Программа, демонстрирующая принцип работы глобальных переменных.

**foo.h**

```
extern int foo;   //Объявление переменной
void inc_foo();   //Прототип функции, работающей с переменной
```

**foo.c**

```
#include "foo.h"

int foo = 0;      //Определение переменной, инициализация значения

void inc_foo()    //Реализация функции inc_foo;
{
    foo++;
}
```

**main.c**

```
#include "foo.h"
#include <stdio.h>

int main()
{
    printf("foo = %d\n", foo);      //Получение значения переменной
    inc_foo();                      //Изменение значения переменной
    printf("foo = %d\n", foo);
    return 0;
}
```

Для компиляции необходимо скомпилировать объектный файл foo.o а затем добавить его в при компиляции main

```
$ gcc –c foo.c –o foo.o

$ gcc foo.o main.c –o program
```

Если попытаться перепределить объявленную глобальную переменную в другом файле (bar.c) (другой единице трансляции) то при линковке программы мы получим ошибку.

```
$ gcc –c bar.c –o bar.o

$ gcc bar.o foo.o main.c –o program

>> /tmp/ccgYtndZ.o:(.data+0x0): multiple definition of `foo'

foo.o:(.bss+0x0): first defined here

collect2: error: ld returned 1 exit status
```

**bar.c**

```
#include "foo.h"

int foo = 3;

void dec_foo()
```

```
{
    --foo;
}
```

Программа, выводящая на экран все переданные аргументы командной строки.

**Options output**

```
#include <stdio.h>

int main(int argc, char * argv[])
{
    for(int i = 0; i < argc; i++)
        printf("Argument %d : %s\n", i, argv[i]);

    return 0;
}
```

Программа, демонстрирующая работу функции `getopt()` для обработки коротких опций (`-v`, `-R` и т. д.)

**Simple options**

```
#include <unistd.h>
#include <stdio.h>


int main(int argc, char *argv[])
{
    char option_symb;

    /* Циклический вызов функции getopt() для обработки всех аргументов. */
    while( (option_symb = getopt(argc, argv, "ab:c")) != -1)
        switch(option_symb)
        {
            case 'a':
                printf("-a is set\n");
                break;
            case 'b':
            /* Опция -b ожидает аргумент. Указатель на него храниться в optarg */
                printf("-b is set with arg = %s\n", optarg);
                break;
            case 'c':
                printf("-c is set\n");
                break;
            case '?':   //'?' возвращается, в случае неправильной опции.
            default :
                printf("Unknown option.\n");
                break;
        }

    return 0;
}
```

Программа, иллюстрирующая работу функции `getopt_long()` для обработки “длинных” опций (`--verbose` и пр.)

**Long options.**

```
#include <stdio.h>
#include <unistd.h>
#include <getopt.h> //Объявления getopt_long() и struct option;

int main(int argc, char *argv[])
{
    int do_all = 0, level = 0; //Начальные значения управляющих флагов
    char *filename;

    /* Инициализация массива структур с информацией об опциях. */
    struct option opts[] = {
        {"all", no_argument, &do_all, 1},
        {"file", required_argument, NULL, 'f'},
        {"loglevel", optional_argument, NULL, 'l'},
        {0, 0, 0, 0}
    };

    char opt;

    /* Аналогичный getopt() цикличный вызов функции. */
    while( (opt = getopt_long(argc, argv, ":f:", opts, NULL)) != -1)
        switch(opt)
        {
            case 'f':
                filename = optarg; //Сохранить имя файла.
                break;
            case 'l':
                break;

            case 0:
                break;

            default:
                printf("Error:\n");
                break;
        }

    /* Вывод результирующих значений. */
    printf("do_all = %d\n", do_all);
    printf("filename = %s\n", filename);
    printf("loglevel = %d\n", level);

    return 0;
}
```

Задание для практики – разобраться с поведением функции `getopt_long()` в случае обработки опций с опциональным аргументом `optional_argument` (здесь `loglevel`). Задача – получить значение, если оно было передано или установить значение по умолчанию.

Две небольшие программы, иллюстрирующие работы системных вызовов `lseek()` и `truncate()`.

**lseek()**

```
#include <sys/types.h>
#include <unistd.h>
#include <fcntl.h>
#include <stdio.h>

#include <string.h>

int main()
{
    int fd = creat("hello.txt",0644); //Создание файла.
    if(fd > 0)
    {
        //Смещение на 100 байт от начала. При создании файла длина равна 0.
        off_t res = lseek(fd, 100, SEEK_END);
        if(res > 0)
        {
            char str[100] = "Hello";
            int writen_bytes = write(fd, str, strlen(str)); //Записываем строку
            if(writen_bytes > 0)
                printf("Success.\n");
        }
        close(fd);      //Закрываем файл.
    }
    return 0;
}
```

**truncate()**

```
#include <unistd.h>
#include <sys/types.h>
#include <stdio.h>

int main()
{
    int res = truncate("hello.txt", 50); //Усекаем существующий файл на 50 байт.
    if(res > 0)
        printf("Truncated.\n");
    return 0;
}
```

Задание для практики – добавить в программы передачу необходимых параметров (имя файла, смещение, позиция для смещения и пр.) с помощью опций командной строки.