華中科技大學

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

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”

2) 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”

3) 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”

4) 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”

5) 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”

6) 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”

7) 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!”

8) 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	输出“后继不存在!”	输出“后继不存在!”

	2.按提示输入要查找的元素值，输入 1		

9) 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”

10) 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”

11) 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	输出“操作成功！当	输出“操作成功！当

	2.按提示输入要操作的线性表序号，输入 2	前线性表：表 2” 按回车后当前线性表表数更新为 2	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”

12) 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”

13) 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”

14) 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。

⑮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

⑯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。

⑰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。

⑱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。

⑲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。

⑳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

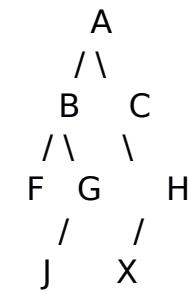
```
CreateBiTree
Select 1
Assign   10
Assign L 40
Assign R 20
Assign LR 320
Assign LL 220
Assign RL 210
Assign LRL 3240
```

.....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,



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 LRLRRLRLRLLRLLLLLRL 210
Assign LRLRRLRLRLLRLLLLLRL 3240
Assign LRLRRLRLRLLRLLLLRL 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

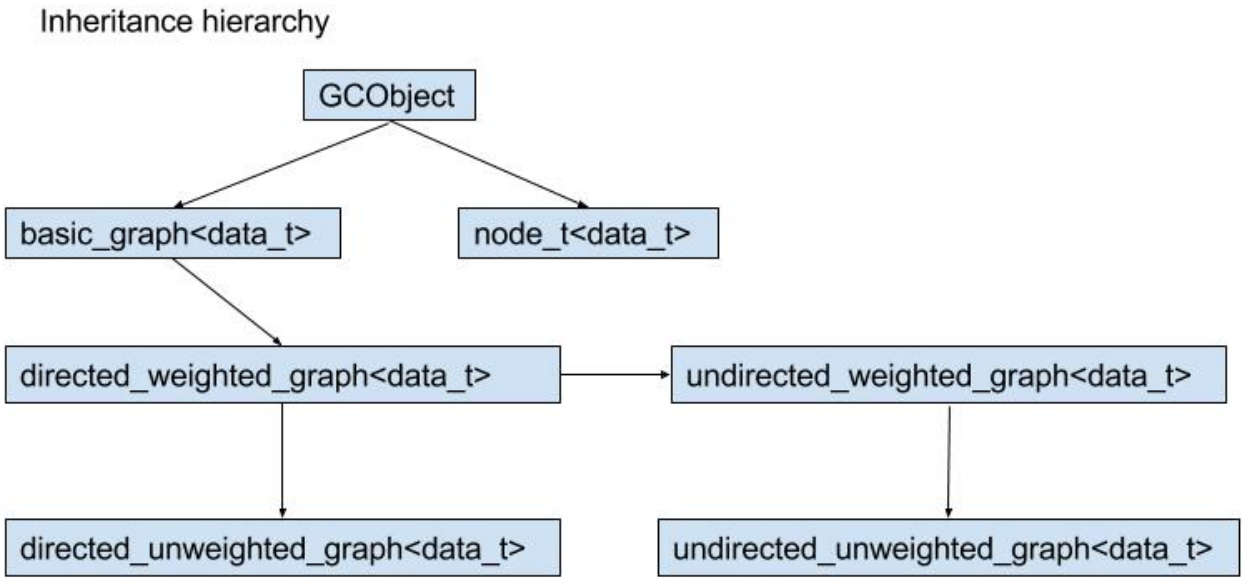
本系统实现图的基本运算。遵守 C++14 标准。

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 $\Theta(n)$ 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 CreateCraph(&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 DestroyBiTree(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::from
EdgeLanguage(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. ADTs, Data Structures, and Problem Solving with C++. Second Edition, Calvin College,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