**Отчет по лабораторной работе №10**

**Динамические массивы**

*РК6-16Б*

16. Петраков Станислав Альбертович

Преподаватель: Федорук Елена Владимировна

**Цель работы**

Изучение функций управления динамической памятью. Применение этих функций для организации динамических массивов.

**Ответы на контрольные вопросы**

1. *Что такое указатель?*

Указатель – переменная, значением которой является адрес ячейки памяти. То есть указатель ссылается на блок данных из области памяти, причём на самое его начало. Указатель может ссылаться на переменную или функцию. Для этого нужно знать адрес переменной или функции.

1. *Какие операции выполняются над указателем?*

Указатели в языке Си поддерживают ряд операций: присваивание, получение адреса указателя, получение значения по указателю, некоторые арифметические операции и операции сравнения.

1. *Что такое адресная арифметика?*

Адресная арифметика Адресная арифметика (address arithmetic) - это способ вычисления адреса какого-либо объекта при помощи арифметических операций над указателями, а также использование указателей в операциях сравнения. Адресную арифметику также называют арифметикой над указателями (pointer arithmetic).

1. *Почему надо правильно объявлять тип указателя?*

Если бы тип указателя ptr\_ar был char, то при использовании оператора инкремента ++, его значение увеличилось бы на единицу, а не на четыре и переход к следующему элементу массива осуществлялся бы некорректно.

1. *Какие функции позволяют выделять память динамически? Чем они отличаются друг от друга?*

Для выделения памяти на куче в си используется функция malloc (memory allocation) из библиотеки stdlib.h.

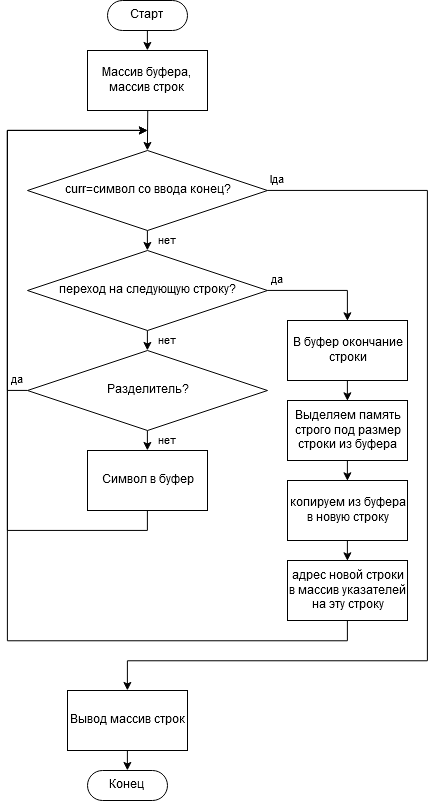
1. *С помощью какой функции память освобождается? Что при этом происходит?*

После того, как мы поработали с памятью, необходимо освободить память функцией free.

**Программа**

*Задание: Разработайте программу, которая читает из стандартного потока ввода строки и размещает их в динамических массивах строк. Перед размещением из строки удаляются все символы-разделители (пробелы и табуляция). Адреса этих массивов запоминаются в массиве указателей. После окончания ввода содержимое массивов выводится в стандартный поток вывода*.

**Блок-схема:**



**Сама программа:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define SIZE\_BUF 1023

#define LINES\_AMOUNT 64

/\*

\*plines - pointer for line

lines - amount of input lines

\*result - pointer for massiv of chars from input

buffer - buffer for curret lines

i - amount of chars current lines

curr - current char

\*/

void output(char\* res[])

{

int i = 0;

putchar('\n');

while (res[i] != NULL)

{

printf("%s", res[i]);

free(res[i]);

i++;

putchar('\n');

}

}

int main()

{

char buffer[SIZE\_BUF + 1], \*plines, curr, \* result[LINES\_AMOUNT];

int i = 0, lines = 0;

while ((curr = getchar()) != '~' && curr != EOF)

{

if (!(curr == '\n'))

{

if (curr == ' ' || curr == '\t') continue;

if (i > SIZE\_BUF)

{

printf("Massiv is overflowed. Terminate.\n");

exit(1);

}

buffer[i] = curr;

i++;

}

else

{

buffer[i] = '\0';

if ((plines = (char\*)malloc((i + 1) \* sizeof(char))) == NULL)

{

printf("Memory allocation error.\n");

exit(2);

}

strcpy(plines, buffer);

if (lines > LINES\_AMOUNT)

{

printf("Massiv is overflowed. Terminate.\n");

exit(1);

}

result[lines] = plines;

lines++;

i = 0;

}

}

result[lines] = NULL;

output(result);

return 0;

}

**Результаты работы программы**

fuisgtuio

ifhbpisu uiuru p9upiosj7 jepioj

ehtoiw

uhofuh rigjpij

~

fuisgtuio

ifhbpisuuiurup9upiosj7jepioj

ehtoiw

uhofuhrigjpij

**Вывод**

Я изучил функции управления динамической памятью malloc и free и научилась их применять для организации динамических массивов.