Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Лабораторная работа № 4

по дисциплине ОАиП

Вариант 4

Выполнил студент гр. 150502: Альхимович Н.Г.

Проверил:         Дулько П.А.

Минск 2022

Задание:

Создать однонаправленное (двунаправленное) кольцо, элементами которого является структура, содержащая информацию об учащемся: фамилия, номер группы, указатель на стек оценок, полученных за сессию. Для кольца предусмотреть возможность его дополнения, сортировки (по фамилии, по номеру группы), удаления информации (по учащимся, получившим три и более неудовлетворительные оценки), поиска и редактирования информации в кольце.

Листинг кода:

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <strings.h>

struct stack

{

int mark;

struct stack \*ptr;

};

struct node

{

struct student

{

char surname[25];

int group;

struct stack \*tos;

} str;

struct node \*prev;

struct node \*next;

};

void add(struct node \*\*); //прототип функции, добавляющей элементы в кольцо

void show(struct node \*); //прототип функции, отображающей содержимое кольца

struct node \*delete(struct node \*); //прототип функции, удаляющей элемент в соответствии с условием

struct node \*sort(struct node \*); //прототип функции, выполняющей сортировку списка

struct node \*choice(char c, struct node \*e); //прототип функции, предоставляющей выбор направления движения по кольцу

void push\_stack(struct stack \*\*); //прототип функции, добавляющей значения в стек - элемент кольца

void show\_stack(struct stack \*); //прототип функции, отображающей содержимое стека - элемента кольца

int search\_stack(struct stack \*); //прототип функции, реализующей поиск по содержимому стека

int main()

{

struct node \*enter; //объявление указателя на точку входа в кольцо

enter = NULL;

while(1)

{

puts("\nВыберите действие: ");

puts(" 1. добавить информацию о студенте;");

puts(" 2. просмотреть информацию о студентах;");

puts(" 3. исключить студента;");

puts(" 4. отсортировать список студентов;");

puts(" 5. завершить работу.");

fflush(stdin);

switch(getchar())

{

case '1': add(&enter); break;

case '2': show(enter); break;

case '3': if(enter) enter = delete(enter);

else printf("\nИнформация о студентах не введена\n"); break;

case '4': if(enter) enter = sort(enter);

else printf("\nИнформация о студентах не введена\n"); break;

case '5': return 0;

default: printf("\nОшибка ввода. Повторите попытку\n");

}

sleep(1); //задержка момента очередного появления меню

}

return 0;

}

void add(struct node \*\*enter)

{

struct node \*temp1, \*temp2;

struct student \*str;

if(!\*enter) //если кольцо еще не создано

{

if(!(temp1 = (struct node \*) calloc(1, sizeof(struct node)))) //создание первого элемента кольца

{

printf("\nПамять не выделена");

return;

}

if(!(str = (struct student \*) calloc(1, sizeof(struct student)))) //указатель на структуру

{

printf("\nПамять не выделена");

return;

}

printf("\nВводите информацию о студенте");

printf("\nФамилия: ");

scanf("%s", temp1->str.surname);

printf("Номер группы: ");

while((scanf("%d", &(temp1->str.group))) != 1)

{

printf("\nНекорректный ввод, повторите попытку: ");

scanf("%d", &(temp1->str.group));

fflush(stdin);

}

printf("Отметки, полученные за сессию: \n");

push\_stack(&(temp1->str.tos));

temp1->prev = temp1->next = temp1; //элемент в обоих направлениях указывает сам на себя

\*enter = temp1; //указатель на кольцо, пока состоящее из одного элемента

}

else temp1 = (\*enter)->next; //указатель направо от точки входа

do

{

if(!(temp2 = (struct node \*) calloc(1, sizeof(struct node))))

{

printf("\nПамять не выделена");

return;

}

if(!(str = (struct student \*) calloc(1, sizeof(struct student))))

{

printf("\nПамять не выделена");

return;

}

printf("\nВводите информацию о студенте");

printf("\nФамилия: ");

scanf("%s", temp2->str.surname);

printf("Номер группы: ");

while((scanf("%d", &(temp2->str.group))) != 1)

{

printf("\nНекорректный ввод, повторите попытку: ");

scanf("%d", &(temp2->str.group));

fflush(stdin);

}

printf("Отметки, полученные за сессию: \n");

push\_stack(&(temp2->str.tos));

temp1->prev = temp2; //связь между имеющимся элементом и новым (по часовой стрелке)

temp2->next = temp1; //связь между имеющимся элементом и новым (против часовой стрелки)

temp1 = temp2; //рабочий указатель передвигается на новый элемент для последующего добавления очередного элемента

printf("Если хотите продолжить вводить данные о студенте, нажмите y, иначе - n: ");

fflush(stdin);

} while(getchar() == 'y');

temp2->prev = \*enter; //связь между последним добавленным элементом и точкой входа (против часовой стрелки)

(\*enter)->next = temp2; //связь между точкой входа и последним добавленным элементом (по часовой стрелке)

}

void push\_stack(struct stack \*\*tos)

{

struct stack \*temp = \*tos;

do

{

if(!(\*tos = (struct stack \*) calloc(1, sizeof(struct stack)))) //указатель на новый элемент стека

{

printf("\nПамять не выделена");

return;

}

while((scanf("%d", &((\*tos)->mark))) != 1)

{

printf("\nНекорректный ввод, повторите попытку: ");

scanf("%d", &((\*tos)->mark));

fflush(stdin);

}

(\*tos)->ptr = temp; //связь между новым элементом и вершиной стека

temp = \*tos; //новый элемент становится вершиной стека

printf("Если хотите продолжить вводить отметки, нажмите y, иначе - n: ");

fflush(stdin);

} while(getchar() == 'y');

}

void show(struct node \*e)

{

struct node \*temp1;

char c;

if(!e)

{

printf("\nИнформация о студентах не введена\n");

return;

}

temp1 = e; //рабочий указатель на точку входа в кольцо

printf("\nЕсли хотите просмотреть информацию, двигаясь по часовой стрелке, нажмите r, иначе - l\n");

fflush(stdin);

switch(c = getchar())

{

case 'r': case 'R':

case 'l': case 'L':

do

{

printf("\n%s\n", temp1->str.surname);

printf("%d\n", temp1->str.group);

show\_stack(temp1->str.tos);

printf("\n");

temp1 = choice(c, temp1); //переход к следующему элементу кольца

} while(temp1 != e); //пока вновь не достигнута точка входа

break;

default: printf("\nОшибка ввода. Повторите попытку\n"); return;

}

printf("\nВся информация выведена\n");

return;

}

void show\_stack(struct stack \*tos)

{

if(!tos)

{

printf("Информация об отметках не введена");

return;

}

do

{

printf("%d ", tos->mark);

tos = tos->ptr; //переход к предыдущему элементу стека

} while(tos);

}

struct node \*choice(char c, struct node \*e)

{

switch (c)

{

case 'r': case 'R': return e->next; //начиная с последних обновлений (по часовой стрелке)

case 'l': case 'L': return e->prev;

default: printf("\nОшибка ввода. Повторите попытку\n"); return -1;

}

}

struct node \*delete(struct node \*e)

{

struct node \*temp1;

int counter = 0;

temp1 = e;

do

{

counter = search\_stack(temp1->str.tos); //подсчет неудовлетворительных отметок

if(counter != 3) temp1 = temp1->next; //если текущий элемент не подлежит удалению

else

{

if(temp1->next == temp1) //если в списке, имеющем организацию двунаправленного кольца, всего один элемент

{

free(temp1);

return NULL;

}

if(e == temp1) e = e->next; //смещение точки входа, если она подлежит удалению

temp1->prev->next = temp1->next; //разрывается связь по часовой стрелке

temp1->next->prev = temp1->prev; //разрывается связь против часовой стрелки

free(temp1); //удаление элемента

printf("\nИсключение произведено\n");

return e;

}

} while(temp1 != e);

printf("\nСтуденты с более, чем тремя неудовлетворительными отметками не найдены\n");

return e;

}

int search\_stack(struct stack \*tos)

{

int counter = 0;

do

{

if(tos->mark < 4) counter++; //если найдена неудовлетворительная отметка, счетчик увеличивается

tos = tos->ptr; //переход к следующему элементу стека

if(counter == 3) break; //если условие уже точно выполняется, происходит выход из цикла

} while(tos);

return counter;

}

struct node \*sort(struct node \*e)

{

struct node \*temp1, \*temp2, \*temp3;

char c;

printf("\nЕсли хотите отсортировать список студентов по фамилии, нажмите s; если по номеру группы, нажмите g\n");

fflush(stdin);

c = getchar();

temp1 = e; //указатель на упорядочиваемый элемент

do

{

temp2 = temp1->next; //рабочий указатель, движущийся по списку, имеющему организацию двунаправленного кольца

temp3 = temp1; //указатель на элемент, подходящий под условие сортировки

do

{

if(c == 's')

{

if(strcmp(temp3->str.surname, temp2->str.surname) > 0) temp3 = temp2;

}

else if(c == 'g')

{

if(temp3->str.group > temp2->str.group) temp3 = temp2;

} //сравнение данного элемента с упорядочиваемым. если условие выполняется, элемент запоминается

else

{

printf("\nОшибка ввода. Повторите попытку\n");

return e;

}

temp2 = temp2->next; //движение по списку продолжается

} while(temp2 != e); //пока не достигнут конец списка

if(temp3 != temp1) //если найден элемент, подходящее под условие

{

if(e == temp1) e = temp3; //если этот элемент - точка входа, ее позиция меняется

temp3->prev->next = temp3->next; //найденный элемент извлекается из списка

temp3->next->prev = temp3->prev;

temp1->prev->next = temp3; //элемент перед упорядочиваемым теперь указывает на найденный (если двигаться по часовой стрелке)

temp3->next = temp1; //упорядочиваемый идет после найденного (если двигаться по часовой стрелке)

temp3->prev = temp1->prev; //восстанавливается с предыдущей частью списка

temp1->prev = temp3; //упорядочиваемый идет после найденного (если двигаться против часовой стрелки)

}

else temp1 = temp1->next; //если подходящий элемент не найден, продолжается движение по списку

} while(temp1->next != e); //пока список не закончится

printf("\nСортировка произведена\n");

return e;

}

Примеры работы программы:



