Програмни Езици, КСТ, бак, V сем.

Лабораторни упражнения

З А Д А Ч А N 4

Построяване на многомодулен многоезиков (C и C++) проект в конзола

Цел : Да се състави многомодулен C и C++ проект. Да се усвоят правилата и препоръките за оформяне на многоезикови декларации в хедърните модули, вкл. работата с директиви на препроцесора за условно транслиране. Да се установят факти относно правилата за декориране на имената при двата езика. Да се направят изводи за възможностите програмен код на всеки от двата езика да използва код на другия език.

Изисквания : Упражнението се изпълнява въз основа на програмния проект от лаб.упр. No.3, изцяло съобразно изградената там модулна структура.

Задачи : Да се изпълни следната поредица от действия :

--- Програмно решение ----------

Да се използва програмното решение на задачата от лаб.упр. No.3 !

За целите на разглеждането се приема, че : модулът с дефиниции на глобалните структури от данни е именован <*data*>.*c* ; модулът с дефиницията на функция main е <*main*>.*c* ; модулът с декларациите на глобалните стуктури от данни е <*data*>.*h* ; модулът с дефиницията на някоя от другите функции е с условно име <*function*>.*c* ; модулът с декларация на тази функция е с условно име <*function*>.*h* .

--- Транслиране ----------

Тъй като предстои да се извършва транслиране на изходни модули на C (.*c*) и на C++ (.*cpp*), то следва да се отбележи, че при тази среда за разработка (транслатор и свързващ редактор от MS Visual Studio) двата транслатора (на C и C++) са обединени в един инструмент. Поради това транслирането в двата случая се извършва по един и същи начин, а инструментът се ориентира за какъв език става дума в модула по разширението му :

Hint: *cl /c /W3 <file_source_c>.c*

Hint: *cl /c /W3 <file_source_cpp>.cpp*

--- Свързване ----------

Свързването се извършва по същия начин, както в лаб.упр. No.3, тъй като след транслиране в обектните модули няма следи от изходния език и няма значение, че различните модули са били на различни езици :

Hint: *link <file_source_1>.obj <file_source_2>.obj ... <file_source_N>.obj*

--- Конфигурации ----------

Съобразно поставените в упражнението цели, следва да се изготвят и построят 8 (осем) варианта на програмния проект, като за всеки един от тях следва да се установи :

- Как точно трябва да изглежда декларацията на всеки глобален идентификатор в съответния хедърен модул ?
- Как точно изглежда всеки декориран глобален идентификатор в транслирания модул, където се намира дефиницията на идентификатора ?
- Как точно изглежда всеки декориран глобален идентификатор във всеки транслиран модул, където се намира използване на идентификатора ;

– Ако вариантът е възможен за построяване – защо построяването е успешно, а ако е невъзможен – защо това е невъзможно ?

За анализ на множеството глобални имена в даден изходен модул и тяхното декориране при транслиране, може да се използва инструмента *dumpbin*, който се прилага върху съответния обектен модул, получен след транслацията на изходния.

Hint: *dumpbin /SYMBOLS <file_source>.obj*

Варианти :

1. Всички source модули – на C ;
2. Всички source модули – на C++ ;
3. *main.cpp* – на C++, *data.c* – на C, всички функции <*function*>.*cpp* – на C++ ;
4. *main.cpp* – на C++, *data.c* – на C, всички функции <*function*>.*c* – на C ;
5. *main.cpp* – на C++, *data.cpp* – на C++, всички функции <*function*>.*c* – на C ;
6. *main.c* – на C, *data.cpp* – на C++, всички функции <*function*>.*c* – на C ;
7. *main.c* – на C, *data.c* – на C, всички функции <*function*>.*cpp* – на C++ ;
8. *main.c* – на C, *data.cpp* – на C++, всички функции <*function*>.*cpp* – на C++ ;

Hint: За да не настъпи тотален и всепоглъщащ хаос още с преминаване от 3-ти към 4-ти вариант, горещо се препоръчва всеки от вариантите да се оформи като самостоятелен и независим от другите проект – в отделна под-директория на работната. При такава организация ще се окаже, че *.bat* файловете за подготовка на конзолата и за почистване на построеното ще са еднакви за всички варианти и могат да присъстват с по един екземпляр в основната работна директория, а *.bat* файловете за построяване ще са специфични за всеки вариант и съответно ще присъстват в съответната под-директория на варианта.

!!! При разработване на всеки един от вариантите и особено при анализа на проекта относно декорирането на имената и възможността да бъде построен или не, да се отчита простият и съществен, но като цяло неглижиран факт, че при транслиране на даден модул се приема, че целият (!!!) съставящ го текст е на такъв език (C/C++), за какъвто го счита транслатора. Последното се решава от разширението на файла, представляващ подложения на транслиране модул (*.c*/*.cpp*) ! Това важи с пълна сила и за текст от хедърен модул (*.h*), вмъкнат с директива на препроцесора #include. Особено внимание изисква фактът, че един и същи хедърен модул сменя езиковия си контекст при вмъкване в модули на различни езици. Това налага изграждане на „обвивка“ от директиви на препроцесора за условно транслиране, според езика на текущата трянслация.

!!! Изключително внимателно (!!!) следва да се преценява кои декларации да бъдат „обвивани“ с директиви за условно транслиране. В случая на съвместно използване на езиците C и C++, такава „грижа“ изискват идентификаторите, чиито дефиниции се извършват на C. Защо ? За идентификатори с дефиниции на C++ няма как да се положи каквато и да било „грижа“ ! Защо ?

--- Изводи ----------

!!! В резултат на извършената работа по 8-те варианта на проекта да се направят изводи относно :

– как декларациите играят ролята освен на синхронизиращ механизъм за осигуряване на съответствие между дефиниция и използване на един идентификатор, също така и в ролята на съвместяващ механизъм за правилно декориране на идентификаторите ?
– как чрез директивите на препроцесора за условно транслиране става възможно в един и същи хедърен файл да се поместват версии на една и съща декларация, но на различни езици ?
– какви са възможните (допустими) варианти в текст на някой от двата езика (C/C++) да се използват идентификатори, чиито дефиниции са направени на другия език (C++/C) ?

Съдържание на протокола : списък на файловете в работната директория и под-директориите на вариантите на проекта; листинг на изходния код на хедърните файлове за всеки от вариантите на проекта; оформен таблично списък от декорираните глобални идентификатори за всеки транслиран модул на всеки от вариантите на проекта с изрично отбелязани проблемни несъответствия на декорации при дефиниране и използване на имената; отговори и коментари на въпросите от секция Изводи.

Източници :

Desolation
https://www.youtube.com/watch?v=CJV-RT4Y54o

extern (C++)
https://docs.microsoft.com/en-us/cpp/cpp/extern-cpp?view=msvc-160

Linking C and C++ code
https://en.wikipedia.org/wiki/Compatibility_of_C_and_C%2B%2B#Linking_C_and_C++_code

Design time
https://www.youtube.com/watch?v=4oihRkIu_zs

Preprocessor directives
https://docs.microsoft.com/bg-bg/cpp/preprocessor/preprocessor-directives?view=msvc-160

#if, #elif, #else, and #endif directives
https://learn.microsoft.com/en-us/cpp/preprocessor/hash-if-hash-elif-hash-else-and-hash-endif-directives-c-cpp?view=msvc-160

Predefined macros (__cplusplus)
https://docs.microsoft.com/en-us/cpp/preprocessor/predefined-macros?view=msvc-160

Code workout
https://www.youtube.com/watch?v=8v4oxxvs8iY

DUMPBIN Reference
https://learn.microsoft.com/en-us/cpp/build/reference/dumpbin-reference?view=msvc-170

DUMPBIN options - /SYMBOLS
https://learn.microsoft.com/en-us/cpp/build/reference/symbols?view=msvc-170

Anticipation
https://www.youtube.com/watch?v=xV_LnAhPPGI