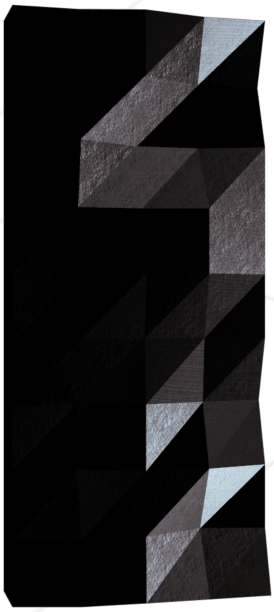

Coding Standard - Academy+Plus

Versiunea 2.0.0

Staff Academy+Plus contact@academyplus.ro

Sumar: Acest document descrie standardul de scriere a codului in limbajului C, in vigoare la Academy+Plus. Un standard de programare defineste un set de norme ce reglementeaza scrierea unui cod. Acest standard e obligatoriu de respectat atata vreme cat scrieti cod in limbajul C la Academy+Plus.

Cuprins

Capitolul I

Prefata

Acest document descrie standardul de scriere a codului in limbajului C, in vigoare la Academy+Plus. Un standard de programare defineste un set de norme ce reglementeaza scrierea unui cod. Acest standard e obligatoriu de respectat atata vreme cat scrieti cod in limbajul C la Academy+Plus.

I.1 De ce necesara impunerea unui standard?

Un standard are doua obiective principale: 1-Uniformizarea codului scris de voi astfel ca toata lumea sa-l poata citi cu usurinta, studenti si profesori. 2-Scrierea codului trebuie sa fie simpla si clara.

I.2 Standardul in munca voastra

Toate fisierele sursa C trebuie sa respecte standardul de la Academy+Plus. Standardul va fi verificat de catre cei care va corecteaza lucrarile si pentru orice nerespectare a standardului veti primi nota 0, la orice proiect sau exercitiu. Apoi, prin intermediul conexiunii peer, cel care va corecteaza lucrarile va lansa "Norminette"prezenta in directorul vostru de lucru. Doar lucrarile validate de "Norminette" vor fi luate in considerare. Vor fi verificate de "Norminette" doar partile obligatorii ale Standardului.

I.3 Sfaturi

Dupa cum poate v-ati dat seama Coding Standards nu reprezinta o constrangere. Dimpotriva, el reprezinta o masura de protectie pentru a va ghida in scrierea simpla a codului C. De aceea e absolut vital sa scrieti codul conform acestui Standartd chiar daca asta duce la o scriere mai lenta in primele ore. Un fisier sursa ce e scris neconform cu Coding Standards e la fel de gresit ca unul scris gresit. Fiti rigurosi si aplicati Standardul pentru ca in scurt timp acesta sa devina un automatism.

I.4 Termeni si conditii

Eventualele bug-uri existente in "Norminette" va rugam sa ni le raportati pe sectiunea de forum a intranetului (in romana ori engleza). "Norminette" va fi actualizata, pe baza

rapoartelor trimise de voi.

Capitolul II

Norme

II.1 Conventii de nume

Parte obligatorie

- Numele unei structuri trebuie sa inceapa cu `s_`.
- Un nume de typedef trebuie sa inceapa cu `t_`.
- Un nume de union trebuie sa inceapa cu `u_`.
- Un nume de enum trebuie sa inceapa cu `e_`.
- Un nume de variabila globala trebuie sa incepa cu `g_`.
- Numele de variabile si de functii trebuie sa aiba in compunere exclusiv minuscule, cifre si caracterul '_' (Unix Case).
- Numele fisierelor si directoarelor trebuie sa aiba in compunere exclusiv minuscule, cifre si caracterul '_' (Unix Case).
- Fisierele trebuie sa fie compilabile.
- Caracterele ce nu fac parte din tabela ascii standard nu sunt permise.

Recomandari

- Obiectele (variabile, functii, macro-uri, tipuri, fisiere sau directoare) trebuie sa aiba numele cele mai explicite sau mnemonice. Doar 'compteurs' pot fi numiti dupa preferinta.
- Abrevierile sunt tolerate doar in masura in care ele permit reducerea semnificativa a dimensiunii unui nume fara a pierde sensul. Numele compuse din mai multe cuvinte vor fi separate de '_'.
- Toti identificatorii (functii, macro-uri, tipuri, variabile, etc) trebuie sa fie in engleza.

- Utilizarea tuturor variabilelor globale trebuie justificata.

II.2 Formatare

Parte obligatorie

- Toate fisiere voastre trebuie sa inceapa cu header-ul standard al ACADEMIEI+PLUS pe prima linie. Acest header este disponibil in editoarele `emacs` si `vim` puse la dispozitie.

- Va trebui sa va indentati codul cu un tabulator de 4 spatii. Atentie! Acesta nu e echivalent cu 4 spatii.

- Fiecare functie va trebui sa aiba maxim 25 linii fara a lua in considerare acoladele blocului functiei.

- Nicio linie nu poate avea mai mult de 80 coloane, cu comentariile incluse. Atentie: un tabulator nu este considerat ca fiind o coloana dar este luat in calcul cu cele n spatii pe cate le reprezinta.

- Se va scrie o singura instructiune pe linie.

- O linie goala nu trebuie sa contina spatii ori tabulatoare.

- O linie nu trebuie sa fie terminata niciodata cu spatiu ori tabulatoare.

- Cand intalniti o acolada deschisa ori inchisa, sau un sfarsit al unei structuri de control, trebuie sa introduceti un retur de linie.

- Fiecare virgula ori punct si virgula trebuie urmate de un spatiu, daca nu urmeaza sfarsitul randului.

- Fiecare operator (binar ori ternar) precum si operanzii corespunzatori trebuie sa fie separati de un spatiu si numai unul.

- Fiecare cuvant cheie al limbajului C trebuie sa fie urmat de un spatiu, in afara de cuvintele cheie de tip (ca int, char, float, etc.) precum si sizeof.

- Fiecare declaratie de variabila trebuie sa fie indentata pe aceeasi coloana.

- Asteriscul pointerilor trebuie sa fie lipit de numele variabilelor.

- Se va face o singura declarare de variabila pe linie.

- Se poate face o singura declarare si initializare pe linie, cu exceptia variabilelor globale si a celor statice.

- Declararile trebuie sa fie puse in fata functiei si trebuie separate de implementarea acesteia cu o linie goala.
- Nu se admite nicio linie goala pe parcursul declararilor si a implementarii.
- Asignarea (atribuirea) multipla este interzisa.
- Puteti trece la linia urmatoare in cadrul unei instructiuni (adica o instructiune scrisa pe mai multe linii), dar trebuie sa introduceti indentari pentru paranteze si operatori de atribure. Operatorii trebuie sa fie plasati la inceputul liniei.

II.3 Parametri functiei

Parte obligatorie

- O functie poate primi cel mult 4 parametri.
- O functie fara niciun argument va primi in mod explicit argumentul void in prototip.

II.4 Functii

Parte obligatorie

- Parametri prototipurilor functiilor trebuie sa fie numiti.
- Fiecare definitie a unei functii trebuie sa fie separata de o linie goala de urmatoarea.

Recomandari

- Identificatorul unei functii trebuie sa fie aceleasi cu numele fisierul care o contine. Aceasta regula se aplica si fisierelor header C.

II.5 Typedef, struct, enum si union

Parte obligatorie

- Trebuie sa utilizati tabulatorul daca declarati o structura, un enum ori union.
- La declararea unei variabile de tip struct, enum sau union nu puneti decat un spatiu dupa tip.
- Trebuie sa utilizati un tabulator intre cei doi parametri ai unui typedef.
- Cand declarati o structura, union sau enum cu typedef, toate regulile se aplica si trebuie sa aliniati numele typedef cu numele struct, union sau enum.
- Nu puteti declara o structura intr-un fisier .c.

II.6 Header-e

Parte obligatorie

- In fisierele header sunt admise doar includerile de header-e (de sistem sau nu), declaratiile, define-urile, prototipurile si macro-urile.
- Toate include-urile .h trebuie facute la inceputul fisierelor (.c sau .h).
- Header-ele vor fi protejate contra dublei includeri. Daca fisierul se numeste `ft_foo.h`, macro-ul corespunzator va fi `FT_FOO_H`.
- Prototipul functiilor trebuie sa se gaseasca exclusiv in fisiere .h.
- Includerea unui fisier header (.h) ce nu e folosit e interzisa.

Recomandari

- Toate includerile fisierelor header trebuie sa fie justificate fie intr-un fisier .c fie intr-unul .h.

II.7 Macro-uri si Pre-procesare

Parte obligatorie

- Define-urile ce definesc codul sunt interzise.
- Macro-urile multilimba sunt interzise.
- Doar numele macro-urilor sunt scrise cu majuscule.
- Trebuie identificate caracterele ce urmeaza unor #if , #ifdef sau #ifndef

II.8 Lucruri interzise!

Parte obligatorie

- Nu aveti dreptul sa utilizati urmatoarele:
  - for
  - do...while
  - switch
  - case
  - goto
- Operatorul conditional imbricat '?'

II.9 Comentarii

Parte obligatorie

- Comentariile pot fi incluse in toate fisierele sursa.
- Nu trebuie sa existe comentarii in corpul functiilor.
- Comentariile incep si se termina cu o linie goala. Toate liniile intermediare vor fi aliniate si vor incepe cu '**'.
- Nu se fac comentarii cu //.

Recomandari

- Comentariile adaugate trebuie sa fie utile si scrise in engleza.
- Comentariile nu pot justifica scrierea unui cod ilizibil: alegerea numelor variabilelor intr-un mod neintuitiv, prescurtari neclare etc.

II.10 Fisiere

Parte obligatorie

- Nu puteti include un fisier .c.
- Nu sunt permise mai mult de 5 definitii de functii intr-un fisier .c.

II.11 Makefile

Recomandari

- Regulile $(NAME), clean, fclean, re si all sunt obligatorii.
- Proiectul e considerat nefunctional daca makefile re-linkeaza.
- In cazul unui proiect cu mai multe fisiere binare, pe langa regulile precedente, trebuie sa aveti si regula all compiland ambele binare si o regula specifica de compilare pentru fiecare fisier.
- In cazul unui proiect care apeleaza o biblioteca de functii (de exemplu `libft`), fisierul makefile trebuie sa compileze automat aceasta biblioteca.
- Utilizarea 'wildcard'-urilor (*.c de exemplu) e interzisa.