Programozás
(GKxB_INTM021)

Dr. Hatwágner F. Miklós

Széchenyi István Egyetem, Győr

2018. augusztus 16.

Parancssori paraméterek

Parancssorban történő programindításkor a program nevét követően szóközökkel elválasztva *paraméterek* (command line arguments) adhatók meg, melyekkel a program működése befolyásolható.

Könyvtár tartalmának listázása – tömör formátum

```
wajzy@wajzy-notebook:~/Dokumentumok/gknb_intm021/Előadások/ea13$ ls
beolvas1.cpp  ea13.log  ea13.toc    masol2         param2      param4.cpp
beolvas2      ea13.nav  guns.txt    masol2.cpp     param2.cpp
beolvas2.cpp  ea13.out  kiir1       masolat.dat    param2.o
```

Könyvtár tartalmának listázása – hosszú (long) formátum

```
wajzy@wajzy-notebook:~/Dokumentumok/gknb_intm021/Előadások/ea13$ ls -l
összesen 4851004
-rw-rw-r-- 1 wajzy wajzy      393 máj    5 12:13 beolvas1.cpp
-rwxrwxr-x 1 wajzy wajzy    14613 máj    5 12:05 beolvas2
-rw-rw-r-- 1 wajzy wajzy      401 máj    5 12:13 beolvas2.cpp
```

Parancssori paraméterek

Feladat: készítsünk programot, ami kilistázza saját paramétereit!

Kimenet1

```
wajzy@wajzy-notebook:~/Dokumentumok/gknb_intm021/Előadások/ea13$
./param1 egy ketto harom "szokozt tartalmaz"
Az elinditott program neve: ./param1
1. parameter: egy
2. parameter: ketto
3. parameter: harom
4. parameter: szokozt tartalmaz
```

Kimenet2

```
wajzy@wajzy-notebook:~/Dokumentumok/gknb_intm021/Előadások/ea13$
./param1
Az elinditott program neve: ./param1
Nem adtak meg parametereket.
```

Parancssori paraméterek

A `main` függvény paraméterei

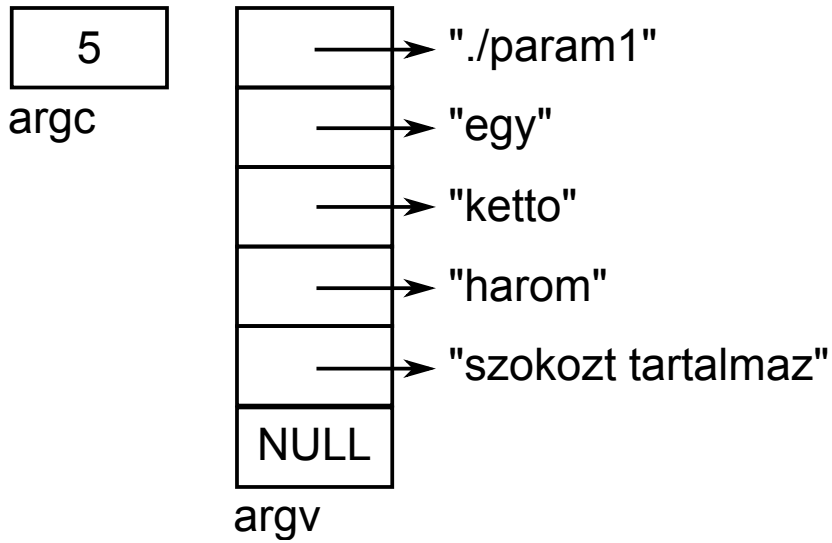
```
int main(int argc, char* argv[]) { /* ... */ }
int main(int argc, char** argv) { /* ... */ }
```

argc

A paraméterek száma (argument count), beleszámolva az elindított programot is

argv

Paraméterek mutatótömbje (argument vector)



Parancssori paraméterek

param1.cpp

```
#include <iostream>
using namespace std;

int main(int argc, char* argv[]) {
  cout << "Az elinditott program neve: "
       << argv[0] << endl;
  if(argc == 1) {
    cout << "Nem adtak meg parametereket.\n";
  } else {
    for(int i=1; i<argc; i++) {
      cout << i << ". parameter: " << argv[i] << endl;
    }
  }
  return 0;
}
```

Parancssori paraméterek

param2.cpp

```
#include <iostream>
using namespace std;

int main(int argc, char** argv) {
  cout << "Az elinditott program neve: "
       << argv[0] << endl;
  if(argc == 1) {
    cout << "Nem adtak meg parametereket.\n";
  } else {
    for(argv++; *argv != NULL; argv++) {
      cout << *argv << endl;
    }
  }
  return 0;
}
```

Parancssori paraméterek

Feladat

Két paraméterrel megadott, zárt intervallumba eső véletlenszám előállítása

Probléma

A paraméterek típusa nem megfelelő

Megoldás1

A `stringstream` használata `char*` $\rightarrow$ `int` konverzióhoz (rugalmasabb)

Megoldás2

C függvény (`atoi`, ASCII to int) használata

Parancssori paraméterek

param3.cpp

```
1  #include <iostream> // cout
2  #include <cstdlib>  // srand, rand
3  #include <ctime>    // time
4  #include <sstream>  // stringstream
5  using namespace std;
6
7  int main(int argc, char* argv[]) {
8    if(argc != 3) {
9      cout << "Hasznalat: " << argv[0] << " min max\n";
10   } else {
11     stringstream ss;
12     int min, max;
13     ss << argv[1] << ' ' << argv[2];
14     ss >> min;
15     ss >> max;
```

Parancssori paraméterek

param3.cpp

```
16      srand(time(NULL));
17      cout << "A [" << min << ", " << max
18           << "] intervallumba "
19           << "eso veletlen szam: "
20           << min + rand()%(max-min+1) << endl;
21    }
22    return 0;
23  }
```

Kimenet

```
wajzy@wajzy-notebook:~/Dokumentumok/gknb_intm021/Elõadások/ea13$
./param3 1 5
A [1, 5] intervallumba eso veletlen szam: 4
```

Parancssori paraméterek

param4.cpp

```
#include <iostream> // cout
#include <cstdlib>  // srand, rand, atoi
#include <ctime>    // time
using namespace std;

int main(int argc, char* argv[]) {
  if(argc != 3) {
    cout << "Hasznalat: " << argv[0] << " min max\n";
  } else {
    int min = atoi(argv[1]);
    int max = atoi(argv[2]);
    srand(time(NULL));
    cout << "A [" << min << ", " << max
         << "] intervallumba "
         << "eso veletlen szam: "
         << min + rand()%(max-min+1) << endl;
  }
  return 0;
}
```

Referenciák

A referencia tekinthető

- új név társításának egy létező változóhoz, vagy
- olyan mutatónak, amelyen automatikusan végrehajtásra kerül az indirekció.

ref1.cpp

```
4  int main() {
5    int darab = 1;            // valtozo
6    int* mutDarab = &darab; // mutato
7    int& refDarab = darab;  // referencia
8    // int& refDarab; // Mi az eredeti valtozo cime?
9    // int& refDarab = 5; // Konstansnak nincs cime
10   cout << "darab=" << darab
11        << " refDarab=" << refDarab
12        << " mutDarab=" << mutDarab
13        << " *mutDarab=" << *mutDarab << endl;
```

Referenciák

ref1.cpp

```
15    darab++;
16    refDarab++;
17    cout << "darab=" << darab
18         << " refDarab=" << refDarab
19         << " mutDarab=" << mutDarab
20         << " *mutDarab=" << *mutDarab << endl;
21    return 0;
22  }
```

Kimenet

```
darab=1 refDarab=1 mutDarab=0x7ffe641fbc0c *mutDarab=1
darab=3 refDarab=3 mutDarab=0x7ffe641fbc0c *mutDarab=3
```

Referenciák

ref2.cpp – Készíthető referenciát fogadó függvény is

```
4  void negyzetErtek(int alap) {
5    alap = alap * alap;
6  }
7
8  void negyzetMutato(int* alap) {
9    *alap = *alap * *alap;
10 }
11
12 void negyzetReferencia(int& alap) {
13   alap = alap * alap;
14 }
```

Referenciák

ref2.cpp – Készíthető referenciát fogadó függvény is

```
16 int main() {
17   int szam = 2;
18   cout << "Kezdetben: szam=" << szam;
19   negyzetErtek(szam);
20   cout << ", negyzetErtek() utan: szam=" << szam;
21   negyzetMutato(&szam);
22   cout << "\nnegyzetMutato() utan: szam=" << szam;
23   negyzetReferencia(szam);
24   cout << ", negyzetReferencia() utan: szam="
25        << szam << endl;
26   // negyzetReferencia(5); // Konstansnak nincs cime
27   return 0;
28 }
```

Kimenet

```
Kezdetben: szam=2, negyzetErtek() utan: szam=2
negyzetMutato() utan: szam=4, negyzetReferencia() utan: szam=16
```

Referenciák

ref3.cpp – Referenciával visszatérő fv.

```
4  int szam = 5;
5
6  int& vissza(int param) {
7    // return 1; // konstansnak nincs cime
8    int lokalis;
9    // return lokalis; // felszabadul, mire hasznalnank
10   // return param;   // detto
11   return szam;
12 }
13
14 int main() {
15   int n = vissza(42)*2;
16   cout << "n=" << n;
17   vissza(42) = 3;
18   vissza(42)++;
19   cout << "\nszam=" << szam;
20   return 0;
21 }
```

Kimenet

```
n=10
szam=4
```

Magas szintű be- és kimenet

Feladat

Írjunk programot, ami a paraméterként adott szövegfájl tartalmát kiírja!

Megoldás

Magas B/K (high-level I/O) használata (operációs rendszertől független, hordozható megoldás, de csak a legfontosabb szolgáltatások érhetőek el)

Fájlkezelés lépései:

1. (Megfelelő típusú folyam) megnyitása
2. B/K műveletek elvégzése
3. Folyam (stream) lezárása

A folyamok hasonlítanak a `cin`/`cout` pároshoz, de azok automatikusan nyílnak/zárulnak!

beolvas1.cpp – Beolvasás karakterenként

```
#include <iostream>
#include <fstream> // ifstream
using namespace std;

int main(int argc, char* argv[]) {
  if(argc != 2) {
    cout << "Hasznalat: " << argv[0] << " fajlnev\n";
  } else {
    ifstream f(argv[1]);
    if(f.is_open()) {
      char c;
      while(f.get(c), !f.eof()) cout << c;
      f.close();
    } else {
      cerr << "Fajl nyitasi hiba.\n";
    }
  }
  return 0;
}
```

Magas szintű be- és kimenet

ifstream f(argv[1]);

`ifstream`: bemeneti folyam típusa (*osztály*)
`f`: a folyamváltozó neve
`argv[1]`: a folyam forrása, ebből fog olvasni

f.is_open()

Ellenőrzi a megnyitás sikerességét (Létezik a fájl? Van jogunk olvasni? . . .)
(Az `ifstream` *osztály tagfüggvénye*)

f.get(c)

Beolvas egyetlen karaktert a `c` változóba
(`istream& get (char& c);`)

f.eof()

Ellenőrzi, hogy a fájl végének (end of file) elérését mutató jelzőbit (flag) magas értékű-e
Csak az első túlolvasási kísérlet után lesz a bit magas értékű!

f.close()

Folyam lezárása

Magas szintű be- és kimenet

Programunk kimenete

```
wajzy@wajzy-notebook:~/Dokumentumok/gknb_intm021/Előadások/ea13$
./beolvas1 guns.txt
She's got a smile it seems to me
Reminds me of childhood memories
Where everything
Was as fresh as the bright blue sky
Now and then when I see her face
```

cat (concatenate files and print) kimenete

```
wajzy@wajzy-notebook:~/Dokumentumok/gknb_intm021/Előadások/ea13$
cat guns.txt
She's got a smile it seems to me
Reminds me of childhood memories
Where everything
Was as fresh as the bright blue sky
Now and then when I see her face
```

Magas szintű be- és kimenet

beolvas2.cpp – Minden szó külön stringben

```
5  int main(int argc, char* argv[]) {
6    if(argc != 2) {
7      cout << "Hasznalat: " << argv[0]
8           << " fajlnev\n";
9    } else {
10     ifstream f(argv[1]);
11     if(f.is_open()) {
12       string s;
13       while(f >> s, !f.eof())
14         cout << s << endl;
15       f.close();
16     } else {
17       cerr << "Fajl nyitasi hiba.\n";
18     }
19   }
20   return 0;
21 }
```

Kimenet

```
She's
got
a
smile
it
seems
to
me
Reminds
me
of
childhood
memories
Where
everything
Was
as
fresh
as
```

Magas szintű be- és kimenet

beolvas3.cpp – Teljes sorok olvasása

```
5  int main(int argc, char* argv[]) {
6    if(argc != 2) {
7      cout << "Hasznalat: " << argv[0] << " fajlnev\n";
8    } else {
9      ifstream f(argv[1]);
10     if(f.is_open()) {
11       string s;
12       while(getline(f, s), !f.eof()) {
13         cout << s << endl;
14       }
15       f.close();
16     } else {
17       cerr << "Fajl nyitasi hiba.\n";
18     }
19   }
20   return 0;
21 }
```

Magas szintű be- és kimenet

kiir1.cpp – Sorok írása

```
1  #include <iostream>
2  #include <fstream> // ofstream
3  using namespace std;
4
5  int main(int argc, char* argv[]) {
6    if(argc != 2) {
7      cout << "Hasznalat: " << argv[0] << " fajlnev\n";
8    } else {
9      ofstream f(argv[1]); // alapértelmezes: feluliras
10     if(f.is_open()) {    //                 (truncate)
11       string dal[] = {
12         "She's got a smile it seems to me",
13         "Reminds me of childhood memories",
```

kiir1.cpp – Sorok írása

```
18            "And if I'd stare too long",
19            "I'd probably break down and cry"
20          };
21          for(unsigned i=0;
22              i<sizeof(dal)/sizeof(dal[0]);
23              i++) {
24            f << dal[i] << endl;
25          }
26          f.close();
27        } else {
28          cerr << "Fajl nyitasi hiba.\n";
29        }
30      }
```

masol1.cpp – Másolás karakterenként

```
5  int main(int argc, char* argv[]) {
6    if(argc != 3) {
7      cout << "Hasznalat: " << argv[0]
8           << " forras cel\n";
9    } else {
10     ifstream be(argv[1]);
11     if(be.is_open()) {
12       ofstream ki(argv[2]);
13       if(ki.is_open()) {
14         char c;
15         while(be.get(c), !be.eof()) {
16           ki << c;
17         }
```

masol1.cpp – Másolás karakterenként

```
18            ki.close();
19         } else {
20            cerr << "Megnyitasi hiba: "
21               << argv[2] << endl;
22         }
23         be.close();
24      } else {
25         cerr << "Megnyitasi hiba: "
26               << argv[1] << endl;
27      }
28   }
29   return 0;
30 }
```

Magas szintű be- és kimenet

masol2.cpp – Másolás blokkonként

```
1  #include <iostream>
2  #include <fstream>
3  using namespace std;
4  #define MERET 65536
5
6  int main(int argc, char* argv[]) {
7    if(argc != 3) {
8      cout << "Hasznalat: " << argv[0]
9           << " forras cel\n";
10   } else {
11     ifstream be(argv[1], ios::binary);
12     if(be.is_open()) {
13       ofstream ki(argv[2], ios::binary);
14       if(ki.is_open()) {
15         char* puffer = new char[MERET];
16         int beolvasva;
```

masol2.cpp – Másolás blokkonként

```
17          do {
18            be.read(puffer, MERET);
19            beolvasva = be.gcount();
20            ki.write(puffer, beolvasva);
21          } while(beolvasva == MERET);
22          delete [] puffer;
23          ki.close();
24        } else {
25          cerr << "Megnyitasi hiba: " << argv[2] << endl;
26        }
27        be.close();
28      } else {
29        cerr << "Megnyitasi hiba: " << argv[1] << endl;
30      }
31    }
32    return 0;
33  }
```

Folyamok egy-egy felültöltött *konstruktora*

```
explicit ifstream (const char* filename,
      ios_base::openmode mode = ios_base::in);
explicit ofstream (const char* filename,
      ios_base::openmode mode = ios_base::out);
```

Megnyitási mód konstansok:

(kombinálhatók *bitenkénti* vagy operátorral, vagy összeadással)

- **in** (input) Megnyitás olvasásra
- **out** (output) Megnyitás írásra
- **binary** Bináris üzemmód (karakteres helyett = nincs *transzláció*)
- **ate** (at end, append) Hozzáfűzés (fájlmutató a fájl végére mozdul)
- **trunc** (truncate) Meglévő tartalom törlése

Magas szintű be- és kimenet

Másolás karakterenként

time ./masol1 nagyfajl.dat masolat.dat
real 5m33.072s
user 4m54.301s
sys 0m19.405s

Másolás blokkonként

time ./masol2 nagyfajl.dat masolat.dat
real 0m53.501s
user 0m0.212s
sys 0m12.795s

Másolás az OS segédprogramjával

time cp nagyfajl.dat masolat.dat
real 0m50.821s
user 0m0.102s
sys 0m12.247s