()`

IFCMD("DestroyGraph")

{

WANT\_ARG(0)

impl.DestroyGraph();

}

//\_\_ccgen\_debug\_\_: `ret name(args)` is `Language LocateVex(data\_t val)`

IFCMD("LocateVex")

{

WANT\_ARG(1)

HAVE\_RETURN\_VALUE

impl.LocateVex(INT\_ARG(1));

PRINT\_RETURN\_VALUE

}

//\_\_ccgen\_debug\_\_: `ret name(args)` is `Language GetVex(Language lang)`

IFCMD("GetVex")

{

WANT\_ARG(1)

HAVE\_RETURN\_VALUE

impl.GetVex(STRING\_ARG(1));

PRINT\_RETURN\_VALUE

}

//\_\_ccgen\_debug\_\_: `ret name(args)` is `null PutVex(Language lang)`

IFCMD("PutVex")

{

WANT\_ARG(1)

impl.PutVex(STRING\_ARG(1));

}

//\_\_ccgen\_debug\_\_: `ret name(args)` is `Language FirstAdjVex(Language lang)`

IFCMD("FirstAdjVex")

{

WANT\_ARG(1)

HAVE\_RETURN\_VALUE

impl.FirstAdjVex(STRING\_ARG(1));

PRINT\_RETURN\_VALUE

}

//\_\_ccgen\_debug\_\_: `ret name(args)` is `Language NextAdjVex(Language lang1, Language lang2)`

IFCMD("NextAdjVex")

{

WANT\_ARG(2)

HAVE\_RETURN\_VALUE

impl.NextAdjVex(STRING\_ARG(1), STRING\_ARG(2));

PRINT\_RETURN\_VALUE

}

//\_\_ccgen\_debug\_\_: `ret name(args)` is `null InsertVex(Language lang)`

IFCMD("InsertVex")

{

WANT\_ARG(1)

impl.InsertVex(STRING\_ARG(1));

}

//\_\_ccgen\_debug\_\_: `ret name(args)` is `null DeleteVex(Language lang)`

IFCMD("DeleteVex")

{

WANT\_ARG(1)

impl.DeleteVex(STRING\_ARG(1));

}

//\_\_ccgen\_debug\_\_: `ret name(args)` is `null InsertArc(Language lang)`

IFCMD("InsertArc")

{

WANT\_ARG(1)

impl.InsertArc(STRING\_ARG(1));

}

//\_\_ccgen\_debug\_\_: `ret name(args)` is `null DeleteArc(Language lang)`

IFCMD("DeleteArc")

{

WANT\_ARG(1)

impl.DeleteArc(STRING\_ARG(1));

}

//\_\_ccgen\_debug\_\_: `ret name(args)` is `null DFSTraverse()`

IFCMD("DFSTraverse")

{

WANT\_ARG(0)

impl.DFSTraverse();

}

//\_\_ccgen\_debug\_\_: `ret name(args)` is `null BFSTraverse()`

IFCMD("BFSTraverse")

{

WANT\_ARG(0)

impl.BFSTraverse();

} IFCMD("exit")

{

rlib::println("bye~");

::std::exit(0);

} IFCMD("help")

{

help\_msg();

}

//impl.debug();

//Code generated by ccgen.py ahead. Do not edit them by hand.

//\_\_ccgen\_managed\_end\_\_

AREA\_END

{

throw std::invalid\_argument("Invalid argument. Try to type `help` to get helped.");

}

}

};

#endif //\_HUST\_\_\_PARSER\_HPP

//README.md

### Compilation note

In CMakeLists.txt, you can uncomment the line 12 `# set(CMAKE\_CXX\_FLAGS "${CMAKE\_CXX\_FLAGS} -DCOMPILE\_NO\_ERASE -DNODES\_PRE\_ALLOC\_MEM=1000000")` to get much higher performance. That is, remove an O(n) factor from dfs/bfs algorithm. However, this will make function `DeleteVex(removeNode)` unavailable, because this will invalidate most node aliases/addresses.

You can use at most `NODES\_PRE\_ALLOC\_MEM` nodes in your test case. By default, that's 1M, which eats you about 333MBytes memory. You can set it to 10M to test my algorithm further, which needs about 3.3GB memory. Don't forget `-DCMAKE\_BUILD\_TYPE=Release`, which gives `-O3` to let your test boost 4 times!

I provide pre-compiled binary in both version, for both linux and windows.

### My tests

- normal version

|nodes|time|memory|

|:---:|:---:|:---:|

|1K|0.15s|tiny|

|10K|27s|tiny|

- high performance version

|nodes|time|memory|

|:---:|:---:|:---:|

|1K|0.04s|tiny|

|10K|0.40s|tiny|

|100K|4.3s|55MB|

|1M|44s|600MB|

|10M|570s|6GB|

//reflected\_impl.hpp

#ifndef HUST\_\_\_REFLECTED\_IMPL\_HPP\_

#define HUST\_\_\_REFLECTED\_IMPL\_HPP\_

/\*

\* You should NEVER use this code in ANY consequence,

\* as these code is just to make hust happy.

\*/

#include <utility>

#include <functional>

#include <algorithm>

#include <vector>

#include "graph.hpp"

#include <rlib/stdio.hpp>

//class reflected\_impl {

//public:

// using data\_t = int;

// using BooleanAsserter = std::function<bool(const data\_t &)>;

// using OperationVisiter = std::function<void(const data\_t &)>;

//

// void InitList() const {}

// void DestroyList() {container.clear();}

// void ClearList() {container.clear();}

// bool ListEmpty() const {return container.size() == 0;}

// size\_t ListLength() const {return container.size();}

// data\_t GetElem(size\_t \_\_\_IndexPlusOne) {

// auto index = \_\_\_IndexPlusOne - 1;

// auto iter = container.begin();

// for(size\_t cter = 0; cter < index; ++cter) {

// ++iter;

// }

// return std::move(\*iter);

// }

// size\_t \_LocateElem(const BooleanAsserter &comparer) {

// auto iter = std::find\_if(container.begin(), container.end(), comparer);

// if(iter == container.end()) {

// return 0;

// }

// return LabUtils::distance(container.begin(), iter);

// }

// size\_t LocateElem(data\_t val) {

// auto comparer = BooleanAsserter([v=val](const data\_t &dat){

// return dat == v;

// });

// return \_LocateElem(comparer);

// }

// data\_t PriorElem(data\_t tofind) {

// auto pos = std::find(container.begin(), container.end(), tofind);

// if(pos == container.end() || pos == container.begin()) {

// throw std::runtime\_error("ElemError: You told me that it's undefined, so I do it.");

// }

// return \*(--pos);

// }

// data\_t NextElem(data\_t tofind) {

// auto pos = std::find(container.begin(), container.end(), tofind);

// if(pos == container.end() || pos == --container.end()) {

// throw std::runtime\_error("ElemError: You told me that it's undefined, so I do it.");

// }

// return \*(++pos);

// }

// void ListInsert(size\_t \_\_\_IndexPlusOne, data\_t elem) {

// auto index = \_\_\_IndexPlusOne - 1;

// auto iter = LabUtils::advance(container.begin(), index);

// container.insert(iter, elem);

// }

// data\_t ListDelete(size\_t \_\_\_IndexPlusOne) {

// auto index = \_\_\_IndexPlusOne - 1;

// auto iter = LabUtils::advance(container.begin(), index);

// auto to\_return = \*iter;

// container.erase(iter);

// return std::move(to\_return);

// }

// void \_ListTraverse(const OperationVisiter &visiter) {

// std::for\_each(container.begin(), container.end(), visiter);

// }

// void ListTraverse() {

// \_ListTraverse(OperationVisiter([](const auto &val){rlib::io::print(val, " ");}));

// rlib::io::println("");

// }

//

// void debug() {

// rlib::io::println\_iter(container);

// rlib::io::println(container.size());

// }

//private:

// Lab::list<data\_t> container;

//};

//using hust\_xxxx::unordered\_btree;

//class reflected\_impl {

//public:

// using data\_t = int;

// using dataref\_t = const data\_t &;

// using nlang = std::string;

// using nlangref = const nlang &;

// reflected\_impl() : containers(1), current(containers.begin()) {}

//

// void Select(size\_t i) {current = containers.begin() + i;}

// void List() {rlib::printfln("You have {} btree now, selecting {}.", containers.size(), current - containers.begin());}

//

// void InitBiTree() {}

// void DestroyBiTree() {containers.erase(current); current = containers.begin();}

// void CreateBiTree() {containers.push\_back(unordered\_btree<data\_t>());}

// void ClearBiTree() {current->clear();}

// bool BiTreeEmpty() {return current->empty();}

// size\_t BiTreeDepth() {return current->depth();}

// nlang Root() {return current->\_root();}

// data\_t Value(nlangref n) {return current->get(n);}

// void Assign(nlangref n, dataref\_t val) {return current->set(n, val);}

// nlang Parent(nlangref n) {return current->parent(n);}

// nlang LeftChild(nlangref n) {return current->lchild(n);}

// nlang RightChild(nlangref n) {return current->rchild(n);}

// nlang LeftSibling(nlangref n) {return current->lchild(current->parent(n));}

// nlang RightSibling(nlangref n) {return current->rchild(current->parent(n));}

// void InsertChild(nlangref n, size\_t toInsert, size\_t LR) {return current->merge(containers[toInsert], n, LR==1);}

// void DeleteChild(nlangref n, size\_t LR) {return current->drop(n, LR==1);}

// void PreOrderTraverse() {current->for\_each(unordered\_btree<data\_t>::printer, hust\_xxxx::foreach\_rule::MIDDLE\_LEFT\_RIGHT);}

// void InOrderTraverse() {current->for\_each(unordered\_btree<data\_t>::printer, hust\_xxxx::foreach\_rule::LEFT\_MIDDLE\_RIGHT);}

// void PostOrderTraverse() {current->for\_each(unordered\_btree<data\_t>::printer, hust\_xxxx::foreach\_rule::LEFT\_RIGHT\_MIDDLE);}

// void LevelOrderTraverse() {current->level\_for\_each(unordered\_btree<data\_t>::printer);}

//

//private:

// std::vector<unordered\_btree<data\_t>> containers;

// decltype(containers.begin()) current;

//};

using namespace hust\_xxxx;

class reflected\_impl {

public:

using data\_t = int;

using dataref\_t = const data\_t &;

using lang\_t = std::string;

using langref\_t = const lang\_t &;

reflected\_impl() : current(containers.begin()) {}

// \_\_py\_ccgen\_begin\_\_

void Select(size\_t i) {current = containers.begin() + i;}

void List() {rlib::printfln("You have {} basic\_graph now, selecting {}.", containers.size(), current - containers.begin());}

void QuickTraverse() {(\*current)->simple\_foreach((\*current)->printer);rlib::println();}

void CreateGraph(langref\_t typeStr) {containers.push\_back(newFromTypeStr(typeStr));}

void DestroyGraph() {containers.erase(current); current = containers.begin();}

lang\_t LocateVex(dataref\_t val) {return (\*current)->findNode(val);}

lang\_t GetVex(langref\_t lang) {return (\*current)->getNodeValue(lang);}

void PutVex(langref\_t lang) {(\*current)->setNodeValue(lang);}

lang\_t FirstAdjVex(langref\_t lang) {return (\*current)->findFirstNearNode(lang);}

lang\_t NextAdjVex(langref\_t lang1, langref\_t lang2) {return (\*current)->findNextNearNode(lang1, lang2);}

void InsertVex(langref\_t lang) {this->PutVex(lang);}

void DeleteVex(langref\_t lang) {(\*current)->removeNode(lang);}

void InsertArc(langref\_t lang) {(\*current)->insertEdge(lang);}

void DeleteArc(langref\_t lang) {(\*current)->removeEdge(lang);}

void DFSTraverse() {(\*current)->dfs((\*current)->printer);rlib::println();}

void BFSTraverse() {(\*current)->bfs((\*current)->printer);rlib::println();}

// \_\_py\_ccgen\_end\_\_

private:

basic\_graph<data\_t> \*newFromTypeStr(const std::string &typeStr) {

if(typeStr == "directed\_weighted\_graph")

return new directed\_weighted\_graph<data\_t>();

if(typeStr == "undirected\_weighted\_graph")

return new undirected\_weighted\_graph<data\_t>();

if(typeStr == "undirected\_unweighted\_graph")

return new undirected\_unweighted\_graph<data\_t>();

if(typeStr == "directed\_unweighted\_graph")

return new directed\_unweighted\_graph<data\_t>();

throw std::invalid\_argument("invalid typestr");

}

std::vector<basic\_graph<data\_t> \*> containers;

decltype(containers.begin()) current;

};

extern reflected\_impl impl;

#endif

//rlib

cat: rlib: 是一个目录

//testgen.py

#!/usr/bin/env python3

import sys

import random

if len(sys.argv) != 2:

print("Usage: ./this.py <test size>")

exit(1)

test\_size = int(sys.argv[1])

prob\_skip\_a\_node = 0.8

edge\_per\_node\_begin = 8

edge\_per\_node\_end = 64

def iToVarName(i):

return 'n'+str(i)

print('''#!./exp4

# `./exp4 input` not implemented, so you cannot run `./input` directly.

CreateGraph directed\_unweighted\_graph

Select 0

''')

for i in range(test\_size):

print('PutVex {}`n{}'.format(i, i))

for i in range(test\_size):

if i != 0 and random.random() < 0.5: # n0 must have edges

continue

for \_ in range(random.randint(edge\_per\_node\_begin, edge\_per\_node\_end)):

# May have multiedge / selfring

j = random.randint(0, test\_size - 1)

print('InsertArc `n{}`n{}'.format(i, j))

print('''

DFSTraverse

BFSTraverse

''')

4.3.2 实验中多次使用的rlib库在此时的最终版本

测试部分被剔除。非关键部分被剔除。所有被使用过的rlib库内容在此均有据可查。

//./traits.hpp

#ifndef RLIB\_TRAITS\_HPP

#define RLIB\_TRAITS\_HPP

#include <type\_traits>

namespace rlib {

template<typename T>

struct is\_callable\_helper {

private:

typedef char(&yes)[1];

typedef char(&no)[2];

struct Fallback { void operator()(); };

struct Derived : T, Fallback { };

template<typename U, U> struct Check;

template<typename>

static yes test(...);

template<typename C>

static no test(Check<void (Fallback::\*)(), &C::operator()>\*);

public:

static const bool value = sizeof(test<Derived>(0)) == sizeof(yes);

};

template<typename T>

struct is\_callable

: std::conditional<

std::is\_class<T>::value,

is\_callable\_helper<T>,

std::is\_function<T>>::type

{};

}

#endif//./Makefile

CXX ?= g++

CC ?= gcc

AR ?= ar

CXXFLAGS = -O3

CFLAGS =

ARFLAGS = rcs

def: compile\_library

compile\_library:

$(CXX) $(CXXFLAGS) -c libr.cc -o libr.o

$(AR) $(ARFLAGS) libr.a libr.o

install\_header:

[ ! -d /usr/include/rlib ] || rm -rf /usr/include/rlib

cp -r . /usr/include/rlib

rm -rf /usr/include/rlib/test /usr/include/rlib/.git

install\_library: compile\_library

cp libr.a /usr/lib/

install: install\_header install\_library

uninstall:

rm -rf /usr/include/rlib

rm /usr/lib/libr.a

clean:

rm \*.o \*.a

//./LICENSE

MIT License

Copyright (c) 2018 Recolic Keghart

Permission is hereby granted, free of charge, to any person obtaining a copy

of this software and associated documentation files (the "Software"), to deal

in the Software without restriction, including without limitation the rights

to use, copy, modify, merge, publish, distribute, sublicense, and/or sell

copies of the Software, and to permit persons to whom the Software is

furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all

copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR

IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,

FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE

AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER

LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,

OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE

SOFTWARE.

//./sys/fdset.hpp

#ifndef R\_FDSET\_HPP

#define R\_FDSET\_HPP

#include <unistd.h>

#include <sys/types.h>

namespace rlib{

class FileDescriptorSet

{

public:

using fd=int;

FileDescriptorSet() : m\_size(0), maxFileDescriptor(NULL) {FD\_ZERO(&m\_fds\_data);}

void push(fd FileDescriptor) {FD\_SET(FileDescriptor, &m\_fds\_data); ++m\_size; maxFileDescriptor = (maxFileDescriptor > FileDescriptor ? maxFileDescriptor : FileDescriptor);}

void pop(fd FileDescriptor) {FD\_CLR(FileDescriptor, &m\_fds\_data); --m\_size;} //It will break maxFileDescriptor.(for performance reason).

void clear() {FD\_ZERO(&m\_fds\_data); m\_size = 0;maxFileDescriptor = 0;}

bool check(fd FileDescriptor) {return FD\_ISSET(FileDescriptor, &m\_fds\_data);}

size\_t size() const {return m\_size;}

int getMaxFileDescriptor() const {return maxFileDescriptor;}

fd\_set \*getptr() {return &m\_fds\_data;}

private:

fd\_set m\_fds\_data;

size\_t m\_size;

int maxFileDescriptor;

};

}

#endif

//./sys/os.hpp

#ifndef R\_OS\_HPP

#define R\_OS\_HPP

#ifndef \_\_OS\_ID\_\_

#if defined(\_Windows) || defined(\_\_WIN32\_\_) || defined(\_WIN64) || defined(WIN32)

# define \_\_OS\_ID\_\_ WINDOWS

#elif defined(\_\_linux\_\_) || defined(\_\_linux)

# define \_\_OS\_ID\_\_ LINUX

#elif defined(\_\_APPLE\_\_)

# include "TargetConditionals.h"

# if TARGET\_IPHONE\_SIMULATOR

# define \_\_OS\_ID\_\_ IOS

# elif TARGET\_OS\_IPHONE

# define \_\_OS\_ID\_\_ IOS

# elif TARGET\_OS\_MAC

# define \_\_OS\_ID\_\_ MACOS

# else

# define \_\_OS\_ID\_\_ UNKNOWN\_UNIX

# endif

#elif defined(\_\_ANDROID\_\_)

# define \_\_OS\_ID\_\_ ANDROID

#elif defined(\_\_unix\_\_) || defined(\_\_unix)

# define \_\_OS\_ID\_\_ UNKNOWN\_UNIX

#else

# define \_\_OS\_ID\_\_ UNKNOWN

#endif

#endif

#include "compiler\_detector"

// Define \_\_COMPILER\_ID\_\_ and \_\_COMPILER\_VER\_\_

#if \_\_cplusplus >= 201103L

namespace rlib {

class OSInfo

{

public:

enum class os\_t {UNKNOWN, WINDOWS, LINUX, MACOS, BSD, IOS, ANDROID, UNKNOWN\_UNIX};

enum class compiler\_t {UNKNOWN, GCC, CLANG, MSVC, INTELC, BORLAND, IARC, SOLARIS, ZAPCC}; //Compiler which not supports cxx1x yet is not listed here. 201708.

static constexpr os\_t os =

#if defined(\_\_OS\_ID\_\_)

os\_t::\_\_OS\_ID\_\_;

#else

os\_t::UNKNOWN;

#endif

static constexpr compiler\_t compiler =

#if defined(\_\_COMPILER\_ID\_\_)

compiler\_t::\_\_COMPILER\_ID\_\_;

#else

compiler\_t::UNKNOWN;

#endif

static constexpr auto compiler\_version =

#if defined(\_\_COMPILER\_VER\_\_)

\_\_COMPILER\_VER\_\_;

#else

0;

#endif

};

}

#endif

#endif

//./sys/cc\_codegen.py

#!/bin/env python3

def genDefList(idarr):

s = '#if'

cter = 1

for i in idarr:

s += ' defined(' + i + ')'

if cter != len(idarr):

s += ' ||'

cter += 1

return s

print('// Generated by cc\_codegen.py. Do not edit it by hand.')

with open("cc\_list") as fd:

osarr=fd.read().split('\n')

for i in osarr:

if i == '':

continue

iarr=i.split(' ')

if len(iarr) < 2:

continue

print('#ifndef \_\_COMPILER\_ID\_\_')

print(genDefList(iarr[:-1:]))

print('#define \_\_COMPILER\_ID\_\_', iarr[-1])

print('#endif')

print('#endif')

print('')

print('#ifndef \_\_COMPILER\_ID\_\_')

print('#define \_\_COMPILER\_ID\_\_ UNKNOWN')

print('#endif')

//./sys/rwlock.hpp

#ifndef R\_SWLOCK\_HPP

#define R\_SWLOCK\_HPP

#include <pthread.h>

namespace rlib {

class RWLock

{

public:

RWLock() : isFree(true) {pthread\_rwlock\_init(&m\_lock, NULL);}

~RWLock() {pthread\_rwlock\_destroy(&m\_lock);}

void acquireShared() {pthread\_rwlock\_rdlock(&m\_lock);isFree = false;}

void acquireExclusive() {pthread\_rwlock\_wrlock(&m\_lock);isFree = false;}

void release() {pthread\_rwlock\_unlock(&m\_lock);isFree = true;}

// bool tryAcquireShared() {return pthread\_rwlock\_tryrdlock(&m\_lock) == 0;}

// bool tryAcquireExclusive() {return pthread\_rwlock\_trywrlock(&m\_lock) == 0;}

private:

pthread\_rwlock\_t m\_lock;

bool isFree;

};

}

#endif//./sys/compiler\_detector

// Generated by cc\_codegen.py. Do not edit it by hand.

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_ACC\_)

#define \_\_COMPILER\_ID\_\_ ACC

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_CMB\_\_)

#define \_\_COMPILER\_ID\_\_ ALTIUM\_MICROBLAZE

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_CHC\_\_)

#define \_\_COMPILER\_ID\_\_ ALTIUM\_HARDWARE

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_ACK\_\_)

#define \_\_COMPILER\_ID\_\_ AMSTERDAM

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_CC\_ARM)

#define \_\_COMPILER\_ID\_\_ ARMCC

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(AZTEC\_C) || defined(\_\_AZTEC\_C\_\_)

#define \_\_COMPILER\_ID\_\_ AZTEC

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_BORLANDC\_\_) || defined(\_\_CODEGEARC\_\_)

#define \_\_COMPILER\_ID\_\_ BORLAND

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_CC65\_\_)

#define \_\_COMPILER\_ID\_\_ CC65

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_clang\_\_)

#define \_\_COMPILER\_ID\_\_ CLANG

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_COMO\_\_)

#define \_\_COMPILER\_ID\_\_ COMEAU

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_DECC) || defined(\_\_DECCXX)

#define \_\_COMPILER\_ID\_\_ COMPAQ

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_convexc\_\_)

#define \_\_COMPILER\_ID\_\_ CONVEX

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_COMPCERT\_\_)

#define \_\_COMPILER\_ID\_\_ COMPCERT

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_COVERITY\_\_)

#define \_\_COMPILER\_ID\_\_ COVERITY

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_CRAYC)

#define \_\_COMPILER\_ID\_\_ CRAY

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_DCC\_\_)

#define \_\_COMPILER\_ID\_\_ DIAB

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_DICE)

#define \_\_COMPILER\_ID\_\_ DICE

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_DMC\_\_)

#define \_\_COMPILER\_ID\_\_ DIGITAL\_MARS

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_SYSC\_\_)

#define \_\_COMPILER\_ID\_\_ DIGNUS

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_DJGPP\_\_)

#define \_\_COMPILER\_ID\_\_ DJGPP

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_EDG\_\_)

#define \_\_COMPILER\_ID\_\_ EDG

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_PATHCC\_\_)

#define \_\_COMPILER\_ID\_\_ EKOPATH

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_FCC\_VERSION)

#define \_\_COMPILER\_ID\_\_ FUJITSU

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_GNUC\_\_)

#define \_\_COMPILER\_ID\_\_ GCC

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_ghs\_\_)

#define \_\_COMPILER\_ID\_\_ GREENHILL

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_HP\_cc)

#define \_\_COMPILER\_ID\_\_ HPC

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_HP\_aCC)

#define \_\_COMPILER\_ID\_\_ HPACXX

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_IAR\_SYSTEMS\_ICC\_\_)

#define \_\_COMPILER\_ID\_\_ IARC

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_IBMCPP\_\_) || defined(\_\_IBMC\_\_)

#define \_\_COMPILER\_ID\_\_ IBMC

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_IMAGECRAFT\_\_)

#define \_\_COMPILER\_ID\_\_ IMAGECRAFT

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_INTEL\_COMPILER) || defined(\_\_ICL)

#define \_\_COMPILER\_ID\_\_ INTELC

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_KCC)

#define \_\_COMPILER\_ID\_\_ KAICXX

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_CA\_\_) || defined(\_\_KEIL\_\_)

#define \_\_COMPILER\_ID\_\_ KEIL\_CARM

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_C166\_\_)

#define \_\_COMPILER\_ID\_\_ KEIL\_C166

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_C51\_\_) || defined(\_\_CX51\_\_)

#define \_\_COMPILER\_ID\_\_ KEIL\_C51

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_LCC\_\_)

#define \_\_COMPILER\_ID\_\_ LCC

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_llvm\_\_)

#define \_\_COMPILER\_ID\_\_ LLVM

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_MWERKS\_\_) || defined(\_\_CWCC\_\_)

#define \_\_COMPILER\_ID\_\_ METROWERKS

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_MSC\_VER)

#define \_\_COMPILER\_ID\_\_ MSVC

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_MRI)

#define \_\_COMPILER\_ID\_\_ MICROTEC

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_NDPC\_\_) || defined(\_\_NDPX\_\_)

#define \_\_COMPILER\_ID\_\_ MICROWAY

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_sgi) || defined(sgi)

#define \_\_COMPILER\_ID\_\_ MIPSPRO

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(MIRACLE)

#define \_\_COMPILER\_ID\_\_ MIRACLE

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_MRC\_\_) || defined(MPW\_C) || defined(MPW\_CPLUS)

#define \_\_COMPILER\_ID\_\_ MPW

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_CC\_NORCROFT)

#define \_\_COMPILER\_ID\_\_ NORCROFT

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_NWCC\_\_)

#define \_\_COMPILER\_ID\_\_ NWCC

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_OPEN64\_\_) || defined(\_\_OPENCC\_\_)

#define \_\_COMPILER\_ID\_\_ OPEN64

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(ORA\_PROC)

#define \_\_COMPILER\_ID\_\_ ORACLE\_PROC

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_SUNPRO\_C) || defined(\_\_SUNPRO\_CC)

#define \_\_COMPILER\_ID\_\_ SOLARIS

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_PACIFIC\_\_)

#define \_\_COMPILER\_ID\_\_ PACIFIC

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_PACC\_VER)

#define \_\_COMPILER\_ID\_\_ PLAM

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_POCC\_\_)

#define \_\_COMPILER\_ID\_\_ PELLES

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_PGI)

#define \_\_COMPILER\_ID\_\_ PORTLAND

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_RENESAS\_\_) || defined(\_\_HITACHI\_\_)

#define \_\_COMPILER\_ID\_\_ RENESAS

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(SASC) || defined(\_\_SASC) || defined(\_\_SASC\_\_)

#define \_\_COMPILER\_ID\_\_ SASC

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_SCO\_DS)

#define \_\_COMPILER\_ID\_\_ SCO\_OPENSERVER

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(SDCC)

#define \_\_COMPILER\_ID\_\_ SDCC

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_SNC\_\_)

#define \_\_COMPILER\_ID\_\_ SN

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_VOSC\_\_)

#define \_\_COMPILER\_ID\_\_ STRATUS\_VOS

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_SC\_\_)

#define \_\_COMPILER\_ID\_\_ SYMANTEC

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_TenDRA\_\_)

#define \_\_COMPILER\_ID\_\_ TENDRA

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_TI\_COMPILER\_VERSION\_\_) || defined(\_TMS320C6X)

#define \_\_COMPILER\_ID\_\_ TEXAS

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(THINKC3) || defined(THINKC4)

#define \_\_COMPILER\_ID\_\_ THINK

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_TINYC\_\_)

#define \_\_COMPILER\_ID\_\_ TINYC

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_TURBOC\_\_)

#define \_\_COMPILER\_ID\_\_ TURBOC

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_UCC)

#define \_\_COMPILER\_ID\_\_ UCC

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_USLC\_\_)

#define \_\_COMPILER\_ID\_\_ USLC

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_VBCC\_\_)

#define \_\_COMPILER\_ID\_\_ VBCC

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_WATCOMC\_\_)

#define \_\_COMPILER\_ID\_\_ WATCOM

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#if defined(\_\_ZTC\_\_)

#define \_\_COMPILER\_ID\_\_ ZORTECH

#endif

#endif

#ifndef \_\_COMPILER\_ID\_\_

#define \_\_COMPILER\_ID\_\_ UNKNOWN

#endif

//./sys/cc\_list

\_ACC\_ ACC

\_\_CMB\_\_ ALTIUM\_MICROBLAZE

\_\_CHC\_\_ ALTIUM\_HARDWARE

\_\_ACK\_\_ AMSTERDAM

\_\_CC\_ARM ARMCC

AZTEC\_C \_\_AZTEC\_C\_\_ AZTEC

\_\_BORLANDC\_\_ \_\_CODEGEARC\_\_ BORLAND

\_\_CC65\_\_ CC65

\_\_clang\_\_ CLANG

\_\_COMO\_\_ COMEAU

\_\_DECC \_\_DECCXX COMPAQ

\_\_convexc\_\_ CONVEX

\_\_COMPCERT\_\_ COMPCERT

\_\_COVERITY\_\_ COVERITY

\_CRAYC CRAY

\_\_DCC\_\_ DIAB

\_DICE DICE

\_\_DMC\_\_ DIGITAL\_MARS

\_\_SYSC\_\_ DIGNUS

\_\_DJGPP\_\_ DJGPP

\_\_EDG\_\_ EDG

\_\_PATHCC\_\_ EKOPATH

\_\_FCC\_VERSION FUJITSU

\_\_GNUC\_\_ GCC

\_\_ghs\_\_ GREENHILL

\_\_HP\_cc HPC

\_\_HP\_aCC HPACXX

\_\_IAR\_SYSTEMS\_ICC\_\_ IARC

\_\_IBMCPP\_\_ \_\_IBMC\_\_ IBMC

\_\_IMAGECRAFT\_\_ IMAGECRAFT

\_\_INTEL\_COMPILER \_\_ICL INTELC

\_\_KCC KAICXX

\_\_CA\_\_ \_\_KEIL\_\_ KEIL\_CARM

\_\_C166\_\_ KEIL\_C166

\_\_C51\_\_ \_\_CX51\_\_ KEIL\_C51

\_\_LCC\_\_ LCC

\_\_llvm\_\_ LLVM

\_\_MWERKS\_\_ \_\_CWCC\_\_ METROWERKS

\_MSC\_VER MSVC

\_MRI MICROTEC

\_\_NDPC\_\_ \_\_NDPX\_\_ MICROWAY

\_\_sgi sgi MIPSPRO

MIRACLE MIRACLE

\_\_MRC\_\_ MPW\_C MPW\_CPLUS MPW

\_\_CC\_NORCROFT NORCROFT

\_\_NWCC\_\_ NWCC

\_\_OPEN64\_\_ \_\_OPENCC\_\_ OPEN64

ORA\_PROC ORACLE\_PROC

\_\_SUNPRO\_C \_\_SUNPRO\_CC SOLARIS

\_\_PACIFIC\_\_ PACIFIC

\_PACC\_VER PLAM

\_\_POCC\_\_ PELLES

\_\_PGI PORTLAND

\_\_RENESAS\_\_ \_\_HITACHI\_\_ RENESAS

SASC \_\_SASC \_\_SASC\_\_ SASC

\_SCO\_DS SCO\_OPENSERVER

SDCC SDCC

\_\_SNC\_\_ SN

\_\_VOSC\_\_ STRATUS\_VOS

\_\_SC\_\_ SYMANTEC

\_\_TenDRA\_\_ TENDRA

\_\_TI\_COMPILER\_VERSION\_\_ \_TMS320C6X TEXAS

THINKC3 THINKC4 THINK

\_\_TINYC\_\_ TINYC

\_\_TURBOC\_\_ TURBOC

\_UCC UCC

\_\_USLC\_\_ USLC

\_\_VBCC\_\_ VBCC

\_\_WATCOMC\_\_ WATCOM

\_\_ZTC\_\_ ZORTECH//./sys/sio.hpp

#ifndef R\_SIO\_HPP

#define R\_SIO\_HPP

#include <cerrno>

#include <cstdlib>

#include <unistd.h>

#include <string>

#include <stdexcept>

#ifndef WIN32

#include <sys/socket.h>

//POSIX Version

namespace rlib {

class fdIO

{

public:

static ssize\_t readn(int fd, void \*vptr, size\_t n) noexcept //Return -1 on error, read bytes on success, blocks until nbytes done.

{

size\_t nleft;

ssize\_t nread;

char \*ptr;

ptr = (char \*)vptr;

nleft = n;

while (nleft > 0) {

if ( (nread = read(fd, ptr, nleft)) < 0) {

if (errno == EINTR)

nread = 0; /\* and call read() again \*/

else

return (-1);

} else if (nread == 0)

return (-1); /\* EOF \*/

nleft -= nread;

ptr += nread;

}

return (n); /\* return success \*/

}

static ssize\_t writen(int fd, const void \*vptr, size\_t n) noexcept //Return -1 on error, read bytes on success, blocks until nbytes done.

{

size\_t nleft;

ssize\_t nwritten;

const char \*ptr;

ptr = (const char \*)vptr;

nleft = n;

while (nleft > 0) {

if ( (nwritten = write(fd, ptr, nleft)) <= 0) {

if (nwritten < 0 && errno == EINTR)

nwritten = 0; /\* and call write() again \*/

else

return (-1); /\* error \*/

}

nleft -= nwritten;

ptr += nwritten;

}

return (n);

}

static ssize\_t readall(int fd, void \*\*pvptr, size\_t initSize) noexcept //Return -1 on error, read bytes on success. pvptr must be a malloc/calloced buffer, I'll malloc one if \*pvptr is NULL.

{

size\_t current = initSize ? initSize : 1024;

void \*vptr = \*pvptr;

if(vptr == NULL)

vptr = malloc(current);

void \*currvptr = vptr;

{

ssize\_t ret = read(fd, currvptr, current / 2);

if(ret == -1) return -1;

if(ret < current / 2)

{

\*pvptr = vptr;

return ret;

}

currvptr = (char \*)vptr + current / 2;

}

while(true)

{

ssize\_t ret = read(fd, currvptr, current / 2);

if(ret == -1) return -1;

if(ret < current)

{

\*pvptr = vptr;

return ret + current / 2;

}

current \*= 2;

void \*vptrBackup = vptr;

if((vptr = realloc(vptr, current)) == NULL) {

free(vptrBackup);

errno = EMSGSIZE;

return -1;

}

currvptr = (char \*)vptr + current / 2;

}

}

static void readn\_ex(int fd, void \*vptr, size\_t n) //never return error.

{

auto ret = readn(fd, vptr, n);

if(ret == -1) throw std::runtime\_error("readn failed.");

}

static void writen\_ex(int fd, const void \*vptr, size\_t n)

{

auto ret = writen(fd, vptr, n);

if(ret == -1) throw std::runtime\_error("writen failed.");

}

static ssize\_t readall\_ex(int fd, void \*\*pvptr, size\_t initSize) //never return -1

{

auto ret = readall(fd, pvptr, initSize);

if(ret == -1) throw std::runtime\_error("readall failed.");

return ret;

}

};

class sockIO

{

public:

static ssize\_t recvn(int fd, void \*vptr, size\_t n, int flags) noexcept //Return -1 on error, read bytes on success, blocks until nbytes done.

{

size\_t nleft;

ssize\_t nread;

char \*ptr;

ptr = (char \*)vptr;

nleft = n;

while (nleft > 0) {

if ( (nread = recv(fd, ptr, nleft, flags)) < 0) {

if (errno == EINTR)

nread = 0; /\* and call read() again \*/

else

return (-1);

} else if (nread == 0)

return -1; /\* EOF \*/

nleft -= nread;

ptr += nread;

}

return (n); /\* return success \*/

}

static ssize\_t sendn(int fd, const void \*vptr, size\_t n, int flags) noexcept //Return -1 on error, read bytes on success, blocks until nbytes done.

{

size\_t nleft;

ssize\_t nwritten;

const char \*ptr;

ptr = (const char \*)vptr;

nleft = n;

while (nleft > 0) {

if ( (nwritten = send(fd, ptr, nleft, flags)) <= 0) {

if (nwritten < 0 && errno == EINTR)

nwritten = 0; /\* and call write() again \*/

else

return (-1); /\* error \*/

}

nleft -= nwritten;

ptr += nwritten;

}

return (n);

}

static ssize\_t recvall(int fd, void \*\*pvptr, size\_t initSize, int flags) noexcept //Return -1 on error, read bytes on success. pvptr must be a malloc/calloced buffer, I'll malloc one if \*pvptr is NULL.

{

size\_t current = initSize ? initSize : 1024;

void \*vptr = \*pvptr;

if(vptr == NULL)

vptr = malloc(current);

void \*currvptr = vptr;

{

ssize\_t ret = recv(fd, currvptr, current / 2, flags);

if(ret == -1) return -1;

if(ret < current / 2)

{

\*pvptr = vptr;

return ret;

}

currvptr = (char \*)vptr + current / 2;

}

while(true)

{

ssize\_t ret = recv(fd, currvptr, current / 2, flags);

if(ret == -1) return -1;

if(ret < current)

{

\*pvptr = vptr;

return ret + current / 2;

}

current \*= 2;

void \*vptrBackup = vptr;

if((vptr = realloc(vptr, current)) == NULL) {

free(vptrBackup);

errno = EMSGSIZE;

return -1;

}

currvptr = (char \*)vptr + current / 2;

}

}

static void recvn\_ex(int fd, void \*vptr, size\_t n, int flags) //return read bytes.

{

auto ret = recvn(fd, vptr, n, flags);

if(ret == -1) throw std::runtime\_error("recvn failed.");

}

static ssize\_t sendn\_ex(int fd, const void \*vptr, size\_t n, int flags)

{

auto ret = sendn(fd, vptr, n, flags);

if(ret == -1) throw std::runtime\_error("sendn failed.");

return ret;

}

static ssize\_t recvall\_ex(int fd, void \*\*pvptr, size\_t initSize, int flags) //never return -1

{

auto ret = recvall(fd, pvptr, initSize, flags);

if(ret == -1) throw std::runtime\_error("recvall failed.");

return ret;

}

};

}

#else

#include <winsock2.h>

//WINsock version

namespace rlib {

class sockIO

{

private:

static int WSASafeGetLastError()

{

int i;

WSASetLastError(i = WSAGetLastError());

return i;

}

public:

static ssize\_t recvn(SOCKET fd, char \*vptr, size\_t n, int flags) noexcept //Return -1 on error, read bytes on success, blocks until nbytes done.

{

size\_t nleft;

ssize\_t nread;

char \*ptr;

ptr = (char \*)vptr;

nleft = n;

while (nleft > 0) {

if ( (nread = recv(fd, ptr, nleft, flags)) == SOCKET\_ERROR) {

if (WSASafeGetLastError() == WSAEINTR)

nread = 0; /\* and call read() again \*/

else

return (-1);

} else if (nread == 0)

return (-1); /\* EOF \*/

nleft -= nread;

ptr += nread;

}

return (n); /\* return >= 0 \*/

}

static ssize\_t sendn(SOCKET fd, const char \*vptr, size\_t n, int flags) noexcept //Return -1 on error, read bytes on success, blocks until nbytes done.

{

size\_t nleft;

ssize\_t nwritten;

const char \*ptr;

ptr = (const char \*)vptr;

nleft = n;

while (nleft > 0) {

if ( (nwritten = send(fd, ptr, nleft, flags)) <= 0) {

if (nwritten == SOCKET\_ERROR && WSASafeGetLastError() == WSAEINTR)

nwritten = 0; /\* and call write() again \*/

else

return (-1); /\* error \*/

}

nleft -= nwritten;

ptr += nwritten;

}

return (n);

}

static ssize\_t recvall(SOCKET fd, void \*\*pvptr, size\_t initSize, int flags) noexcept //Return -1 on error, read bytes on success. pvptr must be a malloc/calloced buffer, I'll malloc one if \*pvptr is NULL.

{

size\_t current = initSize ? initSize : 1024;

void \*vptr = \*pvptr;

if(vptr == NULL)

vptr = malloc(current);

void \*currvptr = vptr;

{

\_retry\_1:

ssize\_t ret = recv(fd, (char \*)currvptr, current / 2, flags);

if(ret == SOCKET\_ERROR) {

if(WSASafeGetLastError() == WSAEINTR)

goto \_retry\_1;

return SOCKET\_ERROR;

}

if(ret < current / 2)

{

\*pvptr = vptr;

return ret;

}

currvptr = (char \*)vptr + current / 2;

}

while(true)

{

ssize\_t ret = recv(fd, (char \*)currvptr, current / 2, flags);

if(ret == SOCKET\_ERROR) {

if(WSASafeGetLastError() == WSAEINTR)

continue; //retry

return SOCKET\_ERROR;

}

if(ret < current)

{

\*pvptr = vptr;

return ret + current / 2;

}

current \*= 2;

void \*vptrBackup = vptr;

if((vptr = realloc(vptr, current)) == NULL) {

free(vptrBackup);

WSASetLastError(WSAEMSGSIZE);

return SOCKET\_ERROR;

}

currvptr = (char \*)vptr + current / 2;

}

}

static void recvn\_ex(SOCKET fd, char \*vptr, size\_t n, int flags) //never return error.

{

auto ret = recvn(fd, vptr, n, flags);

if(ret == -1) throw std::runtime\_error("recvn failed.");

}

static ssize\_t sendn\_ex(SOCKET fd, const char \*vptr, size\_t n, int flags)

{

auto ret = sendn(fd, vptr, n, flags);

if(ret == -1) throw std::runtime\_error("recvn failed.");

return ret;

}

static ssize\_t recvall\_ex(SOCKET fd, void \*\*pvptr, size\_t initSize, int flags) //never return -1

{

auto ret = recvall(fd, pvptr, initSize, flags);

if(ret == -1) throw std::runtime\_error("recvn failed.");

return ret;

}

};

class fdIO

{

public:

static ssize\_t readn(SOCKET fd, void \*vptr, size\_t n) noexcept //Return -1 on error, read bytes on success, blocks until nbytes done.

{

return sockIO::recvn(fd, (char \*)vptr, n, 0);

}

static ssize\_t writen(SOCKET fd, const void \*vptr, size\_t n) noexcept //Return -1 on error, read bytes on success, blocks until nbytes done.

{

return sockIO::sendn(fd, (const char \*)vptr, n, 0);

}

static ssize\_t readall(SOCKET fd, void \*\*pvptr, size\_t initSize) noexcept //Return -1 on error, read bytes on success. pvptr must be a malloc/calloced buffer, I'll malloc one if \*pvptr is NULL.

{

return sockIO::recvall(fd, pvptr, initSize, 0);

}

static void readn\_ex(SOCKET fd, void \*vptr, size\_t n) //return read bytes.

{

return sockIO::recvn\_ex(fd, (char \*)vptr, n, 0);

}

static ssize\_t writen\_ex(SOCKET fd, const void \*vptr, size\_t n)

{

return sockIO::sendn\_ex(fd, (const char \*)vptr, n, 0);

}

static ssize\_t readall\_ex(SOCKET fd, void \*\*pvptr, size\_t initSize) //never return -1

{

return sockIO::recvall\_ex(fd, pvptr, initSize, 0);

}

};

}

#endif

#endif

//./class\_decorator.hpp

#ifndef RLIB\_CLASS\_DECO\_HPP\_

#define RLIB\_CLASS\_DECO\_HPP\_

#include <rlib/require/cxx11>

namespace rlib {

namespace \_noncp\_ {

class noncopyable

{

public:

noncopyable() = default;

~noncopyable() = default;

noncopyable(const noncopyable &) = delete;

noncopyable &operator=(const noncopyable &) = delete;

};

}

typedef \_noncp\_::noncopyable noncopyable;

}

namespace rlib {

namespace \_nonmv\_ {

class nonmovable : private noncopyable

{

public:

nonmovable() = default;

~nonmovable() = default;

nonmovable(const nonmovable &&) = delete;

nonmovable &operator=(const nonmovable &&) = delete;

};

}

typedef \_nonmv\_::nonmovable nonmovable;

}

namespace rlib {

namespace \_nonconstructible\_ {

class nonconstructible : private rlib::nonmovable

{

public:

nonconstructible() = delete;

~nonconstructible() = delete;

};

}

typedef \_nonconstructible\_::nonconstructible nonconstructible;

typedef nonconstructible static\_class;

}

#endif//./opt.hpp

/\*

This opt\_parser works well for correct cmd args,

but not guaranteed to works well in all condition

(for example, some ill formed argument).

It's possible to read wrong information rather than

raise an exception on some rare ill formed arguments.

\*/

#ifndef R\_OPT\_HPP

#define R\_OPT\_HPP

#include <rlib/require/cxx14>

#include <rlib/noncopyable.hpp>

#include <rlib/string/fstr.hpp>

#include <rlib/scope\_guard.hpp>

#include <string>

#include <vector>

#include <algorithm>

#include <stdexcept>

namespace rlib {

class opt\_parser : private noncopyable

{

public:

opt\_parser() = delete;

opt\_parser(size\_t arglen, char \*\*argv) {

for(size\_t cter = 1; cter < arglen; ++cter)

args.push\_back(std::move(std::string(argv[cter])));

}

std::string getValueArg(const std::string &argName, bool required = false)

{ //If required argument not exist, I'll throw. Else, return "" if arg is not read.

bool useEqualSym = false;

auto pos = std::find\_if(args.cbegin(), args.cend(), [&](auto &ele)->bool{

if(ele == argName) return true;

if(ele.size() > argName.size() && ele.substr(0, argName.size()+1) == argName + "=") {

useEqualSym = true;

return true;

}

return false;

});

if(required && pos == args.cend())

throw std::invalid\_argument(fstr("Required argument '%s' not provided.", argName.c\_str()));

if(pos == args.cend())

return std::move(std::string(""));

defer(([&, pos]{if(!useEqualSym) args.erase(pos+1); args.erase(pos);}));

if(useEqualSym)

return std::move(pos->substr(argName.size() + 1));

else

{

if(++pos == args.cend())

throw std::invalid\_argument(fstr("Argument '%s' must provide value.", argName.c\_str()));

return \*pos;

}

}

std::string getValueArg(const std::string &argName, const char \*pAnotherCStr)

{ //getValueArg("--long", "-l") may be converted to getValueArg("--long", true).

return std::move(getValueArg(argName, pAnotherCStr, false));

}

bool getBoolArg(const std::string &argName)

{ //Return if it's defined.

auto pos = std::find(args.cbegin(), args.cend(), argName);

if(pos == args.cend()) return false;

args.erase(pos);

return true;

}

std::string getValueArg(const std::string &longName, const std::string &shortName, bool required = false)

{

std::string valueL = getValueArg(longName);

std::string valueS = getValueArg(shortName);

std::string value = valueL.empty() ? valueS : valueL;

if(required && value.empty())

throw std::invalid\_argument(fstr("Required argument '%s/%s' not provided.", longName.c\_str(), shortName.c\_str()));

return value;

}

bool getBoolArg(const std::string &longName, const std::string &shortName)

{

return getBoolArg(longName) || getBoolArg(shortName);

}

bool allArgDone() const

{

return args.empty();

}

private:

std::vector<std::string> args;

};

}

#endif

//./libr.cc

namespace rlib {

bool enable\_endl\_flush = true;

}//./stdio.hpp

#ifndef R\_STDIO\_HPP

#define R\_STDIO\_HPP

#include <rlib/require/cxx11>

// Must link libr.a

#include <string>

#include <iostream>

#include <rlib/string/string.hpp>

namespace rlib {

template<typename PrintFinalT>

void print(PrintFinalT reqArg);

template<typename Required, typename... Optional>

void print(Required reqArgs, Optional... optiArgs);

template<typename... Optional>

void println(Optional... optiArgs);

void println();

template<typename Iterable, typename Printable>

void print\_iter(Iterable arg, Printable spliter);

template<typename Iterable, typename Printable>

void println\_iter(Iterable arg, Printable spliter);

template<typename Iterable>

void print\_iter(Iterable arg);

template<typename Iterable>

void println\_iter(Iterable arg);

template<typename... Args>

size\_t printf(const std::string &fmt, Args... args);

template<typename... Args>

size\_t printfln(const std::string &fmt, Args... args);

inline std::string scanln()

{

::std::string line;

::std::getline(::std::cin, line);

return std::move(line);

}

// Implements.

extern bool enable\_endl\_flush;

template< class CharT, class Traits >

std::basic\_ostream<CharT, Traits>& endl(std::basic\_ostream<CharT, Traits>& os) {

os << '\n';

if(enable\_endl\_flush)

os.flush();

return os;

}

template<typename PrintFinalT>

void print(PrintFinalT reqArg)

{

::std::cout << reqArg;

}

template<typename Required, typename... Optional>

void print(Required reqArgs, Optional... optiArgs)

{

::std::cout << reqArgs << ' ';

print(optiArgs ...);

}

template<typename... Optional>

void println(Optional... optiArgs)

{

print(optiArgs ...);

println();

}

inline void println()

{

::std::cout << ::rlib::endl;

}

template<typename Iterable, typename Printable>

void print\_iter(Iterable arg, Printable spliter)

{

for(const auto & i : arg)

::std::cout << i << spliter;

}

template<typename Iterable, typename Printable>

void println\_iter(Iterable arg, Printable spliter)

{

print\_iter(arg, spliter);

::std::cout << ::rlib::endl;

}

template<typename Iterable>

void print\_iter(Iterable arg)

{

for(const auto & i : arg)

::std::cout << i << ' ';

}

template<typename Iterable>

void println\_iter(Iterable arg)

{

print\_iter(arg);

::std::cout << ::rlib::endl;

}

template<typename... Args>

size\_t printf(const std::string &fmt, Args... args)

{

std::string to\_print = format\_string(fmt, args...);

::std::cout << to\_print;

return to\_print.size();

}

template<typename... Args>

size\_t printfln(const std::string &fmt, Args... args)

{

size\_t len = ::rlib::printf(fmt, args...);

::std::cout << ::rlib::endl;

return len + 1;

}

}

#endif

//./README.md

# rlib

Here is recolic's private library...

//./terminal.hpp

#ifndef R\_STD\_COLOR\_HPP

#define R\_STD\_COLOR\_HPP

#include <rlib/require/cxx11>

#include <rlib/sys/os.hpp>

#include <iostream>

#include <string>

#include <stdexcept>

#include <exception>

using std::string;

using std::basic\_ostream;

namespace rlib::terminal {

enum class color\_t {color\_unset = 10, black = 0, red, green, brown, blue, magenta, cyan, lightgray};

enum class font\_t {font\_unset = 0, bold = 1, underline = 4, dark = 2, background = 7, striked = 9}; //Edit line53 if (int)font\_t may >= 10 !!

class clear\_t {} clear;

class fontInfo

{

public:

fontInfo(color\_t text\_color) : textColor(text\_color) {}

fontInfo(font\_t font\_type) : fontType(font\_type) {}

fontInfo(color\_t text\_color, font\_t font\_type) : textColor(text\_color), fontType(font\_type) {}

fontInfo(const clear\_t &) : clear(true) {}

fontInfo() = default;

string toString() const

{

if(rlib::OSInfo::os == rlib::OSInfo::os\_t::WINDOWS)

return std::move(std::string());

else

return std::move(clear ? std::string("\033[0m") : (color\_to\_string() + font\_to\_string()));

}

private:

color\_t textColor = color\_t::color\_unset;

font\_t fontType = font\_t::font\_unset;

bool clear = false;

private:

constexpr static int color\_to\_int(const color\_t &\_ct)

{

return static\_cast<int>(\_ct);

}

constexpr static int font\_to\_int(const font\_t &\_ft)

{

return static\_cast<int>(\_ft);

}

constexpr static char color\_to\_char(const color\_t &\_ct)

{

return \_ct == color\_t::color\_unset ? '\0' : '0' + color\_to\_int(\_ct); //Return '\0' if unset.

}

constexpr static char font\_to\_char(const font\_t &\_ft)

{

return \_ft == font\_t::font\_unset ? '\0' :'0' + font\_to\_int(\_ft);

}

string color\_to\_string() const

{

if(textColor == color\_t::color\_unset)

return std::move(std::string());

char toret[] = "\033[3?m";

toret[3] = color\_to\_char(textColor);

return std::move(std::string(toret));

}

string font\_to\_string() const

{

if(fontType == font\_t::font\_unset)

return std::move(std::string());

char toret[] = "\033[?m";

toret[2] = font\_to\_char(fontType);

return std::move(std::string(toret));

}

};

struct \_rosi\_font {\_rosi\_font(const fontInfo &\_ref\_fi) : \_ref\_fi(\_ref\_fi) {} const fontInfo &\_ref\_fi;};

inline \_rosi\_font setfont(const fontInfo &\_\_fi) {return \_rosi\_font(\_\_fi);}

template<typename \_CharT, typename \_Traits>

inline basic\_ostream<\_CharT, \_Traits>&

operator<<(basic\_ostream<\_CharT, \_Traits>& \_\_os, const fontInfo &\_\_f)

{

\_\_os << \_\_f.toString();

return \_\_os;

}

template<typename \_CharT, typename \_Traits>

inline basic\_ostream<\_CharT, \_Traits>&

operator<<(basic\_ostream<\_CharT, \_Traits>& \_\_os, \_rosi\_font \_\_rosi\_f)

{

const fontInfo &\_\_f = \_\_rosi\_f.\_ref\_fi;

return operator<<<\_CharT, \_Traits>(\_\_os, \_\_f);

}

}

#endif

//./scope\_guard\_buffer.hpp

/\*

scope\_guards scope\_exit, scope\_fail;

action1();

scope\_exit += [](){ cleanup1(); };

scope\_fail += [](){ rollback1(); };

action2();

scope\_exit += [](){ cleanup2(); };

scope\_fail += [](){ rollback2(); };

//...

scope\_fail.dismiss();

\*/

#ifndef R\_SCOPE\_GUARD\_BUFFER\_HPP

#define R\_SCOPE\_GUARD\_BUFFER\_HPP

#include <rlib/require/cxx11>

#include <functional>

#include <deque>

#include <rlib/class\_decorator.hpp>

namespace rlib {

class scope\_guards : public std::deque<std::function<void()>>, private noncopyable

{

public:

template<class Callable>

scope\_guards& operator += (Callable && undo\_func) {

emplace\_front(std::forward<Callable>(undo\_func));

}

~scope\_guards() {

for(auto &f : \*this) f(); // must not throw

}

void dismiss() noexcept {

clear();

}

};

}

#endif

//./macro.hpp

#ifndef R\_MACRO\_HPP

#define R\_MACRO\_HPP

#ifndef MACRO\_DECAY

#define MACRO\_DECAY(m) (m)

#endif

#ifndef \_R\_MACRO\_ENSTRING

#define \_R\_MACRO\_ENSTRING(\_s) #\_s

#endif

#ifndef MACRO\_TO\_CSTR

#define MACRO\_TO\_CSTR(m) \_R\_MACRO\_ENSTRING(m)

#endif

#ifndef MACRO\_EQL

#define MACRO\_EQL(a, b) (MACRO\_TO\_CSTR(a) == MACRO\_TO\_CSTR(b))

#endif

#ifndef MACRO\_CAT

#define MACRO\_CAT(a, b) \_MACRO\_CAT\_I(a, b)

#define \_MACRO\_CAT\_I(a, b) \_MACRO\_CAT\_II(~, a ## b)

#define \_MACRO\_CAT\_II(p, res) res

#endif

#ifndef MAKE\_UNIQUE\_NAME

#define MAKE\_UNIQUE\_NAME(base) MACRO\_CAT(base, \_\_COUNTER\_\_)

#endif

#endif

//./string/string.hpp

#ifndef R\_STRING\_HPP

#define R\_STRING\_HPP

#include <vector>

#include <string>

#include <cstdarg>

#include <cstdio>

#include <cstdlib>

#include <stdexcept>

#include <sstream>

#include <type\_traits>

namespace rlib {

std::vector<std::string> splitString(const std::string &toSplit, const char &divider = ' ');

std::vector<std::string> splitString(const std::string &toSplit, const std::string &divider);

template <class ForwardIterator>

std::string joinString(const char &toJoin, ForwardIterator begin, ForwardIterator end);

template <class ForwardIterator>

std::string joinString(const std::string &toJoin, ForwardIterator begin, ForwardIterator end);

template <class ForwardIterable>

std::string joinString(const char &toJoin, ForwardIterable begin, ForwardIterable end);

template <class ForwardIterable>

std::string joinString(const std::string &toJoin, ForwardIterable begin, ForwardIterable end);

size\_t replaceSubString(std::string& str, const std::string &from, const std::string& to);

bool replaceSubStringOnce(std::string& str, const std::string& from, const std::string& to);

template<typename... Args>

std::string format\_string\_c(const std::string &fmt, Args... args);

template<typename... Args>

std::string format\_string(const std::string &fmt, Args... args);

//Implements.

char \*\_format\_string\_c\_helper(const char \*fmt, ...);

template<typename... Args>

std::string format\_string\_c(const std::string &fmt, Args... args)

{

char \*res = \_format\_string\_c\_helper(fmt.c\_str(), args ...);

std::string s = res;

free(res);

return std::move(s);

}

template<typename StdString>

void \_format\_string\_helper(std::stringstream &ss, const StdString &fmt) {

static\_assert(std::is\_same<StdString, std::string>::value, "incorrect argument type to \_format\_string\_helper");

ss << fmt;

}

template<typename Arg1, typename... Args>

void \_format\_string\_helper(std::stringstream &ss, const std::string &fmt, Arg1 arg1, Args... args) {

size\_t pos = 0;

while((pos = fmt.find("{}")) != std::string::npos) {

if(pos != 0 && fmt[pos-1] == '\\') {

++pos;

continue;

}

ss << fmt.substr(0, pos) << arg1;

\_format\_string\_helper(ss, fmt.substr(pos + 2), args ...);

return;

}

\_format\_string\_helper(ss, fmt);

}

template<typename... Args>

std::string format\_string(const std::string &fmt, Args... args) {

std::stringstream ss;

\_format\_string\_helper(ss, fmt, args...);

return ss.str();

}

inline std::vector<std::string> splitString(const std::string &toSplit, const char &divider)

{

std::vector<std::string> buf;

size\_t curr = 0, prev = 0;

while((curr = toSplit.find(divider, curr)) != std::string::npos) {

buf.push\_back(toSplit.substr(prev, curr - prev));

++curr; // skip divider

prev = curr;

}

buf.push\_back(toSplit.substr(prev));

return std::move(buf);

}

inline std::vector<std::string> splitString(const std::string &toSplit, const std::string &divider)

{

std::vector<std::string> buf;

size\_t curr = 0, prev = 0;

while((curr = toSplit.find(divider, curr)) != std::string::npos) {

buf.push\_back(toSplit.substr(prev, curr - prev));

curr += divider.size(); // skip divider

prev = curr;

}

buf.push\_back(toSplit.substr(prev));

return std::move(buf);

}

template <class ForwardIterator>

std::string joinString(const char &toJoin, ForwardIterator begin, ForwardIterator end) {

std::string result;

for(ForwardIterator iter = begin; iter != end; ++iter) {

if(iter != begin)

result += toJoin;

result += \*iter;

}

return std::move(result);

}

template <class ForwardIterator>

std::string joinString(const std::string &toJoin, ForwardIterator begin, ForwardIterator end) {

std::string result;

for(ForwardIterator iter = begin; iter != end; ++iter) {

if(iter != begin)

result += toJoin;

result += \*iter;

}

return std::move(result);

}

template <class ForwardIterable>

std::string joinString(const std::string &toJoin, ForwardIterable buf) {

auto begin = buf.begin();

auto end = buf.end();

return std::move(joinString(toJoin, begin, end));

}

template <class ForwardIterable>

std::string joinString(const char &toJoin, ForwardIterable buf) {

auto begin = buf.begin();

auto end = buf.end();

return std::move(joinString(toJoin, begin, end));

}

inline size\_t replaceSubString(std::string& str, const std::string &from, const std::string& to)

{

if(from.empty())

return 0;

size\_t start\_pos = 0;

size\_t times = 0;

while((start\_pos = str.find(from, start\_pos)) != std::string::npos)

{

++times;

str.replace(start\_pos, from.length(), to);

start\_pos += to.length(); // In case 'to' contains 'from', like replacing 'x' with 'yx'

}

return times;

}

inline bool replaceSubStringOnce(std::string& str, const std::string& from, const std::string& to)

{

size\_t start\_pos = str.find(from);

if(start\_pos == std::string::npos)

return false;

str.replace(start\_pos, from.length(), to);

return true;

}

inline char \*\_format\_string\_c\_helper(const char \*fmt, ...)

{

int n;

int size = 100; /\* Guess we need no more than 100 bytes \*/

char \*p, \*np;

va\_list ap;

if ((p = (char \*)malloc(size)) == NULL)

throw std::runtime\_error("malloc returns null.");

while (1) {

/\* Try to print in the allocated space \*/

va\_start(ap, fmt);

n = vsnprintf(p, size, fmt, ap);

va\_end(ap);

/\* Check error code \*/

if (n < 0)

throw std::runtime\_error("vsnprintf returns " + std::to\_string(n));

/\* If that worked, return the string \*/

if (n < size)

return p;

/\* Else try again with more space \*/

size = n + 1; /\* Precisely what is needed \*/

if ((np = (char \*)realloc (p, size)) == NULL) {

free(p);

throw std::runtime\_error("make\_message realloc failed.");

} else {

p = np;

}

}

}

}

#endif//./string/fstr.hpp

#ifndef \_SRC\_FSTR\_H

#define \_SRC\_FSTR\_H 1

#include <rlib/require/cxx11>

namespace rlib {

template<typename... Args>

std::string format\_string\_c()(const std::string &fmt, Args... args);

template<typename... Args>

std::string format\_string()(const std::string &fmt, Args... args);

}

#endif //SRC\_FSTR\_H

//./functional.hpp

#ifndef RLIB\_FUNCTIONAL\_HPP\_

#define RLIB\_FUNCTIONAL\_HPP\_

#include <rlib/require/cxx14>

#include <rlib/class\_decorator.hpp>

#include <type\_traits>

#include <list>

#include <functional>

#include <chrono>

namespace rlib {

template <class operation\_t, typename... args\_t>

static inline double timed\_func(::std::function<operation\_t> f, args\_t... args)

{

auto begin = std::chrono::high\_resolution\_clock::now();

f(args ...);

auto end = std::chrono::high\_resolution\_clock::now();

return ::std::chrono::duration<double>(end - begin).count();

}

template <class operation\_t, typename... args\_t>

static inline typename ::std::result\_of<operation\_t(args\_t ...)>::type repeat(size\_t count, operation\_t f, args\_t... args)

{

for(size\_t cter = 0; cter < count - 1; ++cter)

f(args ...);

return ::std::move(f(args ...));

}

template <class operation\_t, typename... args\_t>

static inline ::std::list<typename ::std::result\_of<operation\_t(args\_t ...)>::type> repeat\_and\_return\_list(size\_t count, operation\_t f, args\_t... args)

{

::std::list<typename ::std::result\_of<operation\_t(args\_t ...)>::type> ret;

for(size\_t cter = 0; cter < count; ++cter)

ret.push\_back(std::move(f(args ...)));

return std::move(ret);

}

}

#endif

//./scope\_guard.hpp

/\* Exception safe usage:

\*

\* reinforce\_scope\_begin(\_gname, [](){do\_sth();})

\* 1+1;

\* 1+1=2;

\* 2+2=4;

\* reinforce\_scope\_end(\_gname)

\*

\*/

#ifndef R\_SCOPE\_GUARD

#define R\_SCOPE\_GUARD

#include <rlib/require/cxx11>

#include <functional>

#include <rlib/class\_decorator.hpp>

namespace rlib {

class scope\_guard : private noncopyable

{

public:

template<class Callable>

scope\_guard(Callable && undo\_func) : f(std::forward<Callable>(undo\_func)) {}

scope\_guard(scope\_guard && other) : f(std::move(other.f)) {

other.f = nullptr;

}

~scope\_guard() {

if(f) f(); // must not throw

}

void dismiss() noexcept {

f = nullptr;

}

void force\_call() noexcept {

if(f) f();

dismiss();

}

private:

std::function<void()> f;

};

}

#ifndef defer

#include <rlib/macro.hpp>

#define defer(callable) ::rlib::scope\_guard MAKE\_UNIQUE\_NAME(\_guarder\_id\_) (callable)

#endif

#define reinforce\_scope\_begin(guarderName, callable) scope\_guard guarderName = callable; try{

#define reinforce\_scope\_end(guarderName) } catch(...) { guarderName.force\_call(); throw;}

#endif

//./require/gcc

#ifndef R\_GCC\_REQUIRED

#define R\_GCC\_REQUIRED

#include <rlib/sys/os.hpp>

#ifndef GCC

#define GCC 9876

#else

#error macro 'GCC' is already defined.

#endif

#if \_\_COMPILER\_ID\_\_ != 9876

#error Gcc is required but not detected.

#endif

#undef GCC

#endif//./require/win

#ifndef R\_LINUX\_REQUIRED

#define R\_LINUX\_REQUIRED

#include <rlib/sys/os.hpp>

#ifndef WINDOWS

#define WINDOWS 9876

#else

#error macro 'WINDOWS' is already defined.

#endif

#if \_\_OS\_ID\_\_ != 9876

#error Windows is required but not detected.

#endif

#undef WINDOWS

#endif//./require/cxx17

#ifndef R\_CXX17\_REQUIRED

#define R\_CXX17\_REQUIRED

#include <bits/c++17\_warning.h>

#endif//./require/cxx14

#ifndef R\_CXX14\_REQUIRED

#define R\_CXX14\_REQUIRED

#include <bits/c++14\_warning.h>

#endif//./require/linux

#ifndef R\_LINUX\_REQUIRED

#define R\_LINUX\_REQUIRED

#include <rlib/sys/os.hpp>

#ifndef LINUX

#define LINUX 9876

#else

#error macro 'LINUX' is already defined.

#endif

#if \_\_OS\_ID\_\_ != 9876

#error Linux is required but not detected.

#endif

#undef LINUX

#endif//./require/cxx11

#ifndef R\_CXX11\_REQUIRED

#define R\_CXX11\_REQUIRED

#if \_\_cplusplus < 201103L

#error C++11 is required.

#endif

#endif

4.3.3 算法测试

本次实验的测试程序分两部分。正确性测试的样例是手动生成，保存在pregen\_testcases/tiny\_input，可以用最小的样例快速检验每一个功能的正确性。性能测试的样例由testgen.py生成。它先插入test\_size个节点，然后随机选择节点进行连边，其中保证第一个节点的出度一定大于0。最后依次进行一次dfs和bfs。按照testgen.py中的默认参数设定，每个顶点的出度的期望为36，顶点数从参数获得。它会随机生成多重边和自环，但不一定生成连通图。

为了方便没有python3的机器进行测试，我提前生成了nodes为1K,10K,100K的测试样例，位于./pregen\_testcases/中。由于IO耗时较长，请先将测试样例生成到内存盘中，然后再从内存盘的样例文件直接stdin重定向给./exp4。

实验程序编译时可以选择使用std::list或std::vector进行存储，使用COMPILE\_NO\_ERASE开启删除时invalidate地址的std::vector。使用std::vector的DFS和BFS比前者快一个Θ(n)因子。经测试，在这两种情况下都是连接边的过程耗时最长。下面给出性能比较。

使用std::list：

|  |  |  |  |
| --- | --- | --- | --- |
| nodes | edges | time | memory |
| 1K | 36K | 0.15s | tiny |
| 10K | 360K | 27s | tiny |
| 100K | 3.6M | too long | - |

使用std::vector：

|  |  |  |  |
| --- | --- | --- | --- |
| nodes | edges | time | memory |
| 1K | 36K | 0.04s | tiny |
| 10K | 360K | 0.40s | tiny |
| 100K | 3.6M | 4.3s | tiny |
| 1M | 36M | 44s | 600MBytes |
| 10M | 360M | 570s | 6GBytes |

下面给出使用tiny\_input的正确性测试

//./pregen\_testcases/tiny\_input

#!./exp4

# `./exp4 input` not implemented, so you cannot run `./input` directly.

CreateGraph directed\_unweighted\_graph

Select 0

PutVex 100`n0

PutVex 101`n1

PutVex 102`n2

PutVex 103`n3

PutVex 104`n4

InsertVex 105`n5

InsertVex 106`n6

InsertVex 107`n7

InsertVex 108`n8

InsertVex 109`n9

QuickTraverse

GetVex `n2

GetVex `n6

LocateVex 300

DeleteVex `n9

QuickTraverse

InsertArc `n0`n1

InsertArc `n0`n2

InsertArc `n0`n3

InsertArc `n0`n4

InsertArc `n2`n3

InsertArc `n2`n5

InsertArc `n3`n6

InsertArc `n5`n6

InsertArc `n6`n7

InsertArc `n2`n7

InsertArc `n5`n8

InsertArc `n7`n0

InsertArc `n7`n1

# MultiEdge

InsertArc `n3`n6

# SelfRing

InsertArc `n5`n5

FirstAdjVex `n0

NextAdjVex `n0 `n1

DeleteArc `n0`n1

DFSTraverse

BFSTraverse

以及给出testgen.py：

#!/usr/bin/env python3

import sys

import random

if len(sys.argv) != 2:

print("Usage: ./this.py <test size>")

exit(1)

test\_size = int(sys.argv[1])

prob\_skip\_a\_node = 0.8

edge\_per\_node\_begin = 8

edge\_per\_node\_end = 64

def iToVarName(i):

return 'n'+str(i)

print('''#!./exp4

# `./exp4 input` not implemented, so you cannot run `./input` directly.

CreateGraph directed\_unweighted\_graph

Select 0

''')

for i in range(test\_size):

print('PutVex {}`n{}'.format(i, i))

for i in range(test\_size):

if i != 0 and random.random() < 0.5: # n0 must have edges

continue

for \_ in range(random.randint(edge\_per\_node\_begin, edge\_per\_node\_end)):

# May have multiedge / selfring

j = random.randint(0, test\_size - 1)

print('InsertArc `n{}`n{}'.format(i, j))

print('''

DFSTraverse

BFSTraverse

''')

4.3.4 界面测试

rfaketerm和中间的每一层中间层都复用了过去的框架，采用了较好的实现方式和架构，同时使用了代码自动生成，其已经经历多次实验的考验。简单的测试表明，界面的正确性没有问题。

**4.4 实验小结**

此次实验相比上次做了以下更新：

rlib完善了stdio.hpp，重写了string/string.hpp。将存储在libr.cc中的函数抽出，使部分库不需要链接-lr即可编译。以及小的修复和接口更新。

实验程序框架方面，增加了测试样例生成器，可以生成更大的样例，保证程序鲁棒性和可靠性。完善了脚本功能，增加了注释的语法支持，在察觉到用户正在使用脚本时会停止输出annoying的prompt。其他的不完善细节。

本次实验加深了对图的概念、基本运算的理解，掌握了图的基本运算的实现。熟练了图的逻辑结构和物理结构之间的关系。今后的学习过程中应当多从数据结构的角度分析如何进行数据的处理、存储以方便问题的解决，并要勤加练习达到熟能生巧的地步。

# 参考文献

[1] 严蔚敏等.数据结构（C语言版）.清华大学出版社

[2] [Larry Nyhoff](http://www.calvin.edu/~nyhl/index.html). [ADTs, Data Structures, and Problem Solving with C++.](http://vig.prenhall.com/catalog/academic/product/0,1144,0131409093,00.html)Second Edition, [Calvin College](http://cs.calvin.edu/),2005

[3] 严蔚敏等.数据结构题集（C语言版）.清华大学出版社

[4] ISO/IEC 14882:2011 https://www.iso.org/standard/50372.html

[5] ISO/IEC 14882:2014 https://www.iso.org/standard/64029.html

[6] ISO/IEC 14882:2017 https://www.iso.org/standard/68564.html

[7] ISO/IEC N4700 http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2017/n4700.pdf