**ЗВІТ ДО ПР № 10-11**

*Виконав студент 35 групи*

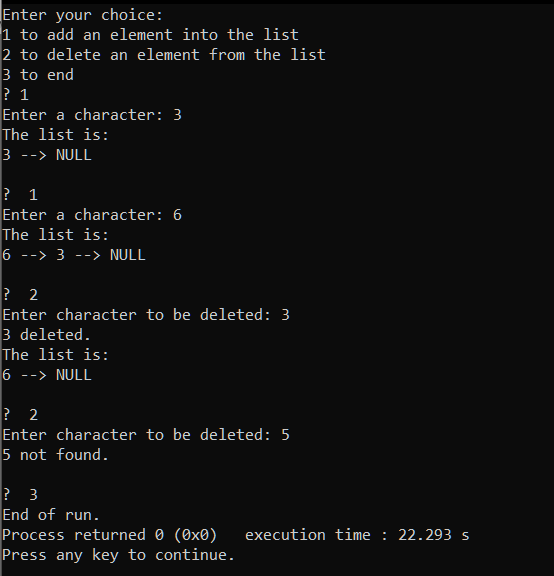
*Рижій Ілля*

**Завдання 1.**

1. Список – це динамічна або лінійна структура даних, в якій кожен елемент посилається або тільки на попередній односпрямований лінійний список, або на попередній і наступний за ним – двонаправлений лінійний список.
2. У однонаправленому списку, який є найпростішим різновидом зв'язаних списків, кожний елемент складається з двох полів: data або даних, та вказівника next на наступний елемент. Якщо вказівник не вказує на інший елемент (інакше: next = NULL), то вважається, що даний елемент — останній в списку.
3. У двонаправленому списку елемент складається з трьох полів — вказівника на попередній елемент prev, поля даних data та вказівника next на наступний елемент. Якщо prev=NULL, то в елемента немає попередника (тобто він є «головою» списку), якщо next=NULL, то в нього немає наступника («хвіст» списка).
4. Доступ до елементів списку здійснюється послідовно, у той час як масиви - структура даних довільного доступу.
5. Зв'язний список – це варіант звичайного лінійного списку, оптимізований для операцій додавання і видалення елементів. Оптимізація полягає в тому, що елементи пов'язаного списку не зобов'язані в пам'яті розташовуватимуться один за одним. Порядок елементів визначається посиланням на перший елемент (не зобов'язаний бути на самому початку виділеної для списку пам'яті) і послідовністю посилань на інші елементи списку.
6. При описі покажчика використовуємо ще не описаний об'єкт struct spis \* next, який буде служити посиланням на наступний елемент списку. Сама остання структура в списку нікуди не вказує, тобто поле next повинно мати значення NULL. Адреса початку (голови) списку зберігається у змінній типу покажчик \* head. На поточну структуру буде вказувати \* p.
7. У двозв'язному списку кожна структура містить два посилання: на попередню й наступну структури. Таким чином, за списком можна переміщатися від початку до кінця і від кінця до початку. Для доступу до початку і кінця списку повинні бути відомі їх адреси, які можуть зберігатися в глобальних змінних типу покажчик.

**Завдання 2.** Скласти програму List.cpp, де реалізувати основні функції роботи зі списком (набір цілих чисел). Передбачати можливість вибору операції над списком:

* Ініціалізація списку
* Додавання елементу у список
* Вилучення елементу із списку
* Виведення елементу (елементів) списку



Код програми:

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

// ініціалізація списку

struct listNode {

int data;

struct listNode \*nextPtr;

};

typedef struct listNode LISTNODE;

typedef LISTNODE \*LISTNODEPTR;

int main()

{

LISTNODEPTR startPtr = NULL;

int choice;

int item;

instructions();

printf("? ");

scanf ("%d", &choice);

while(choice != 3){

switch(choice){

case 1:

printf("Enter a character: ");

scanf("\n%i", &item);

insert(&startPtr, item);

printList(startPtr);

break;

case 2:

if(!isEmpty(startPtr)){

printf("Enter character to be deleted: ");

scanf("\n%i", &item);

if(delete(&startPtr, item)){

printf ("%i deleted.\n", item);

printList(startPtr);

}

else printf("%i not found.\n\n", item);

}

else printf("List is empty.\n\n");

break;

default:

printf("Invalid choice.\n\n");

instructions();

break;

}

printf("? ");

scanf("%i", &choice);

}

printf("End of run.");

return 0;

}

void instructions(void)

{

printf("Enter your choice:\n"

"1 to add an element into the list\n"

"2 to delete an element from the list\n"

"3 to end\n");

}

void insert(LISTNODEPTR \*sPtr, int value) // введення даних у список

{

LISTNODEPTR newPtr, previousPtr, currentPtr;

newPtr = malloc(sizeof(LISTNODE));

if(newPtr != NULL){

newPtr->data = value;

newPtr->nextPtr = NULL;

previousPtr = NULL;

currentPtr = \*sPtr;

if(previousPtr == NULL){

newPtr->nextPtr = \*sPtr;

\*sPtr = newPtr;

}

else{

previousPtr->nextPtr = \*sPtr;

newPtr->nextPtr = currentPtr;

}

}

else

printf("%i not inserted. No memory acailable.\n", value);

}

int delete(LISTNODEPTR \*sPtr, int value) // видалення елементів зі списку

{

LISTNODEPTR previousPtr, currentPtr, tempPtr;

if(value == (\*sPtr)->data){

tempPtr = \*sPtr;

\*sPtr = (\*sPtr)->nextPtr;

free(tempPtr);

return value;

}

else{

previousPtr = \*sPtr;

currentPtr = (\*sPtr)->nextPtr;

while(currentPtr != NULL && currentPtr->data != value){

previousPtr = currentPtr;

currentPtr = currentPtr->nextPtr;

}

if(currentPtr != NULL) {

tempPtr = currentPtr;

previousPtr ->nextPtr = currentPtr -> nextPtr;

free(tempPtr);

return value;

}

}

return 0;

}

int isEmpty(LISTNODEPTR sPtr) // перевірка на заповненість списку

{

return sPtr == NULL;

}

void printList(LISTNODEPTR currentPtr) // виведення списку

{

if(currentPtr == NULL)

printf("List is empty.\n\n");

else{

printf("The list is: \n");

while(currentPtr != NULL){

printf("%i --> ", currentPtr->data);

currentPtr = currentPtr -> nextPtr;

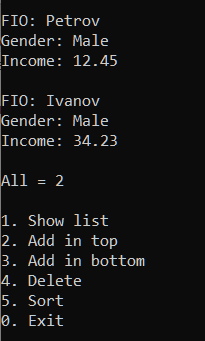
}

printf("NULL\n\n");

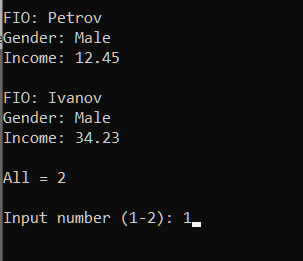
}

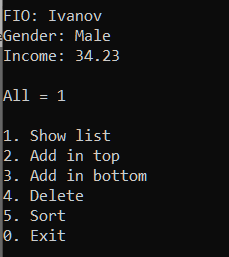
}

**Завдання 3.**

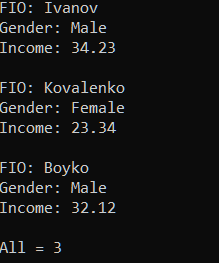


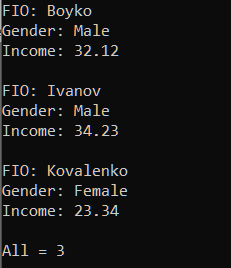
*Виконання функції додавання елементів у список та їх виведення*

**

**

*Виконання функції видалення елемента зі списку*

**

**

*Виконання функції сортування елементів списку*

Код програми:

#include <conio.h>

#include <string.h>

#include <stdlib.h>

#include <stdio.h>

struct person //оголошуємо структуру (список)

{

char fio[100];

char gender[30];

float pay;

int age;

person \*next;

};

person input()

{

person p;

printf("\nFIO->");

fflush(stdin);

scanf("%s", p.fio);

printf("\nGender->");

fflush(stdin);

scanf("%s", p.gender);

printf("\nPay->");

fflush(stdin);

scanf("%f", &p.pay);

return p;

}

int show(person \*head)

{

int count = 0;

if(head)

{

while(1)

{

printf("\nFIO: %s"

"\nGender: %s"

"\nIncome: %.2f\n", head->fio, head->gender, head->pay);

count++;

if(head->next == NULL) break;

head = head->next;

}

}

printf("\nAll = %i\n", count);

return count;

}

void search(person \*head, char \*st)

{

if(head==NULL) return;

struct person \*next = head;

do

{

char \*find = strstr(next->fio, st);

if(find != NULL)

printf ("\n%s",next->fio);

if(next->next == NULL) break;

next = next->next;

} while(1);

}

void copy1(person \*to, person \*from)

{

strcpy(to->fio, from->fio);

strcpy(to->gender, from->gender);

to->pay = from->pay;

to->pay = from->pay;

}

person \*add1(person \*head, person \*st)

{

person \*current = new person;

copy1(current, st);

if(head==NULL)

current->next = NULL;

else{

current->next = head;

head = current;

}

return current;

}

person \*add2(person \*head, person \*st)

{

person \*last = NULL;

if(head != NULL) {

last = head;

while(last->next != NULL)

last = last->next;

}

person \*current = new person;

copy1 (current,st);

current->next = NULL;

if (last) last->next = current;

return current;

}

person \*deleteElement(person \*head0, int n)

{

person \*head = head0;

if(head == NULL) return NULL;

if(n == 1)

{

person \*ptr = head->next;

delete head;

return ptr;

}

person \*prev = NULL, \*start = head;

int i = 1;

while(i < n)

{

prev = head;

head = head->next;

if (head==NULL) return start;

i++;

}

person \*ptr = head->next;

delete head;

prev->next = ptr;

return start;

}

person \*sort(person \*ph)

{

person \*q, \*p, \*pr, \*out=NULL;

while(ph != NULL)

{

q = ph; ph = ph->next;

for(p = out, pr = NULL; p != NULL && strcmp(q->fio, p->fio) > 0; pr = p, p = p->next);

if(pr == NULL){

q->next = out;

out = q;

}

else{

q->next = p;

pr->next = q;

}

}

return out;

}

void instructions(void)

{

printf ("\n1. Show list\

\n2. Add in top \

\n3. Add in bottom \

\n4. Delete \

\n5. Sort \

\n0. Exit \n");

}

int main()

{

person \*head = NULL;

while (1)

{

instructions();

fflush (stdin);

int c;

scanf ("%i",&c);

system("cls");

person p;

person \*cur;

int n, all;

switch (c)

{

case 1:

show(head);

break;

case 2:

p = input();

head = add1(head,&p);

break;

case 3:

p = input();

cur = add2(head,&p);

if (head == NULL)

head = cur;

break;

case 4:

all = show(head);

while(1)

{

printf("\nInput number (1-%d): ", all);

fflush(stdin);

scanf("%i",&n);

if(n >= 1 && n <= all) break;

}

head = deleteElement (head, n);

break;

case 5:

head = sort(head);

break;

default:

exit(0);

break;

}

}

return 0;

}

**Висновки:**  під час практичної роботи узагальнили знання про списки.