Министерство науки и высшего образования

Российской федерации

Федеральное государственное бюджетное

Образовательное учреждение высшего образования

«Новосибирский Государственный Технический Университет»

Кафедра теоретической и прикладной информатики

Лабораторная работа №6

«ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА РАЗРАБОТКИ»

Факультет: прикладной математики и информатики

Группа: ПМИ-12  
Бригада: 1

Студенты: Михайловский М.А.

Швадченко А.В.

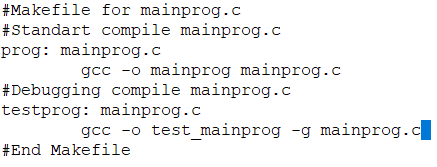
Преподаватели: Кобылянский В.Г.

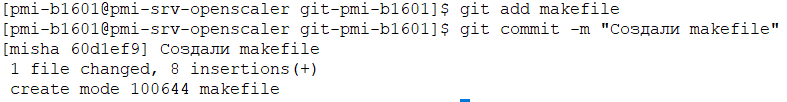
Филиппова Е.В.

Новосибирск, 2023

1. Цель работы:
2. Целью работы является изучение основных этапов разработки и отладки приложений в ОС Linux, а также приобретение практических навыков по использованию инструментальных средств фонда свободного программного обеспечения при компиляции исходного кода, сборке, отладке и тестировании программ, написанных на языке С.
3. Выполнение работы:
4. С помощью редактора **vi** создайте в рабочем каталоге make-файл, указав в качестве аргументов команд имена ваших файлов (см. п. 2.3). Основные сведения по редактору приведены в приложении 1.

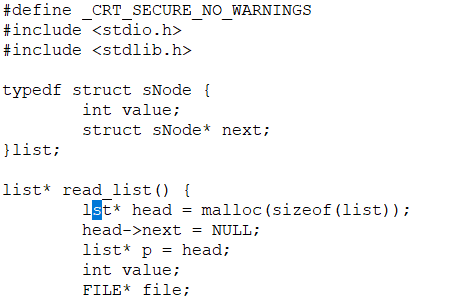
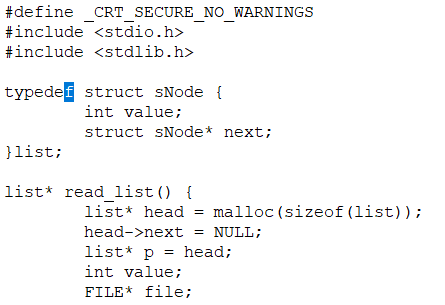




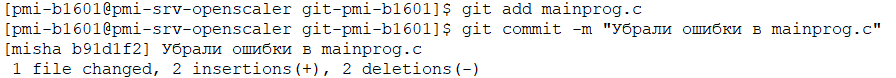
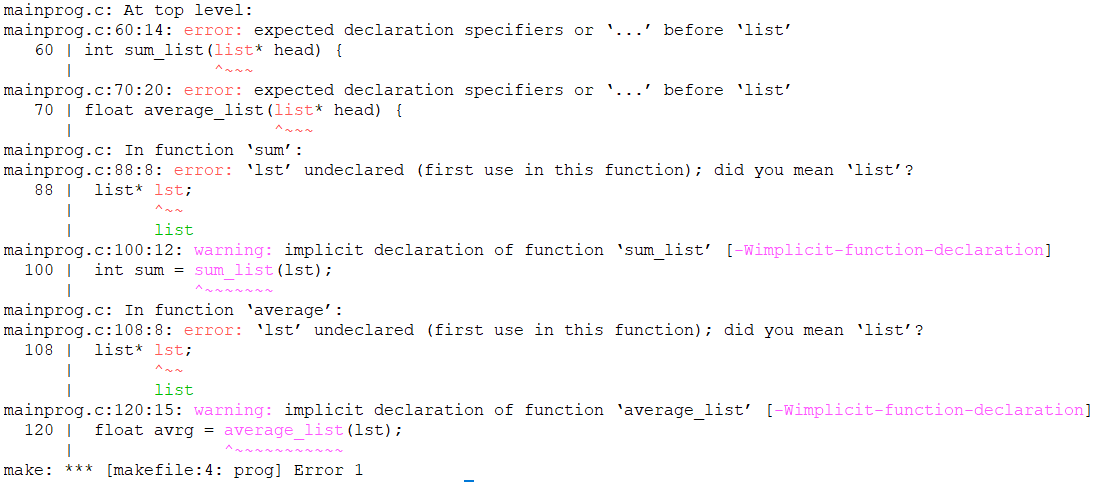
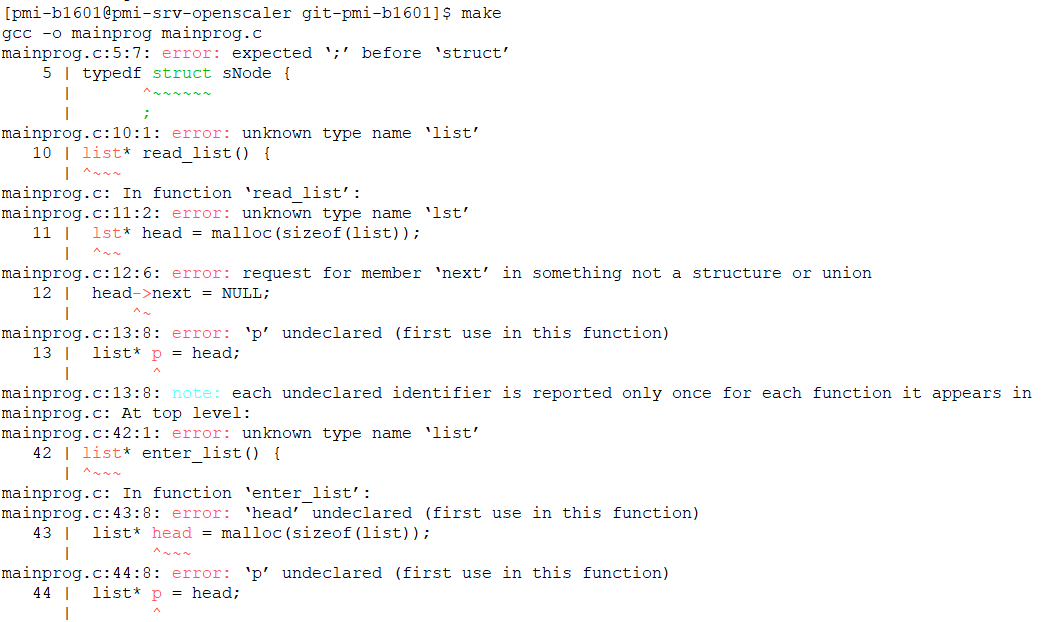
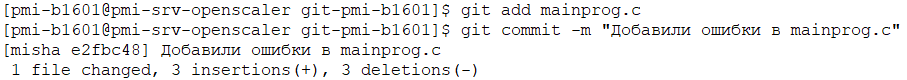


1. Выполните компиляцию файла **mainprog.c** с помощью make-файла, используя правило ***prog***. Если при компиляции были обнаружены ошибки, то используйте для их исправления редактор vi. Каждую исправленную версию программы сохраните в репозитории с поясняющим комментарием. Сведения об ошибках занесите в отчет (номер строки, значение строки до устранения и после устранения ошибки, пояснения).

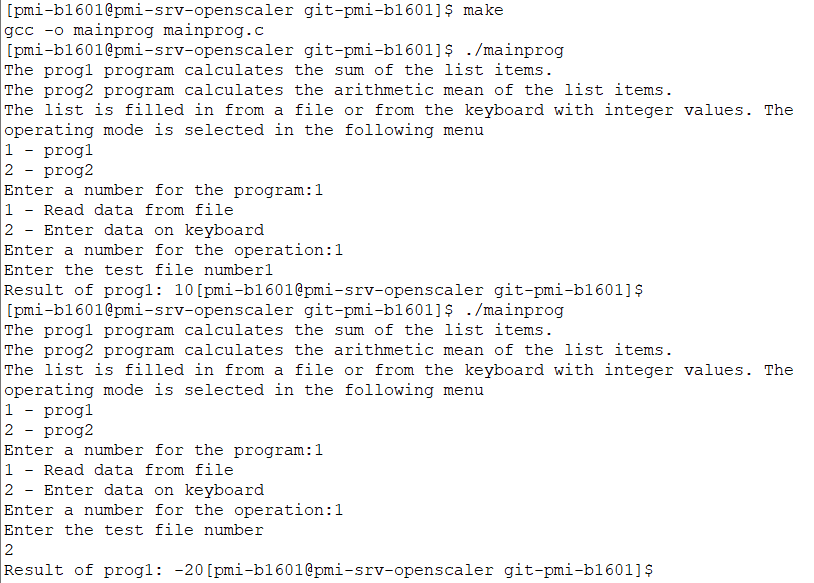
Если ошибок не обнаружено, то внесите в текст программы **mainprog** не менее двух синтаксических ошибок и повторно выполните компиляцию.

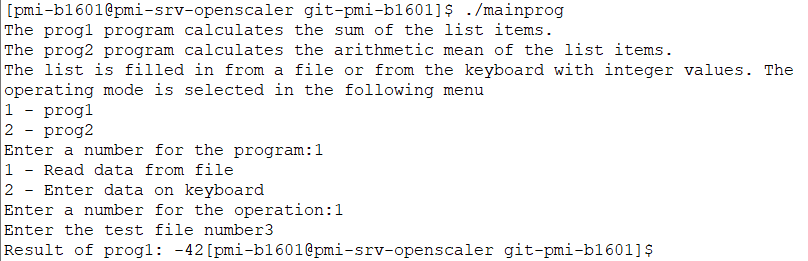


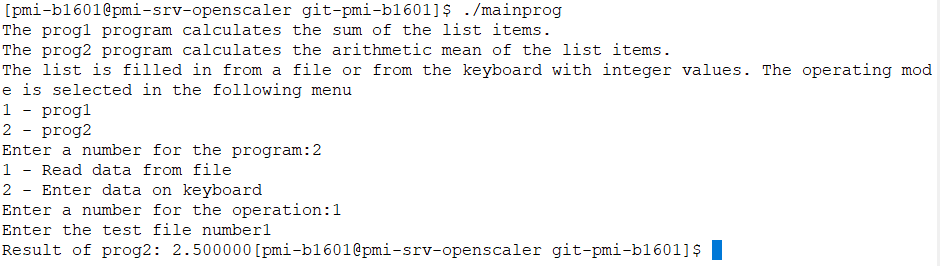
Добавили две ошибки: 5 строка вместо typedef написали typedf, 11 строка вместо list\* head написали lst\* head.

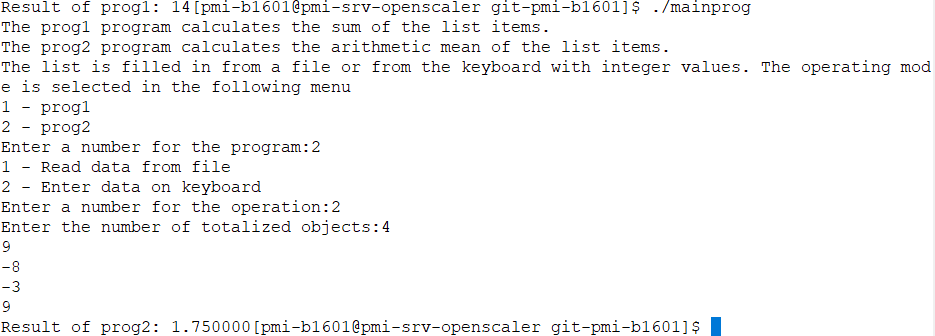
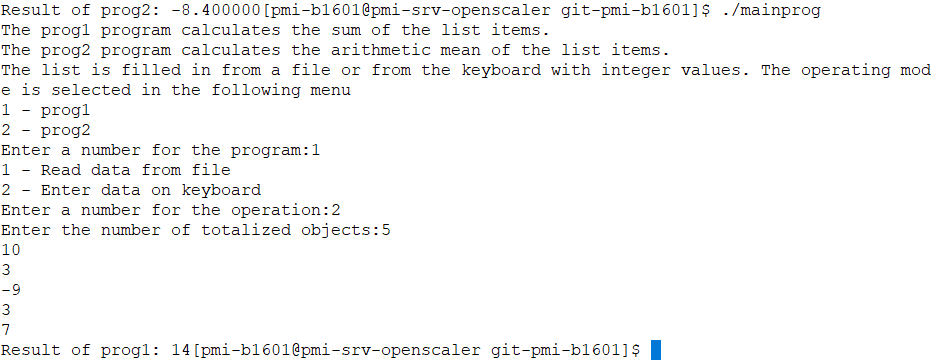
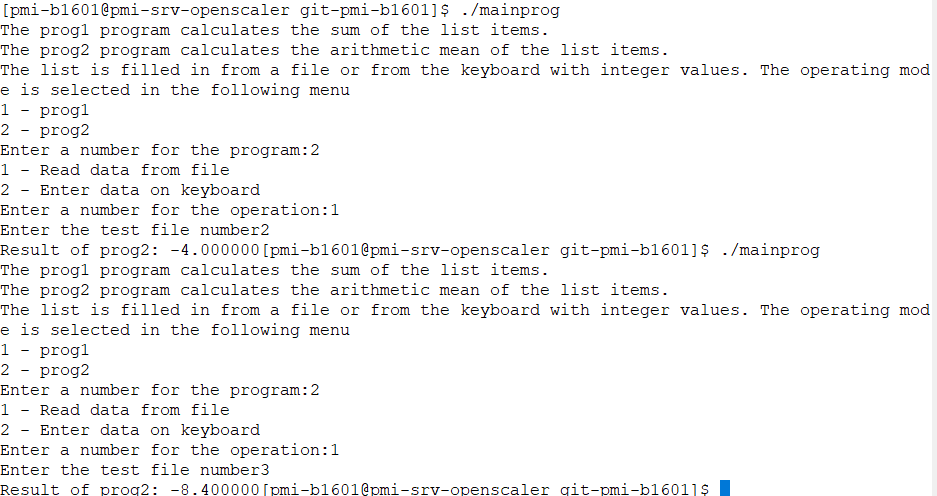


1. Запустите исполняемый файл **mainprog** и проверьте программу на подготовленных наборах тестовых данных, результаты тестирования анесите в отчет.





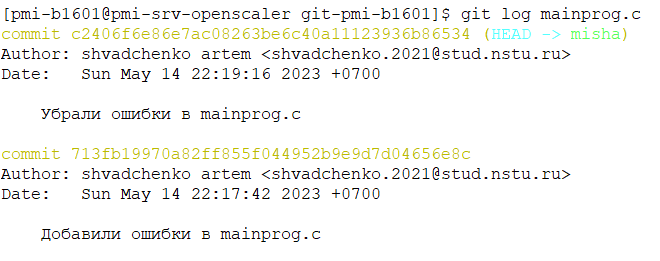




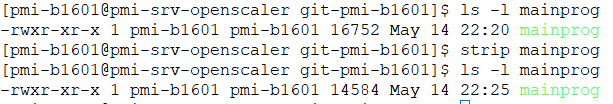
1. При обнаружении семантических ошибок перекомпилируйте программу с помощью правила ***testprog*** make-файла и запустите отладчик **gdb**, с помощью которого найдите причины появления ошибок. Проверьте программу на всех наборах тестовых данных. Сведения об ошибках занесите в отчет, исправленную версию программы сохраните в репозитории с поясняющим комментарием.

Семантических ошибок не обнаружено.

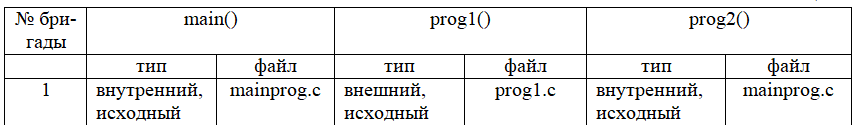
1. Выведите список всех изменений файла **mainprog.c**, выполненных в ходе отладки программы, занесите список в отчет.



1. Определите размер исполняемого модуля отлаженной программы. Удалите всю отладочную информацию и снова определите размер исполняемого модуля, сравните с предыдущим результатом, результат сравнения занесите в отчет.



1. Выполните разбиение программы **mainprog** на функции в соответствии с номером бригады из таблицы 2. Обратите внимание на тип функции (внутренняя или внешняя), тип файла (.c, .h или .o) и тип модуля (исходный или объектный). Занесите в отчет измененный текст программы.



#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

extern int sum\_list1();

extern int main1();

extern int read\_list1();

extern int enter\_list1();

typedef struct sNode {

int value;

struct sNode\* next;

}list;

list\* read\_list() {

list\* head = malloc(sizeof(list));

head->next = NULL;

list\* p = head;

int value;

FILE\* file;

int u;

printf("Enter the test file number");

scanf("%d", &u);

char\* file\_name = "";

if (u == 1)

{

file\_name = "elems1.txt\0";

}

else if (u == 2)

{

file\_name = "elems2.txt\0";

}

else if (u == 3) {

file\_name = "elems3.txt\0";

}

file = fopen(file\_name, "r");

while (fscanf(file, "%d", &value) != EOF) {

p->next = malloc(sizeof(list));

p = p->next;

p->value = value;

p->next = NULL;

}

fclose(file);

return head;

}

list\* enter\_list() {

list\* head = malloc(sizeof(list));

list\* p = head;

int n;

int value;

printf("Enter the number of totalized objects:");

scanf("%d", &n);

while (n > 0) {

scanf("%d", &value);

p->next = malloc(sizeof(list));

p = p->next;

p->value = value;

p->next = NULL;

n--;

}

return head;

}

/\*int sum\_list(list\* head) {

int result = 0;

list\* p = head->next;

while (p != NULL) {

result += p->value;

p = p->next;

}

return result;

}

\*/

float average\_list(list\* head) {

float sum = 0;

float count = 0;

list\* p = head->next;

while (p != NULL) {

sum += p->value;

count += 1;

p = p->next;

}

float result = sum / count;

return result;

}

/\*int sum() {

int option;

printf("1 - Read data from file\n2 - Enter data on keyboard\nEnter a number for the operation:");

scanf("%d", &option);

list\* lst;

if (option == 1) {

lst = read\_list();

}

else if (option == 2) {

lst = enter\_list();

}

else {

printf("No such operation!\n");

lst = malloc(sizeof(list));

lst->next = NULL;

}

int sum = sum\_list(lst);

return sum;

}\*/

float average() {

int option;

printf("1 - Read data from file\n2 - Enter data on keyboard\nEnter a number for the operation:");

scanf("%d", &option);

list\* lst;

if (option == 1) {

lst = read\_list();

}

else if (option == 2) {

lst = enter\_list();

}

else {

printf("No such operation!\n");

lst = malloc(sizeof(list));

lst->next = NULL;

}

float avrg = average\_list(lst);

return avrg;

}

int main() {

int option;

int result1;

float result2;

printf("The prog1 program calculates the sum of the list items. \n");

printf("The prog2 program calculates the arithmetic mean of the list items.\nThe list is filled in from a file or from the keyboard with integer values. The operating mode is selected in the following menu\n");

printf("1 - prog1\n2 - prog2\nEnter a number for the program:");

scanf("%d", &option);

if (option == 1) {

result1 = main1();

}

else if (option == 2) {

result2 = average();

printf("Result of prog2: %f", result2);

}

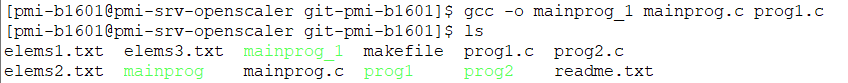
else {

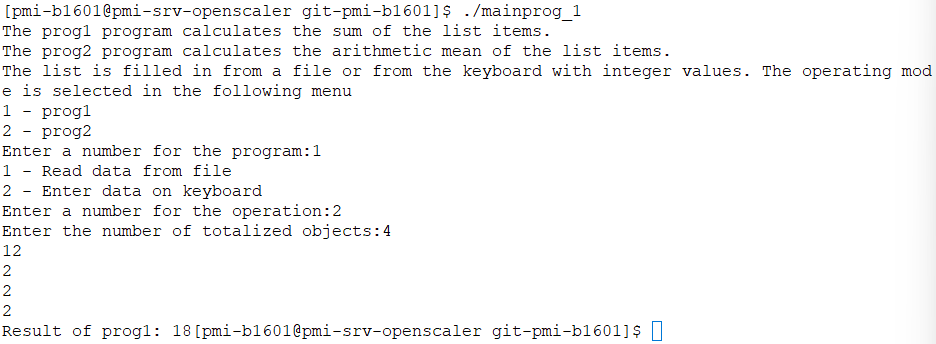
printf("No such function!\n");

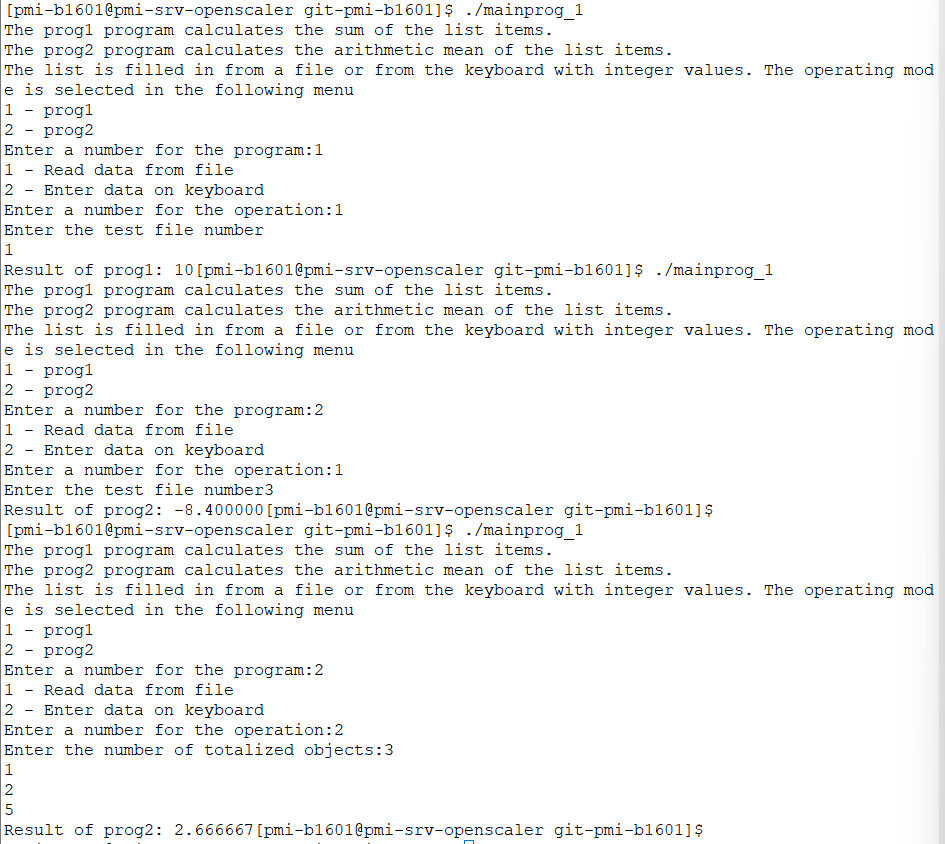
}

}

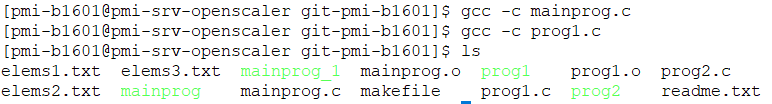
1. Выполните сборку программы в соответствии вариантом задания, используя неявный вызов компоновщика и задав имя исполняемого файла **mainprog\_1,** проверьте корректность работы программы и занесите в отчет результаты ее тестирования.

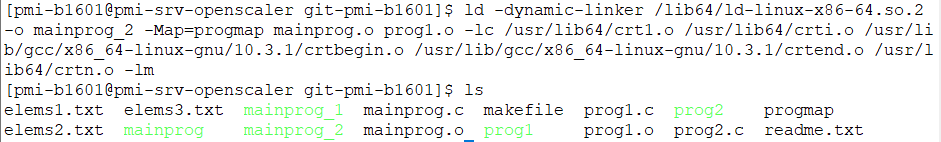


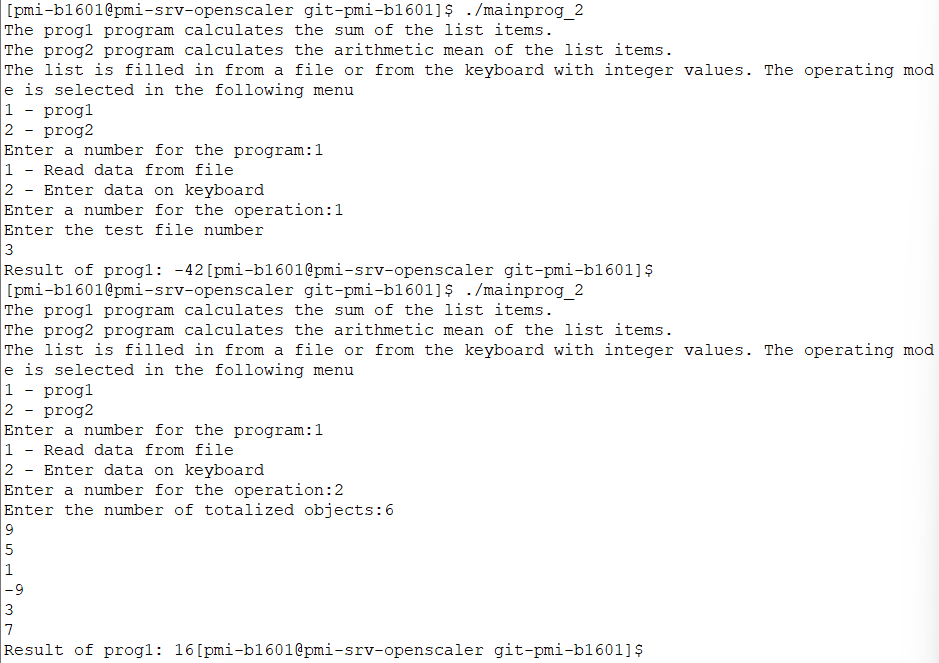


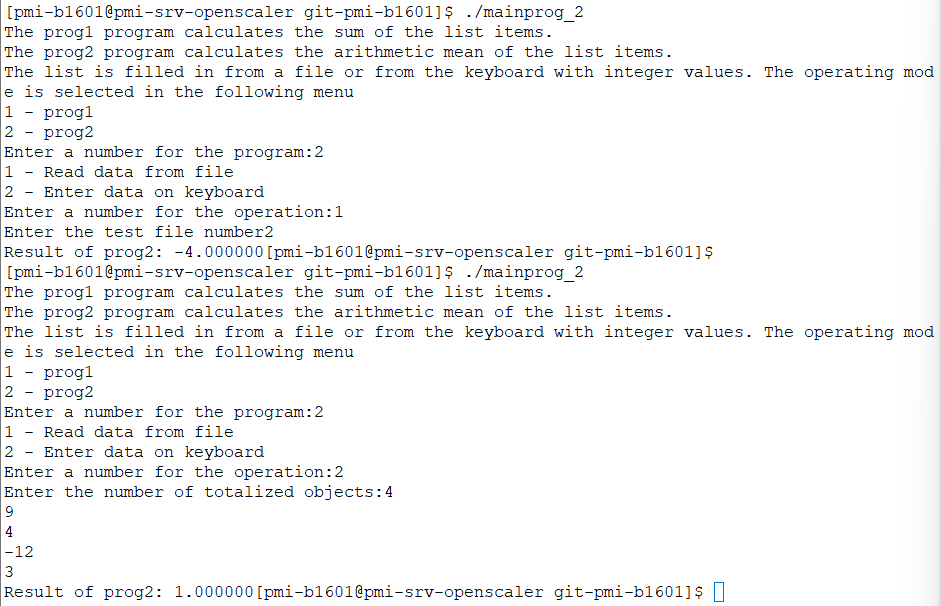


1. Выполните сборку программы в соответствии вариантом задания, используя явный вызов компоновщика. Результатом сборки должны быть исполняемый файл **mainprog\_2** и карта памяти **progmap;** проверьте корректность работы программы и занесите в отчет результаты ее тестирования.



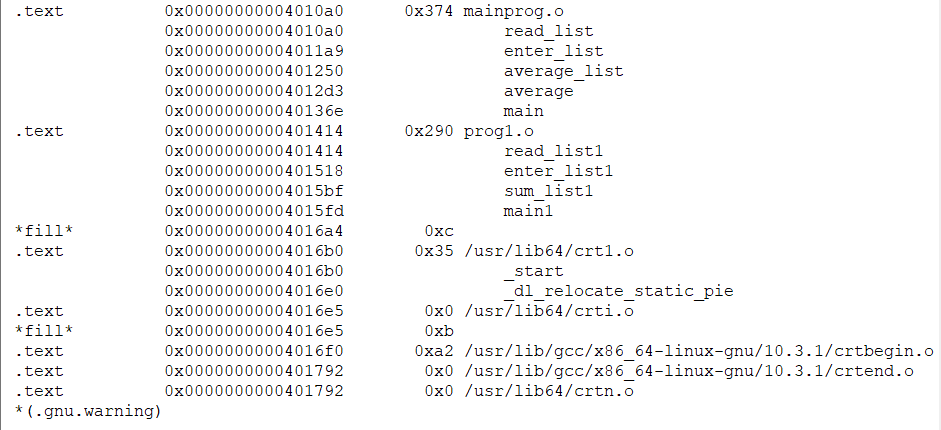


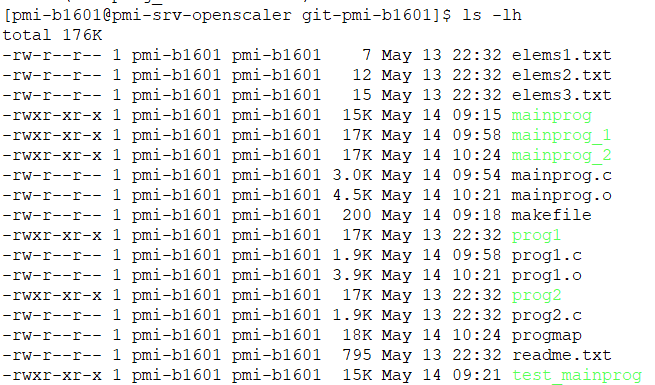




1. Из карты памяти **progmap** определите размеры машинного кода модулей **mainprog.o,** **prog1.o** и **prog2.o,** сравните их с размерами исходного и объектного кода этих модулей. Размеры файлов типа **.с** и **.о** определите с помощью команды ls. Результат представьте в виде таблицы 3, все данные должны быть подтверждены скриншотами.

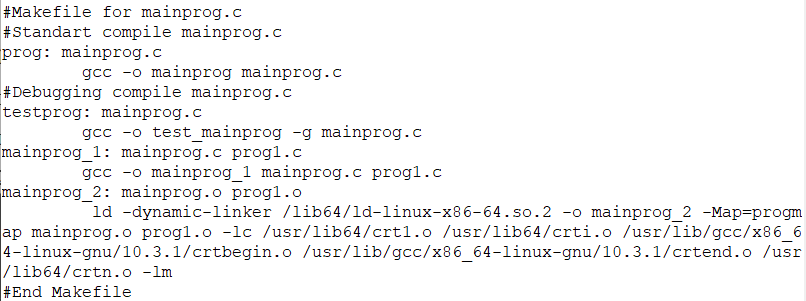




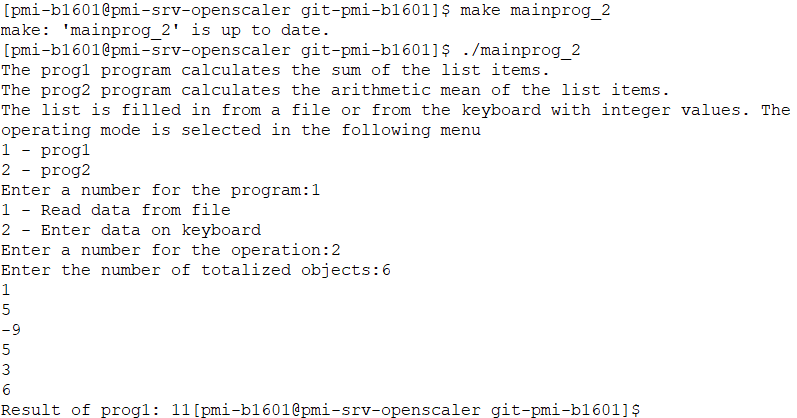


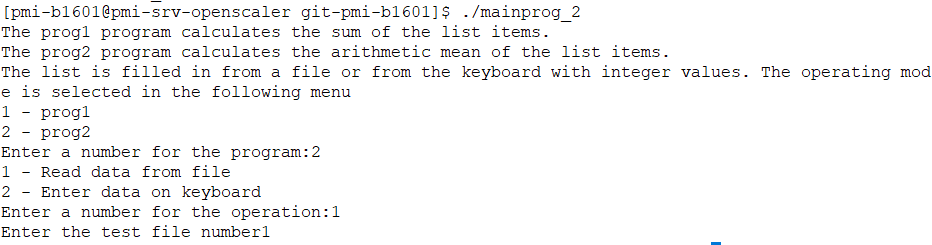
|  |  |  |  |
| --- | --- | --- | --- |
| Имя модуля (функции) | Исходный, Кбайт | Объектный, Кбайт | Машинный код, байт |
| prog1 | 1.9 | 3.9 | 656 |
| prog2 | 1.9 | - | - |
| mainprog | 3 | 4.5 | 884 |

1. Добавьте в make-файл, разработанный при выполнении п. 1, два новых правила, реализующие п. 8 и 9 задания. Проверьте корректность его работы и сохраните его в репозитории.

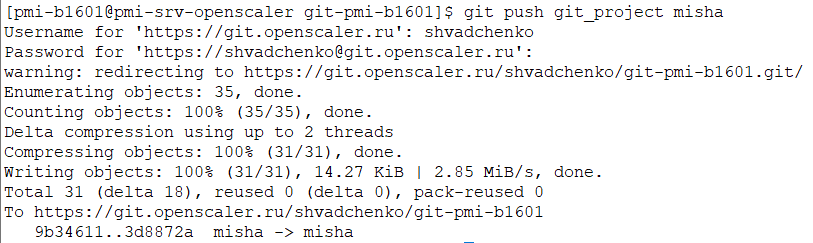


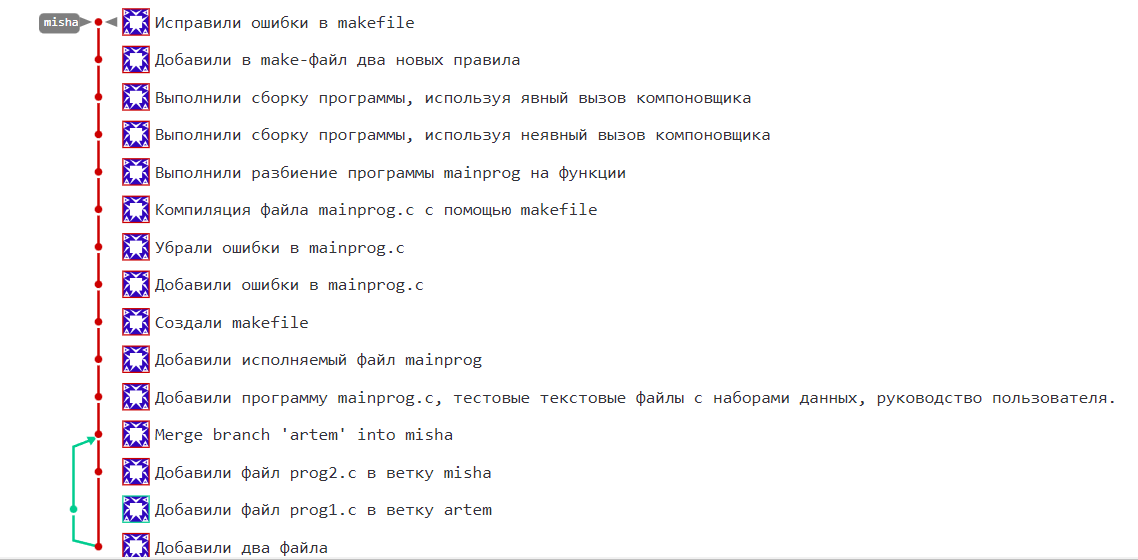






1. Выполните экспорт всех данных из локального репозитория в центральный репозиторий, адрес которого будет указан преподавателем.





1. В ходе выполнения лабораторной работы были получены практические навыки по использованию инструментальных средств фонда свободного программного обеспечения при компиляции исходного кода, сборке, отладке и тестировании программ, написанных на языке С. Проработали контрольные вопросы по теме.