Сборка C++ программ

Headers, make, makefiles

Задачи занятия

- Разобраться со особенностями сборки C++ проектов состоящих из нескольких единиц трансляции;
- Придумать алгоритм сборки таких проектов.
- Автоматизировать процесс сборки с помощью makefile.
- Попробовать объявлять, определять и использовать функции в разных единицах трансляции.

Придумаем задачу

Пусть есть библиотека с именем “1”. Она состоит из двух файлов 1.h и 1.cpp.

Мы пишем новую программу в main.cpp и хотим использовать библиотеку “1”.

Как собрать?

- Собрать единицу трансляции (1.h, 1.cpp), скомпилировать. Если она уже собрана и в исходных файлах ничего не менялось - пропустить шаг;
- Собрать (main.cpp, 1.h) и скомпилировать с теми же условиями.
- Слинковать два объектных файла 1.o и main.o и получить a.out

Makefile (всё ли хорошо?)

```
CPP_FLAGS=-Werror -Wall
1.o: 1.cpp 1.h
    g++ $(CPP_FLAGS) -c 1.cpp
main.o: main.cpp 1.o
    g++ $(CPP_FLAGS) -c main.cpp -o main.obj
main: 1.h main.o
    g++ ${CPP_FLAGS} 1.o main.o
     -o main
clean:
    rm -f main
```

Makefile (ошибки выделены)

```
CPP_FLAGS=-Werror -Wall
all: main
1.o: 1.cpp 1.h
    g++ $(CPP_FLAGS) -c 1.cpp
main.o: main.cpp 1.o
    g++ $(CPP_FLAGS) -c main.cpp -o main.obj
main: 1.h main.o
    g++ ${CPP_FLAGS} 1.o main.o \
     -o main
clean:
    rm -f main *.o
.PHONY: clean all
```

Makefile

```
CPP_FLAGS=-Werror -Wall
all: main
1.o: 1.cpp 1.h
    g++ $(CPP_FLAGS) -c 1.cpp
main.o: main.cpp 1.h
    g++ $(CPP_FLAGS) -c main.cpp -o main.o
main: 1.o main.o
    g++ ${CPP_FLAGS} 1.o main.o \
-o main
clean:
    rm -f main *.o
.PHONY: clean all
```

Общие условия для всех задач

- Все файлы одной задачи лежат в отдельной папке
- В каждой папке есть рабочий Makefile
- Флаги компиляции -Wall -Werror прописаны макросом в каждом Makefile
- Файлы *.cpp и *.h лежат в папке src. Все бинарные артефакты кладутся в папку bin.

Задача 1

Не использовать заголовочные файлы, не включать в main.cpp файл add.cpp директивой #include.

add.cpp

определение функции int add(int, int)

main.cpp

вывод add(10, 20)

Задача 2

Решить задачу 1 используя заголовочные файлы.

Задача 3

a.h

объявление константы A1

определение константы A2 = 20

a.cpp

определение A1 = 10

main.cpp

вывод add(A1, A2)

Задача 4

a.o

объявление и определение переменной global_a

присваивание global_a значения add(A1, A2)

main.cpp

вывод global_a

Задача 5

Заменить global_a из предыдущего задания на функцию (трюк с inline)

Задача 6

Показать 2 способа задания переменной с внутренней компоновкой.

Задача 7. Happy debugging

Создать символ препроцессора HAPPY_DEBUG, при активации которого программа начинает вести себя непредсказуемо. Например add(10, 20) возвращает 31, а код функции не менялся.

Подсказки:

-DHAPPY_DEBUG

-E name.cpp (preprocessor output)

Задача 8. Link C & C++

Определить функцию int sum(int, int) в sum.c, которая выводит сумму.

Вызвать эту функцию из main.cpp. Компилировать необходимо соответствующими компиляторами в отдельные объектные файлы.

(повтор задания с первого семинара + makefile)