T-201-GSKI, GAGNASKIPAN

Sumar 2016

COMPUTER GAME PROGRAMMING - HANGMAN

Handing in the assignment. Students hand in a single .zip archive containing all their code, .cpp and .h files, and storage files on **myschool**. Do **not** hand in binaries or object files ('bin' and 'obj' directories or files). If anything other than "gcc *.cpp" is needed to build, provide instructions or a makefile.
This assignment is **NOT** returned in Mooshak. Points may still be deducted for

- redundant or repeated code,
- messy code,
- code that is difficult to understand and not explained in comments.

HANGMAN

Hangman is a game of words, where one player knows a word while the other player (or group of players) guesses what it is. They may guess single letters or the entire word. For each wrong guess a part of body or gallow is added to an image and the game is lost when the entire hanged man has been drawn. The game is won if the correct word or all its letters are guessed correctly.

Your assignment is to program a text based version of hangman. As it is difficult to draw the image your program will instead inform the player how many guesses they have left before losing the game. The computer will always play as the game master, who knows the correct word, and the user will always play as the guesser. You can use **any data structures in the C++ standard template library** and structures made by yourself, co-students or teachers in previous assignments.

The program should have access to a text file, containing a "word bank" to choose the words from. It should be possible to add words to the file and remove them with the program still running correctly. It is recommended that you read this word bank into a data structure when the program is started to have quick access to it throughout the run. Each time a new game is started a word should be randomly selected from the word bank. Then, before each guess, the program outputs the word, where each letter that hasn't been guessed is a dash ('-'), while each letter that has been guessed is in its correct place. In the beginning the

word will then simply be a line of dashes. The program also outputs the number of guesses left before the game is lost.
Once a game is won or lost, the program should let the user know and offer him the possibility of either quitting or playing another game.

You can implement the following parts in any order you see fit, but it is recommended to finish each group before proceeding to the next. Each part is graded on both **functionality** and **program structure**.

1. BASIC GAME - 50%

- Display word as dashes, e.g. **structure** = --------- … - **10%**
- User can guess a character and is notified whether the word contains it or not - **10%**
- Display guessed characters, e.g. **structure** = **-tr--t-r-** after guessing 'r' & 't' - **10%**
- Display number of guesses left each turn - **5%**
- Detect loss when guesses are finished - **5%**
- Detect victory when word is complete - **10%**

2. MORE REFINED SINGLE GAME - 30%

- User can input or select number of guesses before the game begins - **5%**
- After finishing a game the user can select to quit or start new game - **5%**
- Program stores a word bank in a data structure - **10%**
- Program randomly selects word from word bank - **5%**
- The word bank is stored in and read from a file - **5%**

3. CONNECTED SERIES OF GAMES - 20%

- Keep track of wins and losses throughout the run (store in classes and variables) - **5%**
- Allow user to guess whole word - **5%**
- Find a way to score series of games and keep track of high scores - **5%**
    - Scores stored in file so they live between runs of program and are possibly affected by:
        - # of wrong guesses per word
        - # of total guesses per word (fewer if can guess whole word)
        - # of games before loss (or total score of those games), etc.
- Allow words to be added to the word bank (and file) through the program itself - **5%**

HENGIMANN

Hengimann er orðaleikur þar sem einn leikmaður veit hvert orðið er en annar leikmaður (eða hópur leikmanna) giskar á hvað það er. Þeir geta giskað á stakan staf eða orðið allt. Fyrir hvert rangt gisk er teiknaður hluti af líkama eða gálga og leikurinn tapast ef allur hengdi maðurinn hefur verið teiknaður. Leikurinn er unninn ef giskað er á rétta orðið eða alla stafina úr því.

Ykkar verkefni er að forrita textaútgáfu af hengimanni. Þar sem erfitt getur reynst að teikna myndina í textaham mun forritið ykkar þess í stað upplýsa leikmanninn um fjölda ákiskana sem hann á eftir áður en hann tapar leiknum. Tölvan mun alltaf spila sem leikstjórnandinn, sem ákveður og þekkir orðið, og leikmaðurinn mun alltaf spila giskandann. Þið megið nota hvaða gagnagrindur sem er úr **C++ standard template library** og gagnagrindur smíðaðar af ykkur sjálfum, samnemendum eða kennurum í fyrri verkefnum þessa námskeiðs.

Forritið skal hafa aðgang að textaskrá sem inniheldur "orðabanka" til að velja orð úr. Það þarf að vera mögulegt að bæta orðum í skrána og fjarlægja þau, þannig að forritið virki áfram. Mælt er með því að orðabankinn sé lesinn inn í gagnagrind í upphafi keyrslu þannig að bankinn sé aðgengilegur hratt og örugglega á meðan á keyrslu stendur. Í hvert sinn sem nýr leikur hefst er orð slembivalið úr orðabankanum. Síðan skrifar forritið út, á undan hverju giski, orðið þar sem hver stafur sem ekki hefur verið giskað á er bandstrik ('-') en hver stafur sem giskað hefur verið á er á réttum stað í orðinu. Í upphafi mun orðið því einfaldlega vera röð af bandstrikum. Forritið skrifar einnig út fjölda ágiskana sem leikmaður á eftir.

Þegar leikur hefur unnist eða tapast á forritið að láta leikmann vita og bjóða honum upp á að hætta eða spila annan leik.

Eftirfarandi forritshluta má forrita í hvaða röð sem er en mælt er mað því að klára hvern flokk áður en hafist er handa við þann næsta. Gefið er fyrir bæði **virkni** og **uppbyggingu forrits** í hverjum hluta.

1. GRUNNLEIKURINN - 50%

- Sýna orðið sem röð bandstrika, t.d. **structure** = --------- … - **10%**
- Notandi getur giskað á staf og er látinn vita hvort orðið inniheldur stafinn eða ekki - **10%**
- Sýna réttar ágiskanir, t.d. **structure** = **-tr--t-r-** eftir að hafa giskað á 'r' & 't' - **10%**

- Forritið sýnir fjölda ágiskana sem leikmaður á eftir - **5%**
- Forritið nemur þegar ágiskanir eru búnar og leikmaður hefur tapað leiknum - **5%**
- Forritið nemur sigur þegar orðið hefur fullklárast - **10%**

2. BETRA FLÆÐI Í STÖKUM LEIK - 30%

- Notandi getur slegið inn eða valid fjölda ágiskana áður en leikurinn hefst - **5%**
- Eftir að leik lýkur getur leikmaður valið um að hætta eða hefja leik að nýju - **5%**
- Forritið geymir orðabanka í gagnagrind - **10%**
- Forritið slembivelur orð úr orðabankanum í upphafi hvers leiks - **5%**
- Orðabankinn er geymdur í og lesinn úr skrá - **5%**

3. TENGDAR RUNUR AF LEIKJUM - 20%

- Haldið er utan um sigra og töp á meðan forritið keyrir (geymt í klösum og breytum) - **5%**
- Notanda er leyft að giska á heilt orð - **5%**
- Finnið leið til að reikna stig fyrir röð leikja og haldið utan um stigatöflu - **5%**
    - Stig eru geymd í skrá svo að taflan sé aðgengileg milli forritskeyrslna og eru mögulega undir áhrifum:
        - Fjölda rangra ágiskana á orð
        - Heildarfjölda ágiskana á orð (færri ef leyft að giska á heil orð)
        - Fjölda sigurleikja áður en leikur tapast (eða heildarfjölda stiga þeirra leikja), o.s.frv.
- Forritið leyfir notanda að bæta orðum í orðabankann (og skrána) gegnum forritið sjálft - **5%**