Бюджетное учреждение высшего образования   
Ханты-Мансийского автономного округа   
«Сургутский государственный университет»

Политехнический институт

Кафедра автоматики и компьютерных систем

**Отчет**  
по лабораторной работе № 10 «Динамическое распределение памяти»

по дисциплине «Программирование и основы алгоритмизации»

Выполнил: Скорба Р. В.

студент группы 609-11

Проверил: Гришмановский П. В.

доцент кафедры автоматики и компьютерных систем

Сургут

2023 г.

**Цель работы:** Закрепление знаний и получение практических навыков использования динамической памяти в задачах с заранее неизвестным объемом обрабатываемой информации.

**Задание:** Требуется модифицировать программу, созданную в результате выполнения предыдущей работы, таким образом, чтобы для хранения данных использовалась динамически распределяемая память.

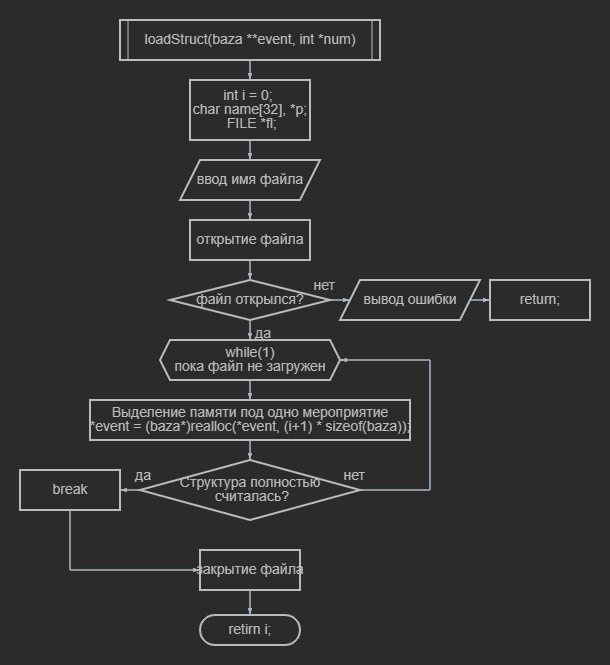
Вариант 27. Задано расписание мероприятий, каждое из которых имеет текстовое описание и характеризуется датой (год, месяц, день) и временем (час, минута) начала и окончания. Обеспечить упорядочивание записей по времени начала, времени окончания, поиск перекрывающихся записей и максимального незанятого промежутка времени. Максимальное количество записей - не менее 25.

**Вариант 27.**

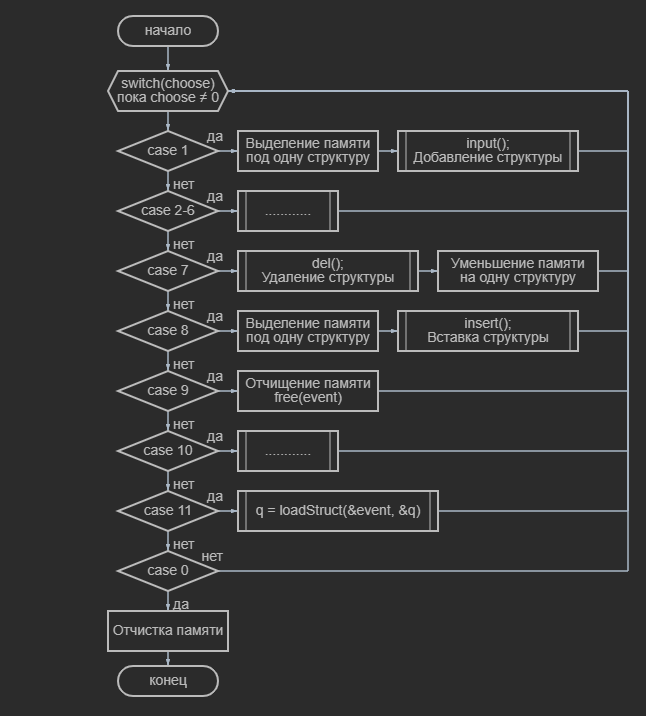
**Блок-схемы алгоритмов:**

В основном выделение памяти реализовано только в операторе switch(рисунок 1). За исключением функции loadStruct(Рисунок 2), там нам заранее неизвестно количество структур в файле и мы выделяем там последовательно под каждую структуру память до тех пор пока не завершиться чтение файла.

Рисунок 1 Рисунок 2



**Листинг строк где используется выделение, освобождение или изменение блоков памяти:**



int main(){

system("chcp 1251 > nul");

baza \*event = NULL;

baza \*t = NULL;

int choose = 0, q = 0;

char name[32];

do {

printf("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n");

printf("Меню - работа с мероприятиями.\n1) Добавить мероприятие.\n2) Просмотреть текущие мероприятия.\n3) Сортировка по времени начала\n4)"

" Сортировка по времени окончания\n5) Поиск перекрывающихся записей\n6) Поиск максимально незанятого промежутка времени"

"\n7) Удаление мероприятия\n8) Вставка мероприятия\n9) Очистить список мероприятий\n"

"10) Сохранить мероприятия\n11) Загрузить мероприятия\n0) Выход\n\nВыберите желаемое действие: ");

while (1){

fflush(stdin);

scanf("%d", &choose);

if (choose >= 0 && choose <= 11)

break;

else printf("\a\nВыберите действие от 0 до 11!\n");

}

printf("\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n");

switch(choose) {

case 1:

t = (baza\*)realloc(event, (q+1) \* sizeof(baza));

if (t != NULL) event = t;

else {

printf("Ошибка выделения памяти!");

break;

}

input(event, q);

q++;

break;

case 2:

output(event, q);

break;

case 3:

s\_tStart(event, q);

printf("Сортировка успешно выполнена!\n\n");

break;

case 4:

s\_tStop(event, q);

printf("Сортировка успешно выполнена!\n\n");

break;

case 5:

s\_Overlap(event, q);

break;

case 6:

s\_free\_interval(event, q);

break;

case 7:

q = del(event, q);

t = (baza\*)realloc(event, q \* sizeof(baza));

if (t != NULL) event = t;

else {

printf("Ошибка выделения памяти!");

break;

}

break;

case 8:

t = (baza\*)realloc(event, (q+1) \* sizeof(baza));

if (t != NULL) event = t;

else {

printf("Ошибка выделения памяти!");

break;

}

q = insert(event, q);

break;

case 9:

free(event);

event = NULL;

q = 0;

printf("Мероприятия удалены!\n\n");

break;

case 10:

saveStruct(event, q);

break;

case 11:

q = loadStruct(&event, &q);

break;

}

} while (choose != 0);

free(event);

event = NULL; // Отчистка памяти перед завершением работы программы

return 0;

}

int loadStruct(baza \*\*event, int \*num){

int i = 0;

char name[32], \*p;

FILE \*fl;

printf("Введите имя файла для загрузки: ");

fflush(stdin);

scanf("%s", name);

strcat(name, ".txt");

fl = fopen(name, "r");

if (fl == NULL){

printf("Данного файла не существует!\n");

return \*num;

}

while(1){

\*event = (baza\*)realloc(\*event, (i+1) \* sizeof(baza));

if(\*event == NULL) {

printf("Ошибка выделения памяти!");

break;

}

if (fscanf(fl, "%d%d%d", &(\*event)[i].day\_start, &(\*event)[i].mounth\_start, &(\*event)[i].year\_start) != 3) {

printf("Файл успешно загружен!\n");

break;

}

if (fscanf(fl, "%d%d\n", &(\*event)[i].hours\_start, &(\*event)[i].min\_start) != 2) {

printf("Файл успешно загружен!\n");

break;

}

if (fscanf(fl, "%d%d%d", &(\*event)[i].day\_stop, &(\*event)[i].mounth\_stop, &(\*event)[i].year\_stop) != 3) {

printf("Файл успешно загружен!\n");

break;

}

if (fscanf(fl, "%d%d\n", &(\*event)[i].hours\_stop, &(\*event)[i].min\_stop) != 2) {

printf("Файл успешно загружен!\n");

break;

}

fgets((\*event)[i].des, 256, fl);

p = strchr((\*event)[i].des, '\n');

if(p != NULL) \*p = 0;

i++;

}

fclose(fl);

return i;

}

Добавление и изменения блоков памяти реализовано с помощью realloc. Она позволяет изменить размер ранее выделенной памяти и получает в качестве аргументов старый указатель и новый размер памяти в байтах.

Для освобождения памяти я использовал free, функция free освобождает память, но при этом не изменяет значение указателя.

**Доказательство того что динамическая память выделяется только при необходимости и вся выделенная динамическая память освобождается к моменту завершения программы:**

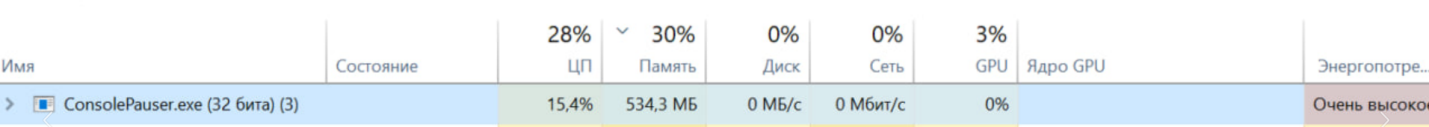
Это довольно легко отследить в основном блоке main, в операторе switch, память выделяется тогда и только тогда, когда пользователю необходимо создать новое мероприятие что исключает случайное выделение памяти. А при завершении программы перед return стоит полное освобождение памяти, что гарантирует при завершении программы полное освобождение памяти.

**Оценка максимального количества объектов, которые могут быть созданы в разработанной программе:**

Я создал бесконечный цикл который будет добавлять готовый шаблон структуры, за час работы было создано 978 308 структур.



Такое количество структур занимало по итогу 543,3 МБ оперативной памяти.



Из этого можно сделать вывод что одна структура занимает около 570 байт памяти, в среднем каждую минуту количество занимаемой памяти увеличивалось на 9 МБ. У меня в системе 16Гб оперативной памяти из которых 700МБ занимает система, при идеальном раскладе может быть создано 26 260 000 структур.

**Вывод:** мною было изучено взаимодействие с динамической памятью в задачах с заранее неизвестным объемом обрабатываемой информации. Научился выделять, освобождать и добавлять размеры блоков памяти.