Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра информатики

Отчет по лабораторной работе №6 Компиляция и запуск программы на C

Выполнил: студент гр. 053505

Харлап Никита Олегович

Руководитель: преподаватель

Удовин И.И.

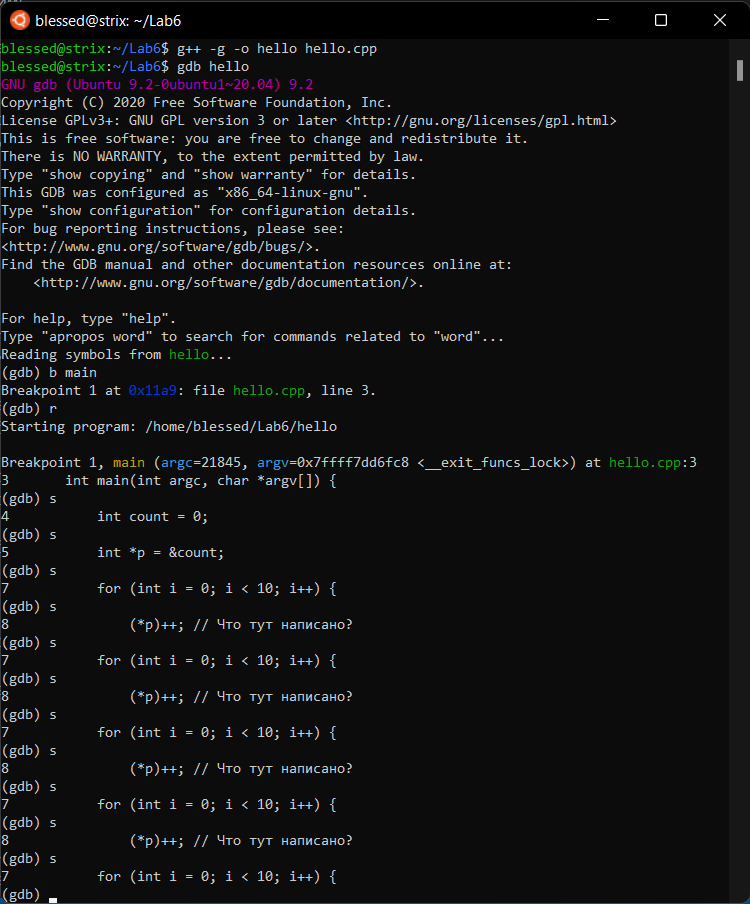
Минск 2021

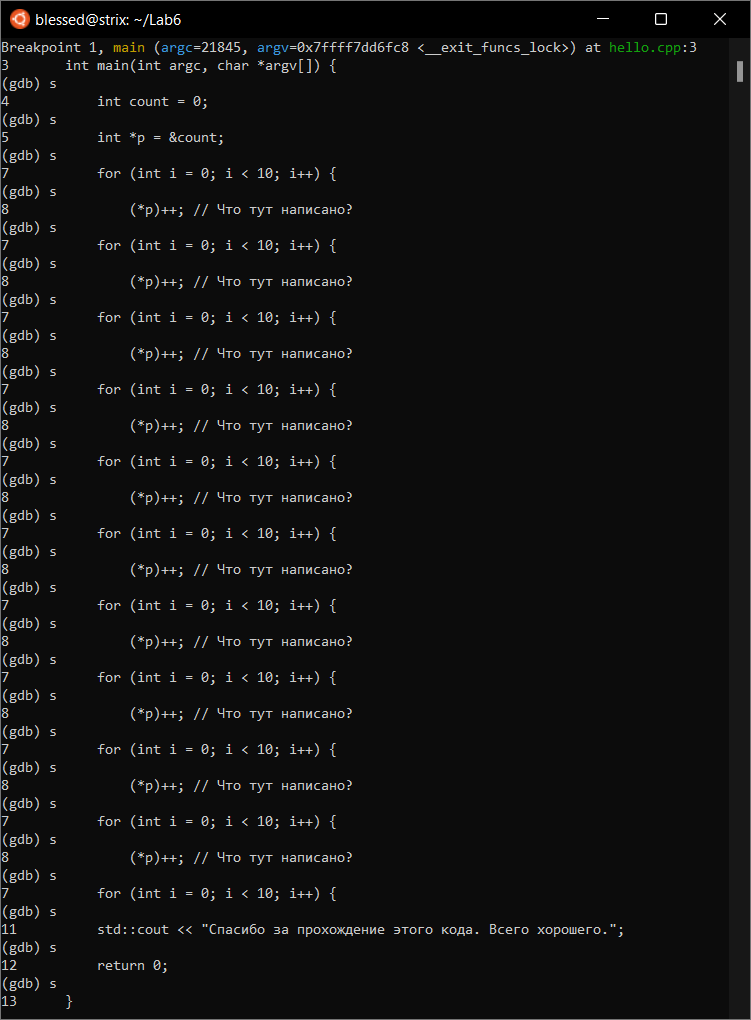
СОДЕРЖАНИЕ

1. Введение
2. Задания
3. Выводы Литература
4. Введение
5. Использование программы командной строки для компиляции программ на C.
6. Работа с GNU De-Bugger.
7. Valgrind / Sanitizer.
8. Компиляция с помощью Makefile.
9. Битовые операции.
10. Задания:

**ЗАДАНИЕ 1.1.** Скомпилируйте hello.cpp с флагом –g, запустите отладчик и пройдите через всю программу, выполнив следующие действия:

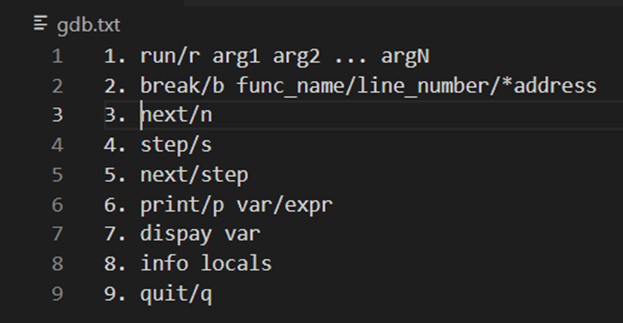
* + - 1. Установка точки останова на функции main
      2. Исполните команду run.
      3. Исполните команду stepep



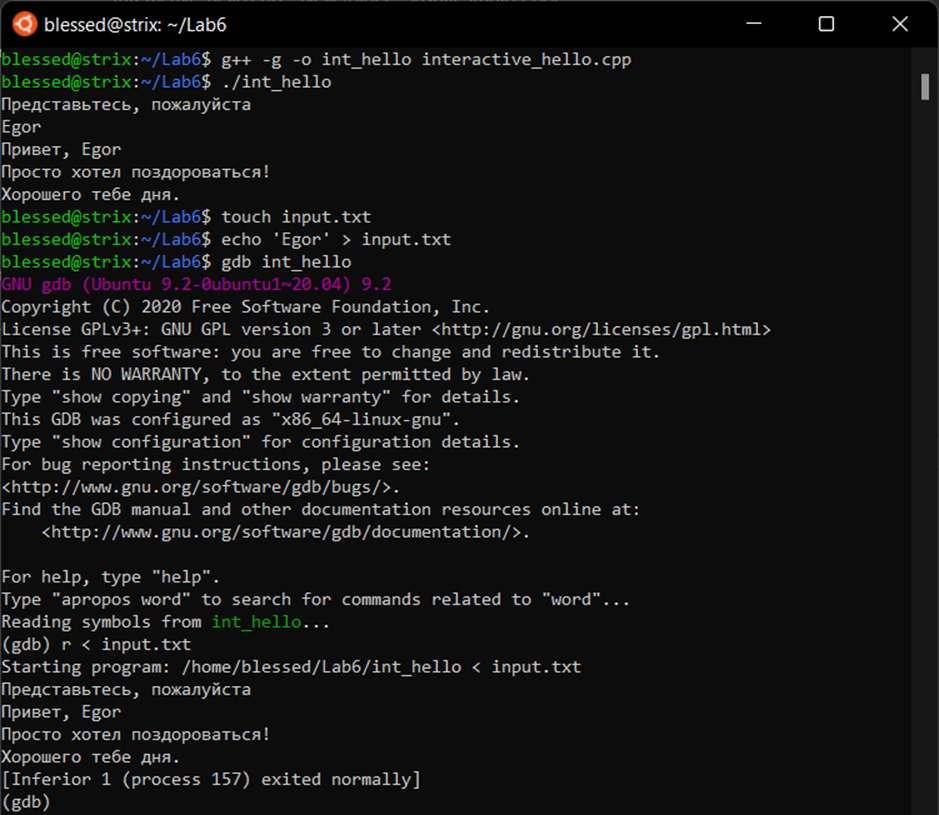


**ЗАДАНИЕ 1.2.** В текстовом файле с именем gdb.txt ответьте на следующие вопросы:

1. Когда вы находитесь в сеансе gdb, как вы устанавливаете аргументы, которые будут передаваться программе при ее запуске?
2. Как создать точку останова?
3. Как выполнить следующую строку кода C в программе после остановки в точке останова?
4. Если следующая строка кода является вызовом функции, вы выполните весь вызов функции сразу, если воспользуетесь своим ответом на # 3. (Если нет, рассмотрите другую команду для # 3!) Как вы сообщите GDB, что хотите вместо этого отладить код внутри функции (т.е. перейти внутрь функции)?
5. Как продолжить выполнение программы после остановки на точке останова?
6. Как вы можете распечатать значение переменной (или даже выражение типа 1 + 2) в gdb?
7. Как настроить gdb так, чтобы он отображал значение переменной после каждого шага?
8. Как вы показываете список всех переменных и их значений в текущей функции?
9. Как выйти из GDB?

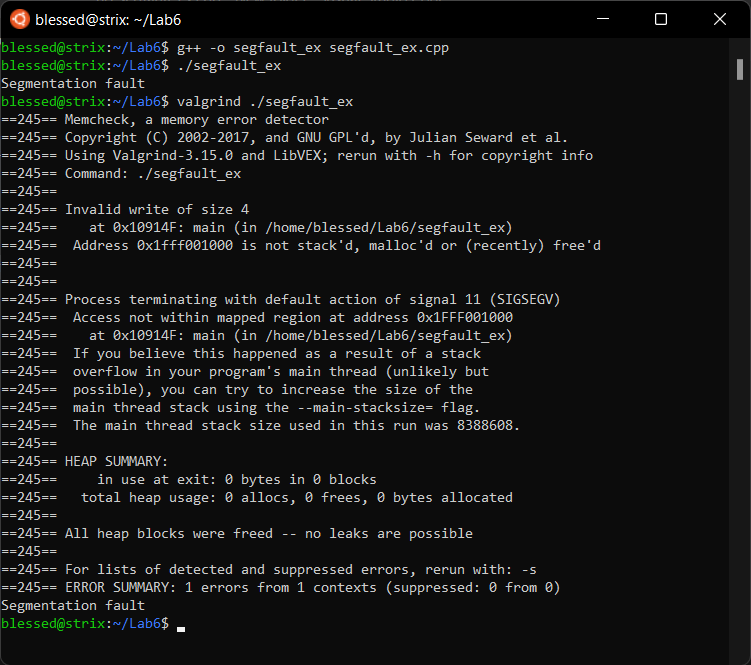


**ЗАДАНИЕ 2.** Собрать и запустить программу, определенную в interactive\_hello.cpp, затем отладить её.



**ЗАДАНИЕ 3.** Соберите два новых исполняемых файла, segfault\_ex из

segfault\_ex.cpp и no\_segfault\_ex из no\_segfault\_ex.cpp (используйте флаг -o).

Исправьте ошибки с помощью Valgrind или санитайзера.

Исправленный segfault\_ex.cpp:

int main()

{

int a[20];

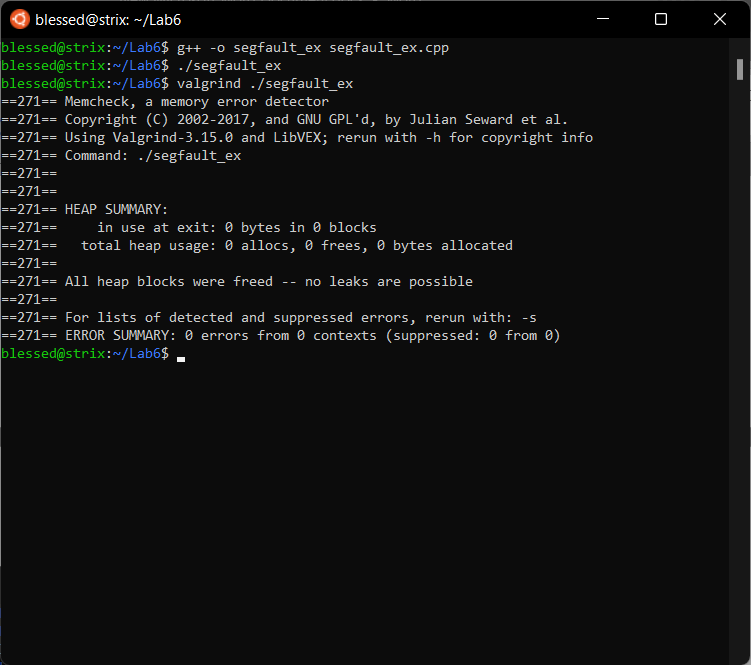
for (int i = 0; **i < 20**; i++)

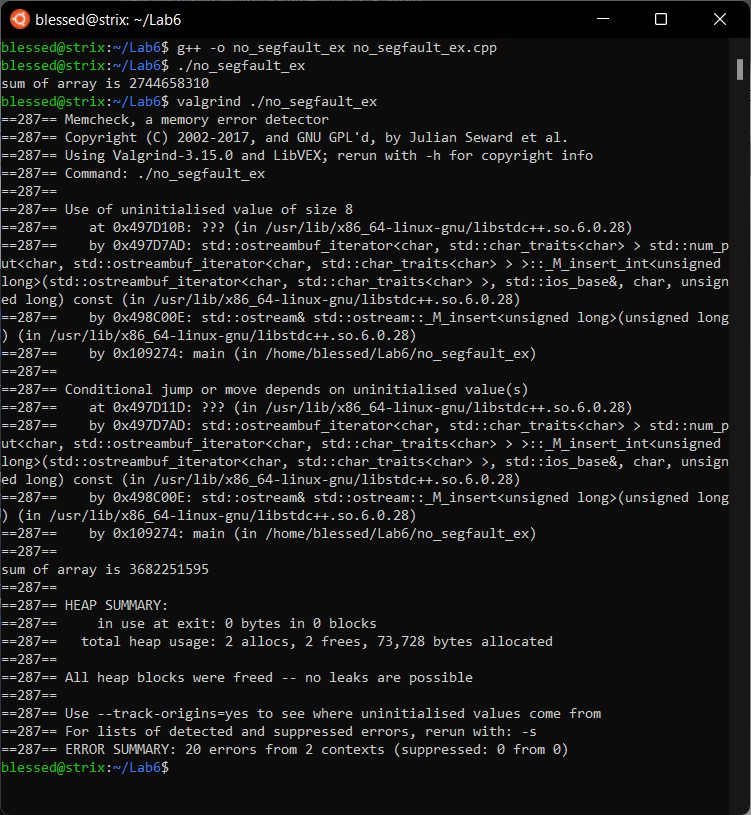
{

a[i] = i;

}

}





Исправленный no\_segfault\_ex.cpp: #include <iostream>

int main() {

int a[5] = {1, 2, 3, 4, 5};

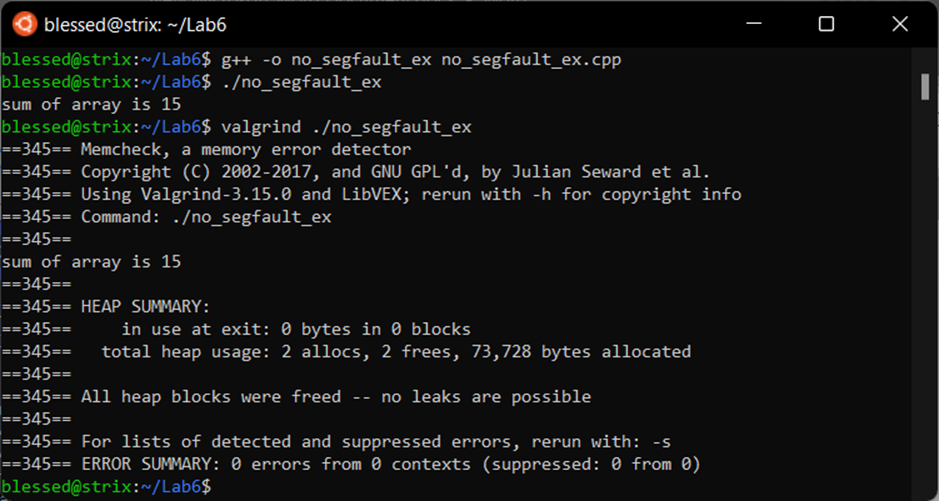
unsigned total = 0;

for (int j = 0; j < sizeof(a) **/ sizeof(int)**; j++) { total += a[j];

}

std::cout << "sum of array is " << total << std::endl;

}

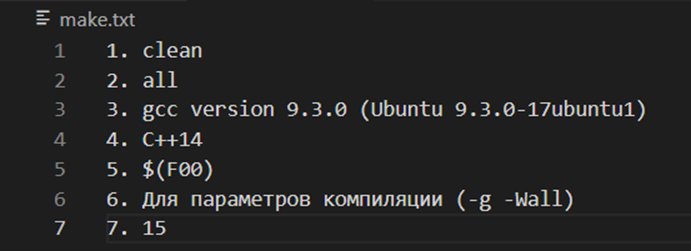


**ЗАДАНИЕ 4.** Взгляните на Makefile, включенный в эту лабораторную работу, и попытайтесь ответить на следующие вопросы в make.txt:

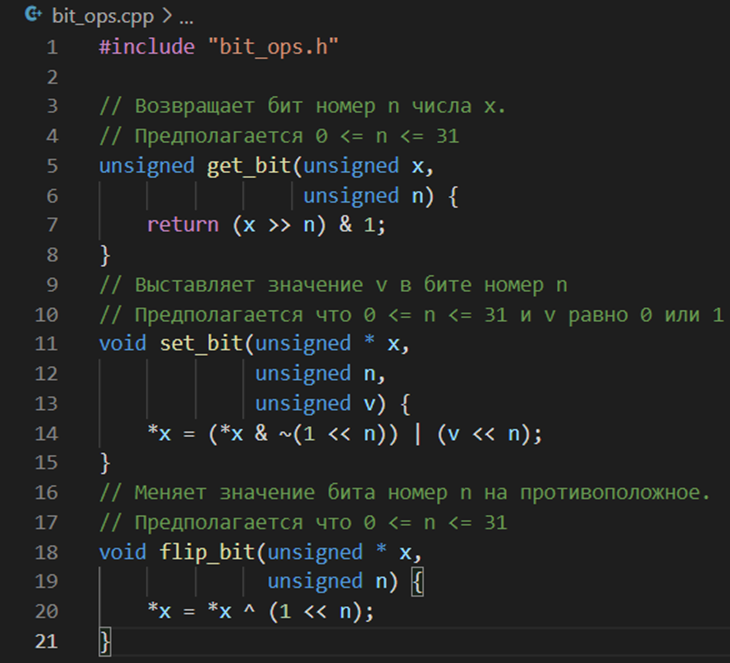
1. Какая цель является частью правила, удаляющего все скомпилированные программы?
2. Какая цель является частью правила, которое создает все скомпилированные

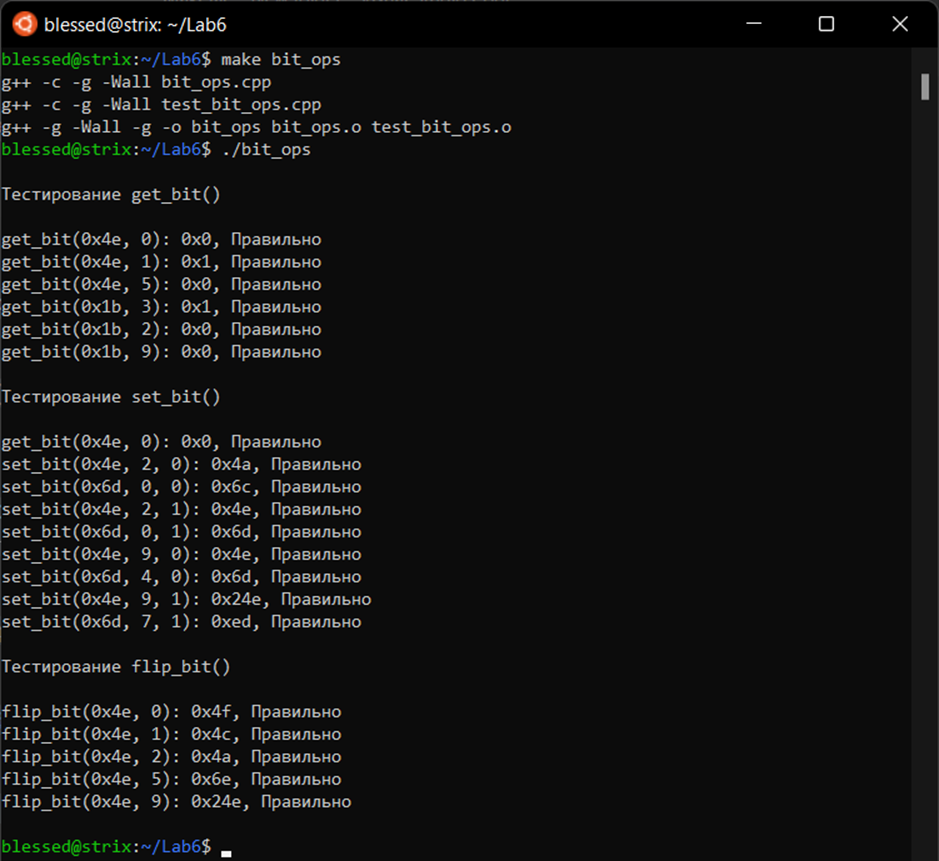
программы?

1. Какой компилятор сейчас используется?
2. Какой стандарт C мы используем в настоящее время?
3. Как мы можем ссылаться на переменную «FOO» в make-файле?
4. Для чего используется переменная CFLAGS?
5. Какая строка создает программу bit-ops из ее объектных файлов? (Укажите номер строки.)



**ЗАДАНИЕ 5.** Завершите реализацию get\_bit, set\_bit и flip\_bit в bit\_ops.cpp.





3. Выводы:

На практике было изучено и опробовано, в соответствии с поставленной задачей: использование программы командной строки для компиляции программ на C, работа с GNU De-Bugger, Valgrind / Sanitizer, компиляция с помощью Makefile, битовые операции.