Sintaksa

U nastavku slijede pravila sintakse u programiranju te imenovanju datoteka. Ona služe da bi naš izvorni kod bio univerzalan za sve članove. Nazivi su na engleskom te nose puna imena osim u slučajevima prikazanima u sljedečoj listi.

* + **Prefiksi**
    - m\_ varijable kao članovi (membere) klase
    - g\_ globalne varijable
    - s\_ statično alocirane varijable (static)
    - V virtualne funkcije
    - C klase
    - I klase sučelja (interfaces)
    - b(ool),d(ouble),f(loat),i(nt),p(ointer)...
      * radi lakšeg prepoznavanja ukoliko ima više varijabli istih imena različitog tipa
      * nije obavezno, preporučljivo u pseudokodu
* **Funkcije, klase i objekti klasa**
  + Prefiks (ukoliko postoji) pa veliko prvo slovo u svakoj riječi bez razmaka i povlaka (underscore)
  + Primjeri
    - MoveObject
    - LoadStatsFromServer
    - VCreateHandle
    - IAudioBuffer
    - CActor \*Actor = GCC\_NEW CActor;
* **Varijable**
  + Prefiks (ukoliko postoji) pa malo početno slovo prve riječi te veliko prvo slovo u svakoj riječi. Kod objekta klase treba se pridržavati pravila za funkcije
  + Primjeri
    - currentObject
    - i
    - m\_lastIndexID
    - g\_time
    - gs\_Connection
    - m\_bGender
    - const float fPi = 3.1418f;
    - const double dPi = 3.1418;
* **Strukture**
  + Povlaka (underscore), malopočetno slovo prve riječi te veliko slovo u svakoj riječi, bez razmaka i povlaka
  + Primjeri
    - \_object
    - \_actorValues
    - \_actor
* **Predkompajlerske definicije / Makroi** (#define...)
  + Sva velika slova, riječi odvojene povlakama (underscore)
  + Primjeri
    - MAX\_OBJECTS
    - PLAYER\_LIMIT
* **Datoteke i mape**
  + Po volji, za datoteke klasa korisiti format za imenovanje klasa
  + Primjeri
    - vertex\_formats.cpp
    - Krovna\_ploca.em
    - CActor.cpp CActor.h
* **Komentari**
  + Raspište se o danoj funkciji, ako treba i u više redova te urediti da bude što jasnije
* **Zagrade, uvlačenja i razmaci**
* **Funkcije**: Nakon imenovanja funkcije i njenih atributa u novom redu otvoriti vitičastu zagradu, te u novom redu sa jednom tabulaturom (default 4 razmaka, korisiti TAB, ne SPACE!) započeti pisati kod funkcije. Nakon svih funkcija u novom redu na početku zatvoriti vitičastu zagradu (poravanje s otvorenom)
  + Primjer:

bool Function()

{

*// Something here*

*return false; // no error*

}

* **Petlje**: Petlje pišemo prema predlošcima, FOR i WHILE kao funkcije, a DO WHILE započinjemo kao funkciju, te na kraju petlje i nakon vitičaste zagrade u ISTOM redu napisati while(uvjet).
  + // FOR

for(int objectid = 0; objectid < MAX\_OBJECTS; objectid++)

{

*// do something*

}

// DO WHILE

do

{

if(stop)

break;

} while(true);

// WHILE

while(!stop)

{

// do something

}

* **Logičke** **i računarne operacije**: Atributi (varijable i konstante) dodiruju zagrade, dok su operatori (kao što su +,-,\*,/,!=,<,>,<=,>=, &&,|| itd. ) odvojeni od samih atributa izuzev operatora negacije, modularnog dijeljenja te inkrementa ili dekrementa itd. (vidi u primjeru) . Operatore Ukoliko izvršavamo samo jednu naredbu nakon ifa pišemo je bez zagrada u novom redu te s jednim tabulatorom dalje od IF-a. Ukoliko naredbi ima više, koristimo format za funkciju.

if(a == b || (a != b && b < c) || c > a)

a = b++ - (c + a – b%2);

else

{

a--;  
 b = a%c;

c++;

}