МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Інститут **ІКНІ**
Кафедра **ПЗ**

ЗВІТ

До лабораторної роботи № 6
На тему: *“особливості роботи з функціями в С. директиви препроцесора”*
З дисципліни: *“Основи програмування”*

Лектор:
ст.викл. каф. ПЗ
Муха Т.О.

Виконав:
ст. гр. ПЗ-15
Марущак А.С.

Прийняла:
асист. каф. ПЗ
Заводовська Н.О.

« ____ » _____ 2021 р.

Σ= ____ .

Львів – 2021

Тема роботи: особливості роботи з функціями в С.Директиви препроцесора.
Мета роботи: поглиблене вивчення можливостей функцій в мові С та основ роботи з препроцесором.

Теоретичні відомості

У ролі аргументів функцій в С можуть використовуватися масиви. У цьому випадку достатньо вказати лише ім'я масиву без необхідності задавати квадратні дужки і розмірність масиву. Так, наприклад, якщо одновимірний масив array оголошений як int array[25]; то передати його у функцію сортування можна так:

```
sort(array);
```

Прототип функції sort можна задати двома способами:

```
void sort(int []);
```

або

```
void sort(int *);
```

С автоматично передає масиви у функції використовуючи механізм виклику функції з передачею адрес. Це означає, що у функцію передається лише адреса масиву (тобто адреса першого елемента масиву), а не його копія (пригадуємо, що у мові С ім'я масиву без індексу є вказівником на перший елемент масиву). Тому функції, яким передається масив, можуть змінювати значення елементів масиву, тобто будь-яка модифікація масиву в функції буде автоматично змінювати масив у точці виклику цієї функції. У тілі самої функції доступ до елементів масиву може здійснюватися або через стандартний механізм індексації, або через механізм вказівника.

Інколи виникають ситуації, коли потрібно заборонити модифікувати елементи масиву всередині функції. Оскільки масив передається за адресою, то контролювати самостійно всі зміни масиву в тілі функції може бути не простою задачею. Краще тут використати спеціальний специфікатор типу мови С, а саме специфікатор const. Коли параметр типу масив описується за допомогою const, то елементи такого масиву в тілі функції стають константами і будь-яка спроба їх модифікувати буде приводити до помилки (для прикладу спробуйте поміняти прототип та заголовок опису функції sort у попередній програмі на void sort(const int [], int); та void sort(const int mas[], int size), відповідно).

Для передачі двовимірних масивів у функції необхідно явно задавати розмірність другого виміру масиву. Це пов'язано з тим, що компілятор використовує це значення розмірності для правильного визначення комірок пам'яті елементів багатовимірного масиву, тобто компілятору потрібна інформація про те скільки елементів знаходиться в одному рядку двовимірного масиву для того, щоб правильно розбити матрицю на окремі рядки.

У С крім функцій з фіксованим числом параметрів є можливість також використовувати функції зі змінною кількістю параметрів, під якими

розуміють функції, в які можна передавати дані, не описуючи їх усіх у прототипі й, відповідно, у заголовку опису функції. Класичним прикладом є успадковані з “чистої” С функції введення/виведення scanf та printf. Список формальних параметрів для функцій із змінним числом параметрів складається з постійних параметрів та змінної частини, яка задається за допомогою конструкції еліпсису, що представляє собою три крапки … , причому спочатку перераховуються постійні параметри. У загальному випадку формат оголошення функції зі змінною кількістю параметрів має такий вигляд:

[тип_результату] ім’я_функції(параметр1, параметр2, ...);

де поля параметр1, параметр2 представляють постійну частину списку параметрів (хоча насправді тут може бути тільки один параметр або більше, ніж два параметри), а еліпсис … - змінну частину, яка означає довільну кількість параметрів.

У стандарт мови С входять макроси для роботи зі списками параметрів змінної довжини. Ці макроси визначені у файлі stdarg.h. При їх використанні також доводиться вказувати у списку постійний параметр, оголосити та встановити на нього вказівник та переміщувати цей вказівник по списку. В кінці списку має бути NULL. Макроси мають наступний формат:

void va_start(va_list prm, останній_фіксований_параметр);
тип va_arg(va_list prm, тип);
void va_end(va_list prm);

Тип вказівника повинен бути va_list, наприклад: va_list p; Макрос va_start встановлює вказівник типу va_list на останній фіксований параметр. Макрос va_arg переміщає вказівник на наступний параметр. Синтаксис наступний: va_arg(вказівник_типу_va_list, тип_чергового_параметра). У момент написання програми програміст повинен знати тип кожного з параметрів. Макрос va_end встановлює вказівник в NULL.

Директива #define має наступний формат:

#define IDENTIFIER text

Або

#define IDENTIFIER(params) text

Використовується для заміни часто вживаних в тексті програми констант, ключових слів, операторів або виразів деякими ідентифікаторами. В останніх двох випадках такі ідентифікатори називаються макровизначеннями або просто макросами. Макроси можуть приймати параметри. Директива #define заміняє всі наступні входження ідентифікатора на вказаний текст. Текстом може бути будь-який фрагмент програми на мові С або пустий фрагмент (відсутність тексту як такого). Для задання багаторядкового тексту заміни можна використати символ \ в кінці кожного рядка, окрім останнього.

ЛАБОРАТОРНЕ ЗАВДАННЯ

1. Ознайомитися з теоретичним матеріалом викладеним вище в даній інструкції і виконати приклади програм.
2. Одержати індивідуальне завдання з Додатку 1.
3. Розробити алгоритм розв’язання індивідуального завдання і подати його у вигляді блоксхеми.
4. Скласти програму на мові С у відповідності з розробленим алгоритмом.
5. Виконати обчислення по програмі.
6. Одержати індивідуальне завдання з Додатку 2.
7. Розробити алгоритм розв’язання індивідуального завдання і подати його у вигляді блоксхеми.
8. Скласти програму на мові С у відповідності з розробленим алгоритмом.
9. Виконати обчислення по програмі при різних значеннях точності і порівняти отримані результати.
10. Одержати індивідуальне завдання з Додатку 3.
11. Скласти програму на мові С для реалізації індивідуального завдання.
12. Підготувати та здати звіт про виконання лабораторної роботи.

Індивідуальне завдання

№1

24. Написати функцію визначення скалярного добутку двох векторів. З її допомогою визначити чи є введена матриця ортогональною, тобто така, що скалярний добуток кожної пари різних рядків дорівнює 0, а скалярний добуток рядка самої на себе відмінний від нуля.

Код програми

(lab06_01_func.h):

```h
#ifndef LAB06_01
#define LAB06_01

#define MAX_SIZE 100
double dot(double vec1[], double vec2[], int size);
int isOrthogonal(double Matrix[][MAX_SIZE], int size);

#endif
```

(lab06_01_func.c):

```c
#include <math.h>
#include "lab06_01_func.h"

double dot(double vec1[], double vec2[], int size){
    double res = 0;
    for(int i = 0; i < size; i++){
        res += vec1[i] * vec2[i];
    }
    return res;
}

int isOrthogonal(double Matrix[][MAX_SIZE], int size){
```

```
    for(int i = 0; i < size - 1; i++){
       for(int j = i; j < size; j++){
           if(i != j && fabs(dot(Matrix[i], Matrix[j], size)) > 1e-4) return 0;
           if(i == j && fabs(dot(Matrix[i], Matrix[j], size)) < 1e-4) return 0;
       }
    }
    return 1;
}
```

(lab06_01.c):

```
#include <stdio.h>
#include <conio.h>
#include "lab06_01_func.h"

int main(){
    int size;
    double M[MAX_SIZE][MAX_SIZE];
    printf("Size: ");
    scanf("%d", &size);
    printf("Elements:\n");
    for(int i = 0; i < size; i++){
        for(int j = 0; j < size; j++){
            scanf("%lf", *(M + i) + j);
        }
    }

    printf(isOrthogonal(M, size) ? "Orthogonal" : "Not orthogonal");
    getch();
    return 0;
}
```

Протокол роботи:

Програма зчитує з клавіатури розмір матриці, а потім її елементи. Після цього перевіряє введену матрицю на ортогональність за допомогою функції перевірки, що використовує функцію скалярного добутку, котра оголошена, як та, що приймає 2 масиви(вектори), їх розмір і повертає результат:

```
F:\Study\Labs\Lab06\lab06_01>lab06_01.exe
Size: 3
Elements:
1 2 3
2 3 4
3 4 5
Not orthogonal
F:\Study\Labs\Lab06\lab06_01>lab06_01.exe
Size: 5
Elements:
1 0 0 0 0
0 0 0 0 1
0 1 0 0 0
0 0 0 1 0
0 0 1 0 0
Orthogonal
F:\Study\Labs\Lab06\lab06_01>lab06_01.exe
Size: 2
Elements:
0.8 -0.6
0.6 0.8
Orthogonal
```

І дійсно, перша матриця не є ортогональною. Друга матриця – одинична з перемішаними стовпцями і є ортогональною за означенням. Третя матриця – матриця повороту і вона також є ортогональною за означенням.

Висновок: Це завдання допомогло нам навчитися оголошувати, реалізовувати та викикати функції, що приймають масиви як аргументи. Це досить важливо, бо функціям нерідко доводиться взаємодіяти з масивами певної інформації.

№2

24. У функцію зі змінним числом параметрів надходять цілі числа, кінець списку – значення -1. Знайти і надрукувати всі парні числа, які не містять цифри 7. Знайти їх кількість або вивести повідомлення про їх відсутність.

Код програми

(lab06_02_func.h):

```h
#ifndef LAB06_02
#define LAB06_02

int containsSeven(int num);
int find_nums(int num, ...);

#endif
```

(lab06_02_func.c):

```c
#include <stdio.h>
#include <math.h>
#include <stdarg.h>
#include "lab06_02_func.h"

int containsSeven(int num){
    num = num < 0 ? -num : num;
    while(num > 0){
        if(num % 10 == 7) return 1;
        num /= 10;
    }
    return 0;
}

int find_nums(int num, ...){
    int n = num;
    va_list args;
    va_start(args, num);
    int count = 0;
    while(n != -1){
        if(n % 2 == 0 && !containsSeven(n)){
            printf("%d ", n);
            count++;
        }
        n = va_arg(args, int);
    }
    va_end(args);
    return count;
}
```

(lab06_02.c):

```c
#include <stdio.h>
#include <conio.h>
#include "lab06_02_func.h"

#define MAX_SIZE 25

int main(){
    int arr[MAX_SIZE] = {0};
    printf("Numbers: ");

    for(int i = 0; i < MAX_SIZE; i++){
        scanf("%d", arr+i);
        if(arr[i] == -1) break;
    }

    printf("Result: ");
    int quantity = find_nums(arr[0], arr[1], arr[2], arr[3], arr[4],
                             arr[5], arr[6], arr[7], arr[8], arr[9],
                             arr[10], arr[11], arr[12], arr[13], arr[14],
                             arr[15], arr[16], arr[17], arr[18], arr[19],
                             arr[20], arr[21], arr[22], arr[23], arr[24]);

    if(quantity != 0) printf("\nFinded numbers: %d\n", quantity);
    else printf("\nNo such numbers.");
    getch();
    return 0;
}
```

Протокол роботи:

Оголошена нами функція приймає змінну к-ть параметрів цілого типу, зчитаних з клавіатури за допомогою масиву і переданих напряму при виклику, проходиться по ним, допоки не зустріне -1, при цьому виводячи парні числа у записі яких відсутня цифра 7. Також функція повертає к-ть таких чисел, що можна потім використати в основній програмі:

```
F:\Study\Labs\Lab06\lab06_02>lab06_02.exe
Numbers: 762534 656325 56735 234 23412 32432 23454 98232 -546 234 -122 -278 -1
Result: 234 23412 32432 23454 98232 -546 234 -122
Finded numbers: 8
```

Функція оперувала наобором чисел 762534 656325 56735 234 23412 32432 23454 98232 -546 234 -122 -278 -1. І дійсно, з цього набору тільки 8 чисел задовольняють цій умові: 234 23412 32432 23454 98232 -546 234 -122.

Висновок:

При виконанні цього завдання ми здобули навички оголошення, реалізації та використання функцій зі змінним числом параметрів. Також познайомились з роботою макросів va_start, va_arg, va_end.

№3

24. Знайти найменше натуральне число Q таке, що добуток його цифр дорівнює заданому числу N. Добуток обчислювати за допомогою макросів з параметрами.

Код програми:

(lab06_03_func.h):

```c
#ifndef LAB06_03
#define LAB06_03

#define POD(num, res) \
    res = 1;\
    while(num > 0){\
        res *= num % 10; \
        num /= 10; \
    } \


int isAnswerExist(int num);

#endif
```

(lab06_03_func.c):

```c
#include "lab06_03_func.h"

int isAnswerExist(int num){
    for(int i = 2; i <= 9; i++){
        while(num % i == 0) num /= i;
    }
    if(num > 1) return 0;
    else return 1;
}
```

(lab06_03.c):

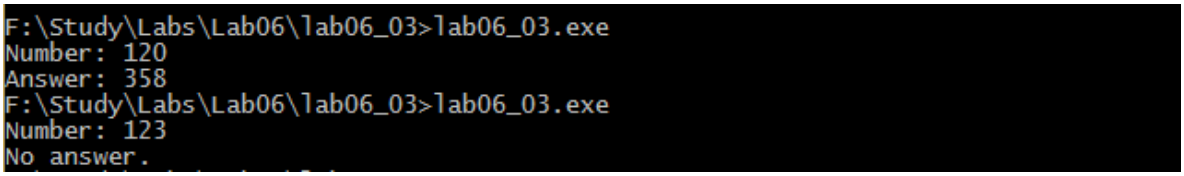
```c
#include <stdio.h>
#include <conio.h>
#include "lab06_03_func.h"

int main(){
    int N, prod, Q = 0;
    printf("Number: ");
    scanf("%d", &N);
    if(isAnswerExist(N)){
        do{
            Q++;
            int temp = Q;
            POD(Q, prod);
            Q = temp;
        }while(prod != N);

        printf("Answer: %d", Q);
    }
    else{
        printf("No answer.");
    }
    return 0;
}
```

Протокол роботи:

Програма приймає число, і, якщо для нього можливо знайти відповідь (це перевіряє окрема функція), то методом перебору за допомогою макроса, що визначає добуток цифр числа, шукає найменше число, що задовольняє умові:



Дійсно, число 120 = 1 * 2 * 3 * 4 * 5 = 3 * 8 * 5, і найменшим числом із цього набору цифр є 358. Для числа 123 неможливо знайти відповідь, оскільки 123 = 3 * 41, запис числа-відповіді мав би містити цифру 3 і цифри, добуток яких дорівнює 41. Але 41 – просте число, і його не можна представити у вигляді добутку цифр в десятковій системі(0-9).

Висновок:

Ми навчилися виносити певний функціонал програми у конструкції мови, що носять назву макроси. Виконання завдання допомогло помітити різницю між макросами і функцій, що допоможе нам у майбутньому.