
Jporta használata feladatbeadás

Szeberényi Imre

BME IIT

<balage@iit.bme.hu>



MŰEGYETEM 1782

Jporta

- A rendszer nem fejlesztő rendszer.
- Alapvetően ellenőrzésre való.
- Igyekszünk minden elkövethető hibát kiszűrni, és támogató megjegyzésekkel ellátni.
- Fontos, hogy feltöltés előtt lokális környezetben le legyen tesztelve a beadandó feladat.
- Ezért minden összetettebb feladathoz az GIT-ről letölthető példákat, tesztprogramokat kínálunk.
- Ezt érdemes megérteni, mert **hasonlóan**, vagy sokszor **pontosan ugyanúgy** tesztel a Jporta is.
- Ha lefutatta a lokális teszteket, akkor fordítási hiba a Jportán csak úgy fordulhat elő, hogy nem pontosan ugyanazokkal a fordítási opciókkal, vagy verziókkal fordított lokálisan.

Compiler hibát jelez

Fontos elolvasni, megérteni, értelmezni a hibajelzést.

Néhány példa:

```
nem_oo.cpp: In function 'char* saját::unique(char*, char*)':  
nem_oo.cpp:76:30: error: comparison between signed and unsigned integer  
expressions  
    for(int i = 0; i < eddigiek.size(); i++)  
                        ^
```

cc1 plus: all warnings being treated as errors

Nem fogadja el, ha van figyelmeztető üzenet. Itt előjelest hasonlít a kód előjeletlennel.

Compiler hibát jelez/2

```
nem_oo_teszt.cpp:47:1: error: expected unqualified-id before using
using std::cout;
^
nem_oo_teszt.cpp: In function int main():
```

A példában nem a hallgató által feltöltött fájlban jelez hibát. Ez minden bizonnyal a feltöltött header (.h, .hpp) fájlban elkövetett hiba következménye. Jelen esetben valószínű pontosvessző hiányzik a header utolsó deklarációjából.
(A preprocessor szövegesen összefűzi a két fájlt.)

Compiler hibát jelez/3

```
nem_oo.cpp: In function 'char* saját::unique(char*, char*)':  
nem_oo.cpp:67:18: error: ISO C++ forbids variable length array 'uj' [-Werror=vla]  
    char uj[meret];  
           ^  
  
cc1 plus: all warnings being treated as errors
```

Nem fogadjuk el, ha van figyelmeztető üzenet. Ai ISO C++ szabvány szerint változó méretű tömb nem használható. (A példában nincs is értelme használni.)

Compiler hibát jelez/4

```
nem_oo_teszt.cpp: In function int main()
nem_oo_teszt.cpp:87:17: error: max is not a member of saját
    cout << saját::max(i1, i2) << endl;
                   ^
```

```
nem_oo_teszt.cpp:87:17: note: suggested alternative:
In file included from /usr/include/c++/5/bits/char_traits.h:39:0,
from /usr/include/c++/5/ios:40,
```

.....

Sok további hasonló hibaüzenet

Valószínű nem a 'sajat' névtérben valósította meg a max sablont!

Linker hibát jelez

```
swap_test.o: In function `main':  
swap_test.cpp:(.text+0x1a2): undefined reference to `swap(char&, char&'  
swap_test.cpp:(.text+0x468): undefined reference to `swap(double&, double&'  
swap_test.cpp:(.text+0x602): undefined reference to `swap(char const* &, char  
const* &)'  
collect2: error: ld returned 1 exit status
```

Nem valósította meg az elvárt függvényeket/objektumokat, vagy nem az összes elvárt függvényt valósította meg!

Linker hibát jelez/2

```
swap_test.o: In function `main':  
swap_test.cpp:(.text+0x7): multiple definition of `main'  
swap.o:swap.cpp:(.text+0x0): first defined here  
collect2: error: ld returned 1 exit status
```

Main függvényt is feltöltötte, de azt a tesztprogram már tartalmazza.

Futási hibák

- A futatott program megállási státusza akkor jó ha az 0.
Jellemző hibakódok:
 - 11 → durva memória kezelési hiba
Jellemzően elszabadult (érvénytelen, pl. NULL) pointer használata, esetleg alul- vagy túlindexelés okozza.
 - 9 → a keretrendszer kilőtte a programot, mert sokáig futott
Jellemzően végtelen ciklus okozza.
 - 120 → memtrace hibát jelzett
- Futáskor a program nem ír ki semmit, vagy csak nagyon keveset a standard outputra.
 - Valószínű végtelen ciklusba kerül, vagy elszáll hibás művelet (pl. memória kezelési hiba, nullával való osztás, stb.) miatt.

<https://infocpp.iit.bme.hu/jporta/NHF>

NHF beadás

- Feltöltendő állományok:
 - **Ne használjon ékezetes** karaktereket az állományok nevében!
 - Forrásfájlokat tartalmazó ZIP fájlban **ne legyen alkatalógus** (fájlok az archívum gyökérében legyenek).
 - Adatfájlokat tartalmazó zip fájl (opcionális). Tartalma abba a katalógusba másolódik, ahol a program fut.
 - Bemeneti adatfájl, melynek tartalmát futtatáskor a program szabványos bemenetére továbbítjuk.
 - Dokumentáció feltöltése a feladat kiértékeléséhez nem szükséges, csak **a végleges beadásnál elvárt**, akkor viszont hiánya a megoldás elutasítását vonja maga után.

Standard input

The screenshot displays a C++ IDE interface with several panels. On the left, a 'Futtatás' (Execution) panel shows a list of input values from 1 to 10. Below it, an 'Ellenőrzések' (Checks) panel indicates a successful memory leak test. In the center, a 'Felhasználói fájlok' (User files) panel lists 'main.zip' (319,0B) and '1.txt' (21,0B). On the right, a 'Részletek' (Details) panel shows the source code for 'main.cpp', which includes headers for iostream and gtest_lite.h, and contains a main function with a while loop and a return statement.

Futtatás
Visszatérési érték: 0
Szabványos kimenet

```
inp: 1  
inp: 2  
inp: 3  
inp: 4  
inp: 5  
inp: 7  
inp: 9  
inp: 10
```

Ellenőrzések

✓ 1. ellenőrzés

Sikeres memóriaszivárgás teszt!

Felhasználói fájlok

név: main.zip
méret: 319,0B
letöltés

név: 1.txt
méret: 21,0B
megtekintés letöltés

Overall code coverage: 100,0 %

Részletek

main.cpp

```
1. #include <iostream>  
2. #include "gtest_lite.h"  
3.  
4. int main() {  
5.     TEST(osszead, három) {  
6.         EXPECT_EQ(3, 2 + 1);  
7.     } END  
8.     int i;  
9.     while (std::cin >> i)  
10.         std::cout << "inp: " << i << std::endl;  
11.     return 0;  
12. }  
13.
```

Jporta futtató környezete

- Linux (nincs widows.h)
- A ZIP-ben feltöltött fájlok **változatlanul** kerülnek bemásolásra. Ezek tartalmazhatnak bináris adatot is, ami architektúra függő lehet!
- Unix/Linux alatt nincs külön szövegfájl és bináris fájl.
 - DOS-ból Windows-ból hozott szövegfájl sorainak végén `\r\n` karaktersorozat van, amit `get()`, `ignore()` 2 db karakternek kezel!

Bináris vs Text

- A szövegfájlokban a sorok végét a különböző operációs rendszerek különböző módon jelölik:
 - `\n` - UNIX/Linux, Mac OS
 - `\r\n` - MS DOS/Windows, VMS,
 - `\n\r` - régi Mac OS
- A bajok ebből származnak, de persze a különböző kódkészletek (ASCII, UTF-8, ISO...) is okoznak gondot.

Bináris vs Text /2

- A C nyelv a UNIX-szal együtt fejlődött ki, a C programok sokezelési logikája erre épül. Így a nyelvben csak a `\n`-t szokás használni.
- Ehhez az `\r\n` ill. `\r` sorvéget tároló rendszerek úgy alkalmazkodtak, hogy bevezették a **bináris/text** módot.
- Text módban megnyitott fájlból olvasáskor a ténylegesen tárolt sorvégből (pl `\r\n`) **egyetlen** `\n` karaktert gyártanak. Íráskor pedig a `\n`-ből a ténylegesen tárolt sorvéget (pl. `\r\n` sorozatot).

Bináris vs Text /3

- Mivel a UNIX/Linux rendszerekben nem kell a program logikája és a tényleges tárolt sorvégek között konvertálni, így ezekben a rendszerekben a bináris és text mód között **nincs működésbeli** különbség.
- A rendszerben alapvetően nem keletkeznek, de **kívülről érkezhetnek** más formátumú fájlok.
- Ezek kezelése:
 - Fájlkonverzió (dos2unix, unix2dos, tr, ...)
 - Program szintű kezelés.

C++ szintű kezelés

- Általában a white-space karaktereket eldobjuk gyakran még a sorstruktúra sem érdekes.
- Alapesetben a skipws be van kapcsolva, ami a white-space karakterek eldobását jelenti. Ezt az extractor egy mezei karakter beolvasásánál is figyelembe veszi. pl:
 - `char c; std::cin >> c; // c-be csak nem white-space kar kerül.`
 - `std::cin >> noskipws >> c; // bármi`
 - `std::cin.get()` // független a skipws-től, de Windows/UNIX függő
 - `std::getline()` // független a skipws-től, de Windows/UNIX függő
- Ha mégis fontos a sorstruktúra ismerete, akkor a `getline()` többnyire minden további nélkül alkalmazható. A sor végén pedig **OS-től függően** lesz egy `\r`.

Hordozható getline

```
#include <iostream>
#include <string>

namespace cp { //cross-platform getline
std::istream& getline(std::istream& is,
                    std::string& str) {
    std::getline(is, str);
    if (str.size() && *str.rbegin() == '\r')
        str.erase(str.size()-1);
    return is;
}
```

Használat:

```
cp::getline(.....)
```

https://git.ik.bme.hu/Prog2/ell_feladat/NHF → `kesz/cross_paltform.h`

memtrace működése

- makrókkal lecseréli a **new** és **delete** kulcsszavakat saját függvényre, ami ellenőriz.
- C++11-ben a delete tagfüggvény törlésre is használható az osztály deklarációjában. A makró ezt is lecseréli, amiből baj van.
- Megoldás: Ezeket az header fájlokat a memtrace.h előtt kell include-olni.
 - „Kézzel”
 - Automatikusan

memtrace include

A memtrace.h azokat a standard fejléc állományokat, amelyek használják a *delete* kulcsszót automatikusan include-olja.

A fejlesztő környezetek azonban változnak. Így a memtrace.h-t folyamatosan frissíteni kell. A **legutolsó memtrace változat** a https://git.ik.bme.hu/Prog2/ell_feladat/Test projektben érhető el