Міністерство освіти і науки України

Львівський національний університет імені Івана Франка

Факультет електроніки та комп'ютерних технологій

Звіт

Про виконання лабораторної роботи №10

"Вказівники"

Виконав:

Студент групи ФЕП-11с

Качмар Денис

Викладач:

Кужій Юрій Іванович

Львів-2024

Мета: Вивчити поняття і застосування вказівників.

Хід роботи

1) Написав функцію для обчислення довжини стрічки, не використовуючи жодних бібліотек, окрім stdio.h.

```c
#include <stdio.h>

int str_length(const char* str) {
    int length = 0;
    while (str[length] != '\0') {
        length++;
    }
    return length;
}

int main() {
    const char* string = "Hello, world!";
    printf("Length: %d\n", str_length(string));
    return 0;
}
```

```
Length: 13
```

2) Написав функцію з прототипом: int* toPoint(int x, int y). Функція повинна виділити вільну пам’ять на два цілочисельні значення (використайте malloc з stdlib.h), записати в них координати точки (x, y) і повернути вказівник на проініціалізовану точку. В main() створити, вивести точку та вказівник на неї, а потім звільнити виділену пам’ять (використайте free з stdlib.h).

```c
#include <stdio.h>
#include <stdlib.h>

int* toPoint(int x, int y) {
    int* point = (int*)malloc(2 * sizeof(int));
    point[0] = x;
    point[1] = y;
    return point;
}

int main() {
    int x = 5;
    int y = 10;
    int* point = toPoint(x, y);
    printf("Координати точки: (%d, %d)\n", point[0],
point[1]);
    printf("Вказівник на точку: %p\n", (void*)point);
    free(point);
    return 0;
}
```

```
Координати точки: (5, 10)
Вказівник на точку: 0x5612774712a0
```

3) Написав програму, що виконує прості арифметичні операції. Створив чотири функції для додавання, віднімання, множення та ділення дійсних чисел. Зберіг вказівники на ці функції у масив вказівників. Зчитав з консолі ввід типу 1.28+3.14, розбив на операнди і оператор, викликав відповідну функцію з масиву вказівників на операції, вніс результат.

```c
#include <stdio.h>
#include <stdlib.h>

typedef double (*Operation)(double, double);

double add(double a, double b) {
    return a + b;
}

double subtract(double a, double b) {
    return a - b;
}

double multiply(double a, double b) {
    return a * b;
}

double divide(double a, double b) {
    if (b != 0) {
        return a / b;
    }
    printf("Помилка: ділення на нуль\n");
    exit(EXIT_FAILURE);
}

int main() {
    Operation operations[4] = {add, subtract, multiply, divide};
    char operator;
    double operand1, operand2;
    char input[50];

    printf("Введіть вираз (наприклад, 1.28+3.14): ");
    fgets(input, sizeof(input), stdin);

    if (sscanf(input, "%lf %c %lf", &operand1, &operator, &operand2) != 3) {
        printf("Невірний формат вводу\n");
        return 1;
    }

    switch (operator) {
        case '+':
            printf("Результат: %lf\n", operations[0](operand1, operand2));
            break;
        case '-':
            printf("Результат: %lf\n", operations[1](operand1, operand2));
            break;
        case '*':
            printf("Результат: %lf\n", operations[2](operand1, operand2));
            break;
        case '/':
            printf("Результат: %lf\n", operations[3](operand1, operand2));
            break;
        default:
            printf("Невідомий оператор\n");
            return 1;
    }

    return 0;
}
```

```
Введіть вираз (наприклад, 1.28+3.14): 2.45+6.54
Результат: 8.990000
```

Висновок: Виконуючи ці завдання, я зрозумів, як важливо працювати з пам'яттю в C, використовуючи malloc і free, щоб уникнути витоків пам'яті. Створення функцій для арифметичних операцій показало мені, як організувати код за допомогою масивів функцій, що робить програму більш гнучкою. Реалізація власної функції для обчислення довжини стрічки допомогла зрозуміти, як обробляти рядки без стандартних функцій. Ці вправи зміцнили мої знання в програмуванні на C і навички роботи з пам'яттю.