Напредни Це

Осмо вежбање

Вежба 1

Модуларност програма се постиже груписањем сродних функционалности и података у засебне и независне модуле. Сваки модул се у општем случају састоји од 2 главне целине: **програмске спреге** (представљене у заглављу *.h*) и **имплементације** (смештене у *.c* датотекама).

Заглавље модула представља програмску спрегу апликације (познату и као *Application Programming Interface – API*) у којој су смештене ***искључиво информације релевантне за корисника*** тј. програмера који користи модул. У те информације спадају пре свега прототипи функција које корисник може да позива из тог модула, декларације јавних променљивих, али и макро дефиниције којима је могуће мењати конфигурацију модула.

Са друге стране, у једној или више *.c* датотека су смештене ***одговарајуће имплементације функција*** чији прототипи су представљени у заглављу, као и све помоћне функције, променљиве и макро дефиниције уско везане за саму имплементацију. Све оно што није релевантно за корисника, већ за саму имплементацију треба да се налази у овим датотекама.

Размислити и навести бар 5 већ готових модула/библиотека које сте до сада користили, било у Це или Це++ језику. Пронаћи и погледати програмске спреге ових модула.

Вежба 2

На основу изложеног у претходној вежби, потребно је преуредити датотеку „cipher.c“ тако да се одвоје функционалности „ceasar“ и „rot13“ у засебне модуле:

1. Направити нови Це пројекат „cipher“ и у њега додати датотеку „exercises/cipher.c“
2. Превести и потом покренути пројекат.
3. Анализирати и прекопирати излаз програма у привремену датотеку.
4. Све функционалности, променљиве и претпроцесорске симболе уско везане за „ceasar“ кодовање одвојити у засебне датотеке ceasar.h и ceasar.c (датотеке треба да се налазе у истом пројекту).

5. Превести и потом покренути пројекат.
6. Упоредити излаз програма са излазом из тачке 3. Излази морају бити идентични.
7. Све функционалности, променљиве и претпроцесорске симболе уско везане за „rot13“ кодовање одвојити у засебне датотеке rot13.h и rot13.c (датотеке треба да се налазе у истом пројекту)
8. Поновити тачке 5. и 6. Излази опет морају бити идентични.

Вежба 3

У овој вежби ће претходно раздвојене датотеке бити додате у независне статичке библиотеке и биће приказан поступак превођења истих.

1. Направите нови C пројекат „ceasar11” и за тип пројекта одаберите Static Library као тип пројекта.
2. Претходно направљене датотеке ceasar.c и ceasar.h ископирати у овај креирани „ceasar11” пројекат и такође их обрисати из „cipher“ пројекта.
3. Избилдујте библиотеку.
4. Направите статичку библиотеку „rot11“ коришћењем претходно направљених датотека rot13.h и rot13.c и такође ове датотеке обрисати из „cipher“ пројекта.
5. Избилдујте библиотеку.
6. Пројекат „cipher“ повезати са направљеним статичким библиотекама „ceasar11” и „rot11“.
7. Избилдујте и покрените програм.

Напомена: У одређеним корацима вежбе ће бити пријављене грешке приликом превођења и/или повезивања. Потребно их је исправити додајући одговарајуће опције за компајлер и/или линкер.

Опције за компајлер се подешавају у оквиру сваког пројекта засебно: *Project → Properties → C/C++ Build → Settings → GCC C Compiler.*

Напредни Це

Опције за линкер се подешавају у оквиру *Executable* пројекта: *Project → Properties → C/C++ Build → Settings → GCC C Linker.*

Вежба 4

У овој вежби је потребно извршити превођење и повезивање динамичких библиотека:

1. Направите нови C пројекат „ceasar12” и за тип пројекта одаберите Shared Library као тип пројекта
2. Додајте ceasar.c и ceasar.h у претходно креирани пројекат
3. Избилдујте библиотеку
4. Поновите претходна 3 корака и направите динамичку библиотеку „rot12“ коришћењем датотека rot13.h и rot13.c.
5. Направите нови C пројекат и за тип пројекта одаберите Executable
6. Додајте cipher.c у претходно креирани пројекат
7. У Build/Run конфигурације додајте нову променљиву окружења (environment variable) LD_LIBRARY_PATH. Вредност ове променљиве треба да садржи путање до претходно креираних динамичких библиотека (путање се раздвајају или са ':' или ';' без додатних размака).
8. Избилдујте и покрените програм

Напомена: Као и у претходној вежби, све пријављене грешке у превођењу и повезивању потребно је исправити додајући одговарајуће компајлер/линкер опције.

Приликом билдовања rot12 динамичке библиотеке обратите пажњу на зависност између те библиотеке и ceasar12 библиотеке.

Вежба 5

RT-RK
COMPUTER BASED SYSTEMS

Напредни Це

У овој вежби је потребно имплементирати динамичко повезивање током извршења програма. Неопходно је написати програм који ће вршити отварање “ceasar12” динамичке библиотеке, добављање адресе симбола функција за криптовање и декриптовање (*ceasarEncrypt* и *ceasarDecrypt*), извршавање одговарајућих претходно добављених функција, приказ резултата и затварање динамичке библиотеке. Користити динамичку библиотеку добијену у претходној вежби.

Напомена: Постоји могућност и да је сама библиотека за динамичко повезивање дељена. То се једноставно сазна приликом превођења, ако се јави грешка да нису дефинисане функције из ове библиотеке, потребно је додати *dl* име библиотеке у оквиру линкер опције *–l*.

Вежба 6

Обратити пажњу на места у коду на којима се користи макро функција *assert*. Овај марко је изузетно користан приликом дебаговања, јер проверава одговарајуће претпоставке које је програмер направио приликом имплементације својих функција. Неке од најчешћих претпоставки се односе на улазне аргументе функције, нпр. провера да су вредности показивача дефинисани на валидне меморијске локације којима се функцијом приступа (зашто је ово битно?).

Са развојем програма се увећава и број провера, па се тиме знатно успорава извршавање самог програма. Зато се све *assert* функије анулирају након што је развој програма завршен и када је програм тј. уређај спреман да се нађе на тржишту (тзв. *Release* верзија).

Одрадити следеће кораке:

1. Пронаћи спецификацију *assert* макроа у *C99* стандарду и разумети његово коришћење (како га је могуће „искључити“ и „укључити“).
2. Проверити да ли се спецификација слаже са имплементацијом макроа који се налази у *assert.h*.
3. Коришћењем компајлерске опције за дефинисање симбола (-D) „искључити“ тј. анулирати све позиве *assert* марко функције.

Напредни Це

Вежба 7

Поред assert позива, било какви друге помоћне методе за дебаговање коришћене током развоја морају бити искључене у финалној верзији кода. Једне од најчешћих таквих метода су свакако исписи на екран.

Датотеку *cipher.c* изменити тако да сви позиви стандардних улазно-изланих функција за испис на екран/терминал буду анулирани тј. избачени на исти начин као и *assert* позиви у претходној вежби (користити исти макро симбол).