

Programozás

(GKxB_INTM114)

Dr. Hatwagner F. Miklós

Széchenyi István Egyetem, Győr

https://github.com/wajzy/GKxB_INTM114.git

2024. április 26.

Készítsünk programot, amely kezeli

- autók adatait:
 - nyilvántartja: gyártás dátumát, utolsó műszaki vizsga dátumát, vizsgák darabszámát.
 - kiszámítja: meddig érvényes a vizsga? (Az első 4 évre szól, minden későbbi 2-re.)
- emberek adatait:
 - nyilvántartja: nevet, születési dátumot.
 - kiszámítja: elmúlt-e már 17 éves? (Kaphat-e „B” kat. jogosítványt?)
- majd ezek felhasználásával kiszámítja:
 - van jogosítvány + érv. műszaki v. → mehetünk autózni
 - van jogosítvány + lejárt műszaki v. → le kell műszakiztatni az autót
 - nincs jogosítvány → kicsit várunk még

emberAuto1.cpp

```
1  #include <iostream>
2  #include <iomanip>
3  using namespace std;
4
5  // Datum struktura es muveletek
6
7  struct datum {
8      int ev, ho, nap;
9  };
10
11 bool szoko(int ev) { // szokoev megallapitas
12     return (ev%4==0 and ev%100!=0) or ev%400==0;
13 }
```

emberAuto1.cpp

```
85 // Sofor adatok es muvelete
86
87 struct ember {
88     string nev;
89     datum szuletes;
90 };
91
92 bool elmult17(const ember* e, const datum* ma) {
93     datum eppen17 = e->szuletes;
94     eppen17.ev += 17;
95     return kulonbseg(&eppen17, ma) >= 0;
96 }
```

emberAuto1.cpp

```
98 // Auto adatok és muvelete
99
100 struct gepkocsi {
101     datum gyartas;
102     datum utolsoMuszaki;
103     int muszakiDb;
104 };
105
106 datum muszakiErvenyesseg(const gepkocsi* gk) {
107     datum lejar = gk->utolsoMuszaki;
108     if(gk->muszakiDb > 1) {
109         lejar.ev += 2;
110     } else {
111         lejar.ev += 4;
112     }
113     return lejar;
114 }
```

emberAuto1.cpp

```
116 // Foprogram
117
118 int main() {
119     ember e = { "Gizike", { 2000, 1, 2} };
120     gepkocsi gk = { { 2017, 10, 3}, { 2023, 10, 7}, 2 };
121     datum ma = { 2024, 4, 16 };
122
123     if(elmult17(&e, &ma)) {
124         datum erv = muszakiErvenyesseg(&gk);
125         if(kulonbseg(&ma, &erv) >= 0) {
126             cout << "Hajtsunk a naplementebe!\n";
127         } else {
128             cout << "Foglalkozni kell a verdaval.\n";
129         }
130     } else {
131         cout << "Tul fiatal vagy meg a vezeteshez.\n";
132     }
133
134     return 0;
135 }
```

Eddigi programjaink egyetlen forrásfájlból álltak, ld. `emberAuto1.cpp`

Problémák:

- Áttekinthetetlenül nagyra nőnek a forrásszövegek
- Több programozó együttes munkája nehézkes

Megoldás:

- modularizáció: több fejfájl (`.h`, `.hpp`) és forrásfájl (`.cpp`) → egy program
 - Fejfájlok: szimbolikus állandók, adattípusok (pl. struktúrák), fv. prototípusok
 - Forrásfájlok: függvény definíciók
- {fej,forrás}fájlok összekapcsolása: *projekt* segítségével (ld. `make`^{1,2}, `CMake`^{1,2}, `Conan`)

Alakítsuk át az iménti programunkat!

Régi módszer:

emberAuto1.cpp

```
bool szoko(int ev) { /* ... */ }
int napok(int ev, int ho) { /* ... */ }
bool ellenoriz(const datum* d) { /* ... */ }
int evNapja(const datum* d) { /* ... */ }
string hetNapja(const datum* d) { /* ... */ }
int bazis(const datum* d, int bazisEv=0) { /* ... */ }
int kulonbseg(const datum* tol, const datum* ig) { /* ... */ }
datum hoEsNap(int ev, int evNapja) { /* ... */ }
void nyomtat(const datum* d) { /* ... */ }

bool elmult17(const ember* e, const datum* ma) { /* ... */ }

datum muszakiErvenyesseg(const gepkocsi* gk) { /* ... */ }

int main() { /* ... */ }
```


Új módszer:

```
#include "ember2.h" ↗  
#include "datum2.h" →  
#include "gepkocsi2.h" ↘
```

emberAuto2.cpp

```
int main() { /* ... */ }
```

ember2.h

```
bool elmult17(const ember* e, const datum* ma);
```

↓ #include "datum2.h"

datum2.h

```
bool szoko(int ev);  
int napok(int ev, int ho);  
bool ellenoriz(const datum* d);  
int evNapja(const datum* d);  
string hetNapja(const datum* d);  
int bazis(const datum* d, int bazisEv);  
int kulonbseg(const datum* tol, const datum* ig);  
datum hoEsNap(int ev, int evNapja);  
void nyomtat(const datum* d);
```

↑ #include "datum2.h"

gepkocsi2.h

```
datum muszakiErvenyesseg(const gepkocsi* gk);
```

Régi módszer:

Fordítás (compile)

```
g++ -Wall -c emberAuto1.cpp
```

Kapcsoló-szerkesztés (link)

```
g++ -o emberAuto1 emberAuto1.o
```

Összeállítás (build)

```
g++ -Wall -o emberAuto1 emberAuto1.cpp
```

Új módszer:

Fordítás (compile)

```
g++ -Wall -c datum2.cpp ember2.cpp gepkocsi2.cpp emberAuto2.cpp
```

Kapcsoló-szerkesztés (link)

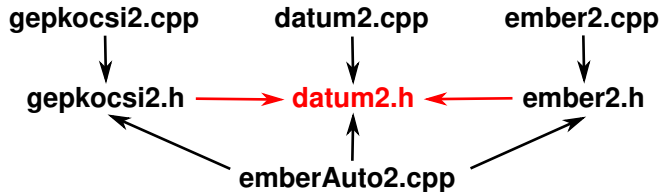
```
g++ -o emberAuto2 datum2.o ember2.o gepkocsi2.o emberAuto2.o
```

Összeállítás (build)

```
g++ -Wall -o emberAuto2 datum2.cpp ember2.cpp gepkocsi2.cpp emberAuto2.cpp
```

Probléma: újradefiniált típus

```
g++ -Wall -o emberAuto2 datum2.cpp ember2.cpp gepkocsi2.cpp emberAuto2.cpp
In file included from gepkocsi2.h:3,
      from emberAuto2.cpp:2:
datum2.h:8:8: error: redefinition of 'struct datum'
      8 | struct datum {
        |      ~~~~~
In file included from ember2.h:3,
      from emberAuto2.cpp:1:
datum2.h:8:8: note: previous definition of 'struct datum'
      8 | struct datum {
        |      ~~~~~
In file included from emberAuto2.cpp:3:
datum2.h:8:8: error: redefinition of 'struct datum'
      8 | struct datum {
        |      ~~~~~
In file included from ember2.h:3,
      from emberAuto2.cpp:1:
datum2.h:8:8: note: previous definition of 'struct datum'
      8 | struct datum {
        |      ~~~~~
```



Források: `gepkocsi2.cpp`, `datum2.cpp`, `ember2.cpp`, `emberAuto2.cpp`

Fejfájlok: `gepkocsi2.h`, `datum2.h`, `ember2.h`

Feltételes fordítás: feltételektől függően bizonyos programrészek megőrzése/kihagyása

Feltétles fordítás

```
#if konstans-kifejezés1 <szekció1>
<#elif konstans-kifejezés2 <szekció2>>
/* ... */
<#elif konstans-kifejezésN <szekcióN>>
<#else <végső-szekció>>
#endif
```

- A konstans-kifejezések típusa logikainak tekintendő.
- Csak karakter és egész állandókat, defined operátort tartalmazhat

A defined operátor

- célja: makrók definiáltságát ellenőrzi
- alakjai:
`defined(azonosító)`
`defined azonosító`
- eredmény: logikai, és logikai operátorokkal együtt használható
- alkalmazása: pl. biztosíthatja, hogy egy fejfájl egyszer kerülhessen csak beépítésre (`include/header guard`), platform-specifikus megoldások közül egynek a használata

fej.h

```
#if !defined(FEJ)
#define FEJ
/* érdemi tartalom */
#endif
```

Az `#ifdef`, `#ifndef` direktívák

- céljuk: makró definiáltságát/definiátlanságát ellenőrzik
- `#if defined(azonosító) ≡ #ifdef azonosító`
- `#if !defined(azonosító) ≡ #ifndef azonosító`

```
fej.h
```

```
#ifndef FEJ
#define FEJ
/* érdemi tartalom */
#endif
```

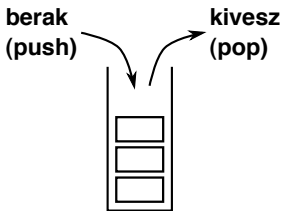

datum3.h

```
1 #ifndef DATUM_H
2 #define DATUM_H
3
4 // Datum struktura es muveletek
5
6 #include <iostream>
7 #include <iomanip>
8
9 using namespace std;
10
11 struct datum {
12     int ev, ho, nap;
13 };
14
15 bool szoko(int ev); // szokoev megallapitas
16
17
18
19
20
21
22
23 void nyomtat(const datum* d); // formazott nyomtatas
24
25 #endif
```

datum3.cpp, ember3.cpp, ember3.h, gepkocsi3.cpp, gepkocsi3.h, emberAuto3.cpp

Verem (stack)

- LIFO (Last In, First Out) szervezésű tár: tárolás sorrendjével ellentétes sorrendben férünk hozzá az adatokhoz
- Műveletek: berak (push), kivesz (pop), kukucskál (peek), ürít (clear), ...
- Alkalmazási terület: pl. böngészési előzmények listája (history) egy webböngészőben



verem1.h

```
1  #ifndef VEREM_H
2  #define VEREM_H
3
4  #include <string>
5  #define VEREM_MAX 128
6
7  bool berak(std::string s);
8  std::string kivesz();
9  bool ures();
10
11 #endif
```

verem1.cpp

```
1 #include "verem1.h"
2 using namespace std;
3
4 // hatokor korlatozas a verem1.cpp-re
5 static string verem[VEREM_MAX];
6 static int n = 0;
7
8 bool berak(string s) {
9     if(n < VEREM_MAX) {
10         verem[n] = s;
11         n++;
12         return true;
13     } else {
14         return false;
15     }
16 }
```

verem1.cpp

```
18 string kivesz() {
19     if(n > 0) {
20         n--;
21         return verem[n];
22     } else {
23         return "";
24     }
25 }
26
27 bool ures() {
28     return n==0;
29 }
```

veremTeszt1.cpp

```
1  #include <iostream>
2  #include "verem1.h"
3
4  using namespace std;
5
6  int main() {
7      berak("alma");
8      berak("barack");
9      berak("citrom");
10
11     while(not ures()) {
12         cout << kivesz() << endl;
13     }
14 }
```

Kimenet

```
citrom
barack
alma
```

Sor (queue)

- First In First Out (FIFO) adatszerkezet
- Alkalmazása: pl. felhasználók egymás után küldenek nyomtatási feladatokat, amiket a nyomtató a sorba rakás sorrendjében hajt végre

sor1.h

```
1 #ifndef SOR_H
2 #define SOR_H
3
4 #include <iostream>
5 #define SOR_MAX 4
6
7 bool berak(int adat);
8 int kivesz();
9
10 #endif
```

sor1.cpp

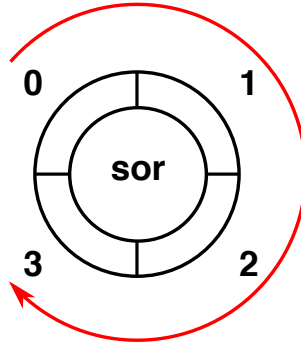
```
1 #include "sor1.h"
2
3 static int sor[SOR_MAX];
4 static int eleje=0, vege=0, db=0;
5
6 bool berak(int adat) {
7     if(db < SOR_MAX) {
8         sor[vege] = adat;
9         db++;
10        vege++;
11        if(vege == SOR_MAX) {
12            vege = 0;
13        }
14        return true;
15    } else {
16        std::cerr << "A sor megtelt.\n";
17        return false;
18    }
19 }
```


sor1.cpp

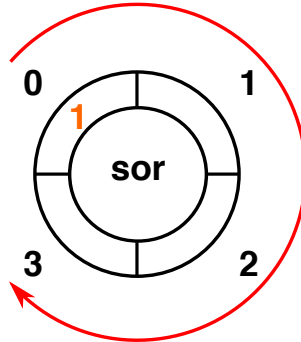
```
21 int kivesz() {
22     if(db > 0) {
23         int adat = sor[eleje];
24         db--;
25         eleje++;
26         if(eleje == SOR_MAX) {
27             eleje = 0;
28         }
29         return adat;
30     } else {
31         std::cerr << "A sor ures.\n";
32         return 0;
33     }
34 }
```

sorTeszt1.cpp

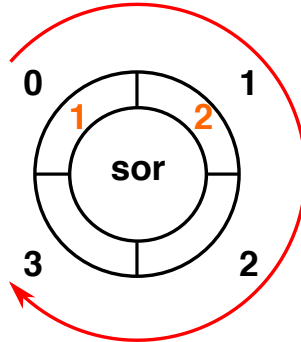
```
1  #include <iostream>
2  #include "sor1.h"
3  using namespace std;
4
5  int main() {
6      berak(1); berak(2); berak(3); berak(4);
7      berak(5); // nem fer bele
8      cout << kivesz() << '\n';
9      cout << kivesz() << '\n';
10     berak(6);
11     cout << kivesz() << '\n';
12     cout << kivesz() << '\n';
13     cout << kivesz() << '\n';
14     // nincs mit kivenni
15     cout << kivesz() << '\n';
16     return 0;
17 }
```



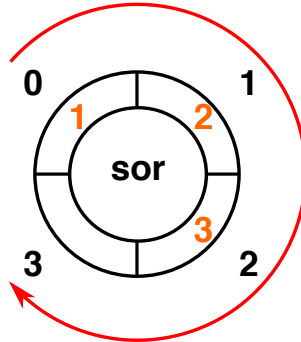
eleje == 0
db == 0
vege == 0



eleje == 0
db == 1
vege == 1



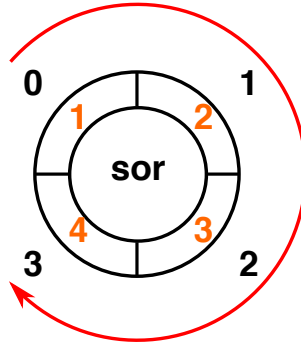
eleje == 0
db == 2
vege == 2



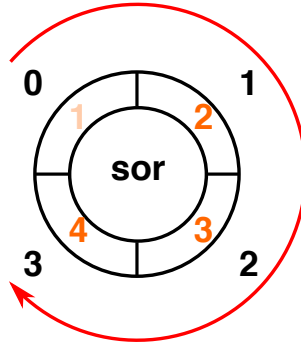
eleje == 0

db == 3

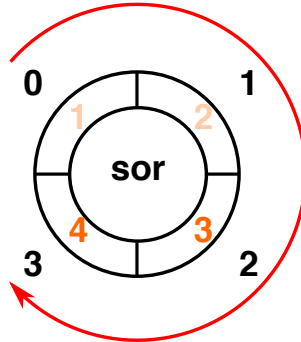
vege == 3



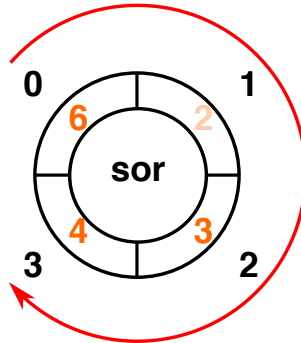
eleje == 0
db == 4
vege == 0



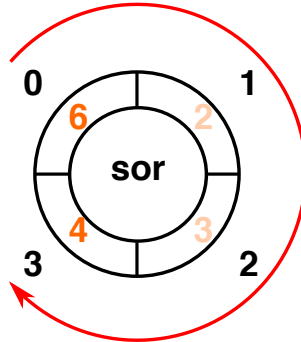
eleje == 1
db == 3
vege == 0



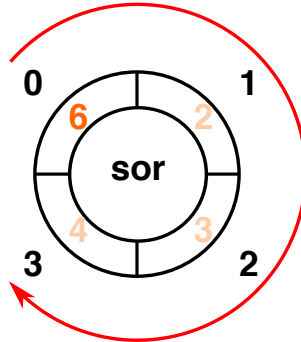
eleje == 2
db == 2
vege == 0



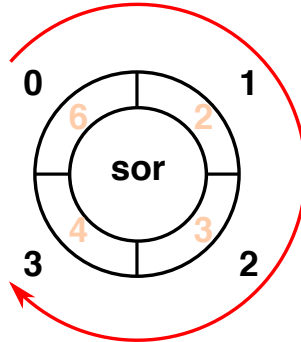
eleje == 2
db == 3
vege == 1



eleje == 3
db == 2
vege == 1



eleje == 0
db == 1
vege == 1



eleje == 1
db == 0
vege == 1