Programozás I.

Programozási hibák megkeresése és kijavítása

Szintaktikai hibák

Mi a szintaktikai hiba?

- A hibák egyszerűbb típusa
- Olyan hiba, ami a kódot értelmezhetetlenné teszi a fordító számára
- Ilyen hiba észlelése esetén a fordító hibaüzenetet ír ki
- Az üzenet alapján a legtöbb esetben egyértelműen megtalálható a hiba

Hogy néz ki egy hiba?

- A hibaüzenet alakja a következő:
- hiba.c:21:10: error: itt van a hiba leírása
  - Szintaktikai hiba, melyet a fordító a hiba.c fájl 21. sorának 10. karakterénél észlelt.
  - Látható, hogy a fordító megadja azt a helyet, ahol a hibát találta.
  - A leírásban azt is megadja, miért nem tudta értelmezni az adott sort.
  - Az “error” helyett “warning” is állhat, amire szintén érdemes figyelni, szemantikai hibára utalhat.

Hiba megkeresése

- A hiba megkeresésének legfontosabb szabálya, hogy mindig a legelső hibával kezdjük.
- A fordítás menete miatt bizonyos hibák a kód későbbi, amúgy jó részien is hibaüzenethez vezetnek.
- Elképzelhető, hogy a legelső hibát kijavítva az összes többi is eltűnik.

Gyakori hibák

- Üzenet: “hiba.c:1:21: error: studio.hu: No such file or directory”
  - Nem találja a studio.hu fájlt.
  - Elírtuk az “stdio.h”-t, javítsuk.

Gyakori hibák

- Üzenet: “hiba.c:6:17: error: ‘aple’ undeclared”
  - Nincs olyan változó, hogy **aple**.
  - Az eredeti változó neve **apple**, csak elgépeltük, javítsuk.
  - Ha mégsem elgépelés, akkor elfelejtettük létrehozni, ekkor hozzuk létre.
  - Elképzelhető még, hogy létrehoztuk, de a kódban még a létrehozás előtt használjuk, ekkor hozzuk előrébb a deklarációt.

Gyakori hibák

- Üzenet: “hiba.c:7:2: error: expected ‘;’ before ‘return’”
  - Nincs ‘;’ a **return** előtt.
  - A **return**-nel kezdődő sort megelőző sor végéről lehagytuk a ‘;’-t, megoldás: pótoljuk.

Gyakori hibák

- Üzenet: “hiba.c:9:1: error: expected ‘;’, indentifier or ‘<‘ before ‘int’”
  - Valószínű hiba: a struktúra-definíció végéről lemaradt a ‘;’. Ellenőrizzük, és ha kell javítsuk.

Gyakori hibák

- Üzenet: “hiba.c:14:2: error: unknown type name ‘computer’”
  - A **computer**-t típusként értelmezné, de nincs ilyen típus.
  - A típust **struct computer**-nek hívják, így is kell használni, vagy a **typedef**-fel átnevezni.

Gyakori hibák

- Üzenet: “hiba.c:14:18: error: ‘struct computer’ has no member named ‘fl’”
  - A **struct computer** struktúrának nincs **fl** nevű tagja.
  - Olyan, mint a nem létező változó hibája.
  - Vagy elgépelés, vagy nem hoztuk létre.

Gyakori hibák

- Üzenet: “hiba.c:16:6: error: request for member ‘fl’ in something not a structure or union”
  - Egy változónak szeretnénk elérni az **fl** adattagját, de az a változó nem struktúra.
  - Az a változó nem struktúra, valószínűleg figyelmetlenség történt, pl.: rossz változó nevét írtuk oda.

Gyakori hibák

- Üzenet: “hiba.c:15:5: error: incompatible types when assigning to type ‘int[3]’ from type ‘int’”
  - Az értékatás két oldalán különböző típus áll.
  - Bármikor előfordulhat, ha a két változó típusa nem egyezik meg, és a fordító nem tudja átalakítani.
  - Jelen esetben a tömb használatakor hagytuk le az indexelést.

Gyakori hibák

- Üzenet: “hiba.c:14:6: error: subsripted value in neither array nor pointer nor vector”
  - Indexelünk valamit, amit nem lehet, ismét figyelmetlenség, pl.: rossz változót használtunk, vagy nem tömbként hoztuk létre.

Gyakori hibák

- Üzenet: “hiba.c:5:2: error: parameter ‘apple’ is initialized”
- Vagy: “hiba.c:5:2: error: expected declaration specifiers before ‘printf’”
  - Valószínűleg az “int main()” sor után lemaradt a ‘{‘. Írjuk oda, és a hiba eltűnik.

Gyakori hibák

- Üzenet: “hiba.c:8:2: error: expected declaration or statement at end of input”
  - A bemenet (vagyis a forrásfájl) végéről valami hiányzik.
  - Ez a valami jellemzően a ‘}’, amit oda kell írni a végére.

Szemantikai hibák

Mi a szemantikai hiba?

- A szemantikai hiba olyan hiba a kódban, amely nem teszi azt értelmezhetetlenné, vagyis a fordító el tudja készíteni a futtatható állományt.
- Azonban, a program működése nem lesz megfelelő az ilyen hibák hatására.
- Mivel szintaktikailag helyes a kód, ezért nehezebb az ilyen hibákat javítani.

Hibaüzenetek

- A fordító írhat ki olyan hibaüzenetet, amelyben “error” helyett “warning” szerepel.
- Ezen üzenetek nem okoznak fordítási hibát.
- Ennek ellenére érdemes ezeket is megvizsgálni, mert szemantikai hibára utalhatnak.

Gyakori hibák

- Üzenet: “hiba.c:14:2: warning: suggest parentheses around assignment used as truth value”
  - Akkor jellemző, ha a feltétel megadásakor == helyett csak =-t írunk.

Gyakori hibák

- Üzenet: “hiba.c:18:6: warning: assignment makes integer from pointer without cast”
  - Egész értékű változónak konverzió nélkül mást adtunk meg.
  - Valószínűleg lemaradt a tömb használatából az indexelés.

Gyakori hibák

- Üzenet: “hiba.c:13:2: warning: too many arguments for format”
  - A printf-be több változót írtunk, mint amennyit a formázásban megadtunk.

Gyakori hibák

- Üzenet: “hiba.c:13:2: warning: format ‘%d’ expects a matching argument ‘int’”
  - A printf-be kevesebb változót írtunk, mint amennyit a formázásban megadtunk.

Gyakori hibák

- Üzenet: “hiba.c:13:2: warning: format ‘%f’ expects argument of type ‘double’, but argument 2 has type ‘int’”
  - A printf-ben nem egyezik meg a változó típusa a formázásban megadott, várt típussal.

Valgrind használata

- A program működését mindig célszerű a valgrind segítségével ellenőrizni, mert képes a memória helytelen használatát jelezni.
  - valgrind ./program

Valgrind használata

- Alap esetben a valgrind csak a hiba megnevezését írja ki, magát a hibát nem.
- Viszont, ha a programot debug módban fordítjuk, akkor a futtatható állomány több információt tartalmaz, így a valgrind kiírja a hiba helyét is.
- Debug fordítás: -g kapcsoló
  - gcc –Wall –g –o prg program.c

Gyakori hibák

- Valgrind: “Conditional jump or move depends on uninitialised value(s)”
  - Nem inicializált érték használata feltételhez.
- Debug fordítás esetén kiírja a használat helyét is:
  - “at 0x400555: main (hibas.c:9)”
  - Vagyis a hibas.c fájl 9. sorában

Gyakori hibák

- Néha azt is jelzi, ha inicializálatlan változót használunk printf-ben, ekkor azonban több helyre is hibát ír.

  “at ….: vfprintf (vfprintf.c: 1660)”

  “at ….: printf (printf.c: 33)”

  “at ….: main (hibas.c: 9)”
- Minket jelenleg csak az utolsó sor érdekel, a több sor jelentése majd a későbbiek folyamán válik érthetővé.