ЛАБОРАТОРНАЯ РАБОТА №5

Цель работы:

Приобрести умения и практические навыки для работы с указателями на функцию и передачей функций как параметров в другие функции; приобрести умения и практические навыки для работы со структурами.

Теоретическая часть:

Имя функции – указатель на функцию, адрес первого байта исполняемого кода функции. Содержит адрес в кодовом сегменте, по которому располагается исполняемый код функции, то есть адрес, по которому передается управление при вызове функции.

Синтаксис:

```
тип (*имя)(список параметров);
```

Работать с указателями на функции можно следующим образом:

```
int (*f)(int, int);
int sum(int a, int b) {return a+b;}
int sub(int a, int b) {return a-b;}
```

Тогда *f* можно присвоить *f = sub; f = sum;*, так как имя функции – адрес первого байта кода функции, то есть *f* становится синонимом *sub* или *sum*.

Например, определим указатель на функцию:

```
void (*message) (void);
```

Здесь определен указатель, который имеет имя message. Он может указывать на функции без параметров, которые возвращают тип void (то есть ничего не возвращают). После названия указателя идет (void), что указывает, что функция не принимает параметров.

Применим этот указатель на функцию:

```
#include <stdio.h>
void hello()
{
```

```
    printf("Hello, World \n");
}
void goodbye()
{
    printf("Good Bye, World \n");
}
int main(void)
{
    void(*message) (void);
    message=hello;
    message();
    message = goodbye;
    message();
    return 0;
}
```

Также стоит обратить внимание на скобки вокруг имени, так, например, если вернуться к использованному выше определению *void (*message) (void)*, то оно не будет аналогично определению *void *message (void)*. Во втором случае определен не указатель на функцию, а прототип функции *message*, которая возвращает указатель типа *void**.

Следует учитывать, что указатель на функцию должен по типу возвращаемого значения и типу параметров соответствовать функции, иначе он не сможет соответствовать этой функции.

Передача функции в другую функцию в качестве параметра производится так: при передаче параметров в функцию в списке формальных параметров указывают тип и имя параметров. При передаче функции в качестве фактического параметра типом этого параметра будет указатель на соответствующую функцию. Если требуется передать функцию в другую функцию в качестве параметра, необходимо:

– иметь описание функции, которую предполагается передавать в другую функцию в качестве параметра;

– описать другую функцию, в которую требуется передавать первую.

Для этого в список параметров другой функции необходимо включить указатель на первую;

– вызывать другую функцию из функции main или другой функции, передавая в нее в качестве параметра имя первой функции.

Прототип такой функции должен иметь следующий вид:

```
тип (*имя (список параметров 1))(список параметров 2);
```

Так должна быть описана функция, в которую передается список параметров 1 и которая возвращает в *main* указатель на функцию со списком параметров 2, который возвращает в вызывающую программу значение указанного типа.

```
# include <stdio.h>
int mul (int a, int b)
{
     return a*b;
}
int sum (int a, int b)
{
     return a+b;
}
int sub (int a, int b)
{
     return a-b;
}
int (* fb (char c)) (int ,int )
{
     switch(c)
     {
     case '*':
          return mul;
          break;
     case '+':
          return sum;
          break;
     case '-':
```

```
            return sub;
            break;
        }
}
void main() {
    printf ("Результат = %d", fb('*')(8,4));
}
```

Структура – составной тип данных, состоящий из фиксированного числа элементов разных типов. Обычно структуры применяются для логического объединения разнородных данных.

Синтаксис:

```
struct имя_структуры {
    тип1 имя1;
    тип1 имя2;
    ...
};
```

Элементы структуры называются *полями*. Они могут иметь любой тип, кроме типа этой же структуры, в том числе и указатели.

В отличие от переменных при определении элементов структуры для них не выделяется память, и их нельзя инициализировать. По сути, просто определяется новый тип данных.

После определения структуры мы можем ее использовать.

```
struct person
{
    char * name;
    int age;
};
struct person tom = {(char*)"Tom", 23};
```

Доступ к полям структуры осуществляется следующим образом:

```
имя_переменной.имя_поля
```

Также инициализировать можно поля по отдельности, обращаясь к ним через точку.

Важно заметить, что имя структуры не является ее адресом.

Над структурами также можно совершать некоторые действия.

Если структуры имеют один и тот же тип, то можно использовать операцию присваивания – при этом выполняется поэлементное копирование. Структуры нельзя сравнивать, но можно использовать операторы == и !=. Структуры можно передавать в функцию в качестве параметра и возвращать из функции.

Размер структуры может не совпадать с размером ее полей, так как при размещении в памяти элементы выравниваются на границу слова.

Задание 1. Написать функцию, которая возвращает значение выражения.

Варианты:

1. $f(x)=2*x;$
2. $f(x)=5+12*x;$
3. $f(x)=7-x;$
4. $f(x)=5*x-32;$
5. $f(x)=7*(1-x);$
6. $f(x)=15-4*x;$
7. $f(x)=(5-x)/x;$
8. $f(x)=x^2-1;$
9. $f(x)=x^3+x^2-1;$
10. $f(x)=x^2-2x-1.$

где x– целочисленная переменная.

Задание 2. Написать функцию *Function*, которая возвращает в вызывающую программу значение выражения в соответствии с вариантом. Здесь *f1*(a) и *f2*(a) – функции с одним параметром целого типа, которые возвращают целочисленное значение (используйте функции из задания 1). Передавать эти функции в *Function* при помощи указателя на функцию. Функция *main* выводит результаты функции *Function.*

Варианты:

1. *f1*(a)+*f2*(a);
2. *f1*(a)-*f2*(a);
3. *f1*(a)**f2*(a);
4. *f1*(a)/*f2*(a);
5. (*f1*(a)+*f2*(a))/*f1*(a);
6. 1/(*f1*(a)+*f2*(a));.
7. (*f1*(a)**f2*(a))-*f1*(a);
8. 1/ (*f1*(a)**f2*(a));
9. $f1(a)^2$-*f2*(a);
10. $f1(a)^3$-$f2(a)^2$.

Задание 3. Написать программу, которая реализует меню из 4 функций, которые выполняют следующие действия: возведение числа a в степень b, произведение a и b, получение остатка от деления a на b, проверка кратности чисел a и b. *Примечание*: a и b – целые числа.

Задание 4. Создать и проинициализировать массив структур из 10 элементов, состоящих из следующих полей:

1. Поля структуры: название книги; автор; год издания. Вывести на экран количество книг, изданных в указанном году. Массив структур проинициализирован при объявлении.

2. Поля структуры: название книги; автор; год издания. Вывести на экран количество книг указанного автора. Массив структур проинициализирован при объявлении.

3. Поля структуры: название книги; автор; год издания. Вывести на экран год издания указанной книги. Массив структур проинициализирован при объявлении.

4. Поля структуры: название книги; автор; год издания. Вывести на экран все названия книг указанного автора. Массив структур проинициализирован при объявлении.

5. Поля структуры: название книги, автор, год издания. Вывести на экран названия всех книг, изданных в указанном году. Массив структур проинициализирован при объявлении.

6. Поля структуры: название детали, цена детали, количество. Вывести на экран название самой дорогой детали. Массив структур проинициализирован при объявлении.

7. Поля структуры: название детали, цена детали, количество. Вывести на экран цену указанной детали. Массив структур проинициализирован при объявлении.

8. Поля структуры: название детали, цена детали, количество. Вывести на экран количество указанной детали в штуках. Массив структур проинициализирован при объявлении.

9. Поля структуры: название детали, цена детали, количество. Вывести на экран название детали, которой больше всего по количеству. Массив структур проинициализирован при объявлении.

10. Поля структуры: название детали, цена детали, количество.

Вывести на экран название самой дешевой детали. Массив структур проинициализирован при объявлении.

Задание 5. Создать и проинициализировать динамический массив структур из 5 элементов, состоящих из следующих полей:

1. Поля структуры: марка машины, пробег, государственный номер. Написать функцию, которая заполняет поля структуры значениями, вводимыми с клавиатуры, и возвращает структуру в вызывающую программу. Вывести заполненную структуру на экран.

2. Поля структуры: название страны, столица страны, численность населения страны в млн. Написать функцию, которая заполняет поля структуры значениями, вводимыми с клавиатуры, и возвращает структуру в вызывающую программу. Вывести заполненную структуру на экран.

3. Поля структуры: имя, фамилия, отчество, возраст. Написать функцию, которая заполняет поля структуры значениями, вводимыми с клавиатуры, и возвращает структуру в вызывающую программу. Вывести заполненную структуру на экран.

Вопросы к теоретическому материалу

1. Чем является имя функции?
2. Укажите синтаксис указателя на функцию.
3. Поясните, для чего можно использовать указатель на функцию.
4. Опишите процесс передачи функции в другую функцию в качестве параметра.
5. Что называется структурой?
6. Укажите синтаксис объявления структуры.
7. Что называется полями?
8. Каким образом осуществляется доступ к полям структуры?
9. Какие действия можно совершать над структурами?

ПРОЦЕСС СДАЧИ ЛАБОРАТОРНОЙ РАБОТЫ

По итогам выполнения каждой лабораторной работы студент:

1. Демонстрирует преподавателю правильно работающие программы;
2. Демонстрирует приобретенные теоретические знания, отвечая на пять вопросов по лабораторной работе;
3. Демонстрирует отчет по выполненной лабораторной работе, соответствующий всем требованиям.

Отчет по лабораторной работе оформляется по шаблону, представленному в приложении 1. Требования к отчету представлены в приложении 2.

ПРИЛОЖЕНИЕ 1. ШАБЛОН ОТЧЕТА

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Казанский национальный исследовательский технический университет им. А.Н. Туполева – КАИ»

Институт компьютерных технологий и защиты информации

Отделение СПО ИКТЗИ (Колледж информационных технологий)

ЛАБОРАТОРНАЯ РАБОТА №

по дисциплине

СИСТЕМНОЕ ПРОГРАММИРОВАНИЕ

Работу выполнил

Студент гр.43__

Фамилия И.О.

Принял

Преподаватель Григорьева В.В.

Казань, 2025 г.

1. **Цель работы.**

2. **Задание на лабораторную работу** – вставляется задание на лабораторную работу, соответствующее индивидуальному, выданному преподавателем, варианту студента.

3. **Результат выполнения работы** – формируется описание хода выполнения работы и вставляются скриншоты с результатами работы разработанных программ (скриншоты должны быть подписаны, например, *Рисунок 1. Начальное состояние программы* и т.п.).

4. **Листинг программы** – вставляется код разработанной программы

ПРИЛОЖЕНИЕ 2. ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЕТА

Лист документа должен иметь книжную ориентацию, поля документа должны составлять: левое – 3 см, правое – 1,5 см, верхнее – 2 см, нижнее 2 см.

Нумерация страниц – внизу страницы по центру, первая страница не нумеруется

Междустрочный интервал – 1,5 (полуторный), отступ первой строки – 1,25.

Текст документа должен быть выполнен с использованием шрифта Times New Roman, размер – 14, выравнивание – по ширине. Заголовки выполняются тем же шрифтом, но размера 16, полужирное начертание, размещение – по центру.

Рисунки должны размещаться по центру, они нумеруются по порядку. Перед рисунком в тексте на него должна быть ссылка. Подпись рисунка должна располагаться по центру и быть выполнена шрифтом Times New Roman, размер – 12. Сначала происходит нумерация рисунка, а затем пишется его название.