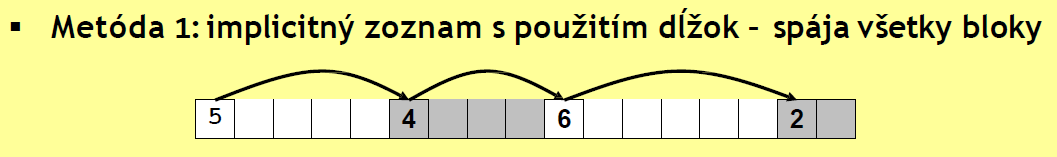
Zadanie 1 DSA dokumentácia



Na vypracovanie môjho projektu som použil metódu 1. Moja implementácia spočíva v tom že každý blok pamäte ma aj hlavičku aj pätičku. Na obrázku môžete vidieť že moja hlavička zaberá 4 bajty a rovnako aj moja pätička zaberá 4 bajty a je v nej napísaný rozmer pamäte ktorú si užívateľ vypýtal ak je číslo

v pätičke a hlavičke kladne znamená to že blok je voľný ak je záporne znamená to že blok už je alokovaný priestor medzi hlavičkou a pätičkou je mieste do ktorého si môže užívateľ zapísať čo chce.

názorná ukážka

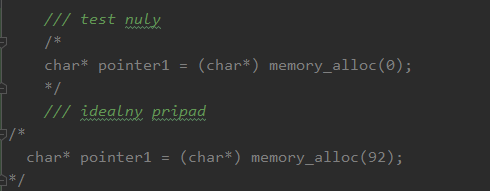
Pri veľkosti poľa char[100] by bolo ideálne riešenie že môže užívateľ využiť celých 100 bajtov ale kvôli mojej réžii u mňa môže použiť iba 92 bajtov takže v ideálnom prípade je moja pamäť 92% efektívna ako ozajstný malloc.

*Moja zložitosť je O(n) pretože v najhoršom prípade musím prejsť všetky bloky aby som mohol alokovať.*

Testovanie: Moje testovanie spočíva vo viacerých testoch všetky mam v Maine za komentovane

a pripravene na vyskúšanie. Môj prvý test kontroluje krajne prípady funkcie malloc to znamená že čo sa stane keď dám malloc(0) vráti sa mi pointer na Null pretože nemôžem alokovať 0 bajtov.

Ďalšia krajná situácia je že alokujem maximálnu veľkosť bloku to znamená v mojom prípade 100 charov ale reálne môžem využiť len 92 kvôli hlavičke a pätičke. Čiže čo sa stane keď dám malloc(92) využijem celu moju dostupnú pamäť a vráti sa mi smerník na miesto za hlavičkou.

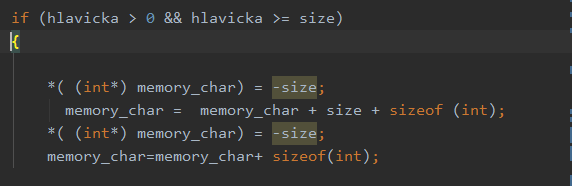


Ďalej testujem svoju funkciu free moje testovanie spočíva vo vyskúšaním krajných situácii to znamená mergovanie prvého bloku s druhým a následne aj posledného s predposledným. Ďalej spočíva vo vyskúšaním všetkých možných kombinácii aké môžu nastáť pri mergovani to znamená situácie typu voľný voľný voľný, plný voľný voľný, voľný voľný plný, iné situácie nastáť nemôžu s toho dôvodu že dostanem

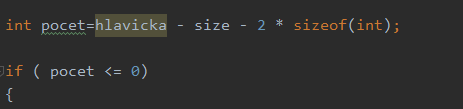
smerník na nejaký obsadený blok a zaujímajú ma už len blok pred nim a za nim. Pri týchto mojich testoch funkcie free samozrejme aj mením hodnoty mallocu a pre lepšiu viditeľnosť a lepšie orientovanie

v memory som použil funkciu memset a zapísal do pamäti rôzne čísla. Pre všetky tieto testy používam pamäť o veľkosti 100 charov čo je 100 bajtov a moja logika je že keď mi všetko funguje na takejto malej pamäti tak všetko bude fungovať aj v pamäti väčšej.

Teraz prejdem na vysvetlenie určitých časti môjho kódu:

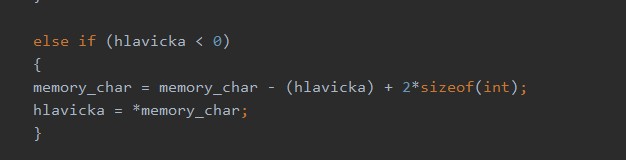


Hlavička sú prvé štyri bajty v ktorých mam zapísane číslo tato podmienka vlastne skontroluje ci je tento blok obsadený alebo nie je a zároveň skontroluje ci je tento blok dostatočne veľký pre pamäť ktorú požaduje užívateľ. Memory char už ako z názvu plynie je char a char ma 1 bajt int číslo potrebuje až 4 to znamená že ho musím pretypovať nato slúži ta zátvorka a int\* keď ho pretypujem na toto miesto následne vložím záporne číslo to mi indikuje že tento blok už je plný. Memory char bez pretypovanie znamená že sa v tom mojom bloku posúvam a posúvam sa po bajtoch keď natrafím na správne miesto to jest miesto kde chcem byt skončím posuv a na toto miesto následne zapíšem pätičku.

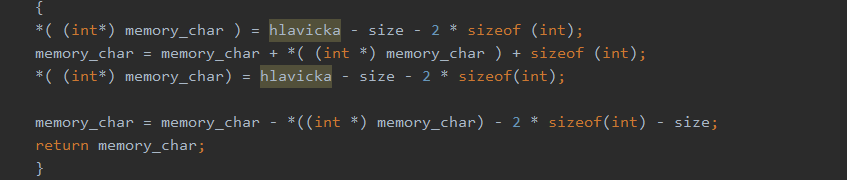


Toto mi slúži na zistenie ci po splitnuti bloku bude nasledujúci blok použiteľný to znamená

že bude mat viac ako 8 bajtov pretože na zápis 1 bajtu užívateľom potrebujem až 9 bajtov 1 bajt plus 4 bajty hlavička plus 4 bajty pätička. Ak by blok splitnuti blok menší ako 9 bajtov tak ho spojím s mojim terajším blokom pretože by som pre tento splitnuti blok nemal nijaké využitie.

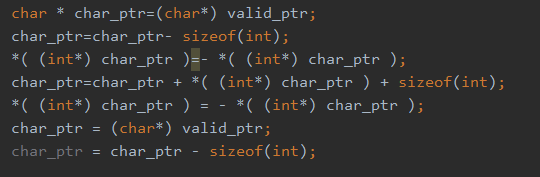


Ak je blok zabratý to znamená hlavička je mínusová posuniem sa na ďalší blok a znova opakujem situáciu zisťujem ci je záporný ak áno znova sa posuniem ak nie ostávam.



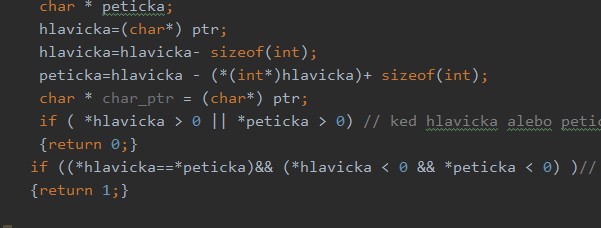
Toto tu slúži na zapísanie veľkosti voľného bloku plus na vrátenie smerníka na prvé miesto za hlavičkou.

Funkcia memory free je veľmi dobre okomentovaná a premenne sú pomenovane intuitívne takže sa v tom da veľmi dobre vyznať preto priložím iba jeden screen kódu



Do funkcie memory free prichádza vlastne smerník na prvý bajt po hlavičke to znamená že ja sa najprv posuniem s mojim char\_ptr na miesto hlavičky a následne tuto hlavičku zmením na kladnú následne sa znova posuniem a zmením pätičku na kladnú to znamená že som tento blok uvoľnil následne si uložím aktuálny predošlý a nasledujúci char ptr na požadovane miesto a zisťujem ci bloky treba mergovat alebo nie.

Funkcia memory check



V tejto mojej funkcii vždy dostanem platný smerník na blok pamäte následne si nastavím premenne hlavička a pätička na hlavičku a pätičku a zisťujem ak je aspoň jedna premenná kladná znamená to že daný blok pamäte nie je správny vrátim nulu ak obidve sú záporne a pätička sa rovná hlavičke tak vrátim jedna pretože daný blok je správny a je pridelený.

Funkcia memory init v tejto funkcii nerobím nič iné iba si pripravím svoje pole to znamená moje pole

o veľkosti 100 rozdelím na 4 bajty na začiatok hlavička 4 bajty nakoniec pätička a 92 blokov ktoré môže užívateľ používať a ktoré používam ja vo svojich testoch.