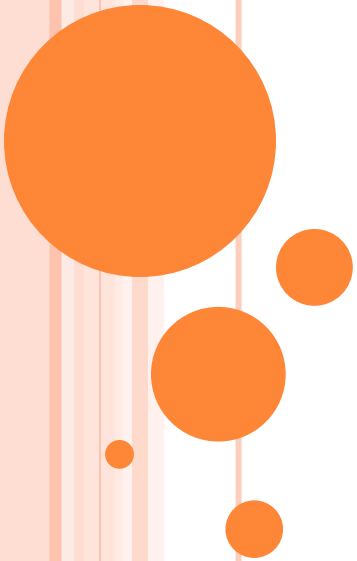


Programovacie jazyky pre vstavané systémy

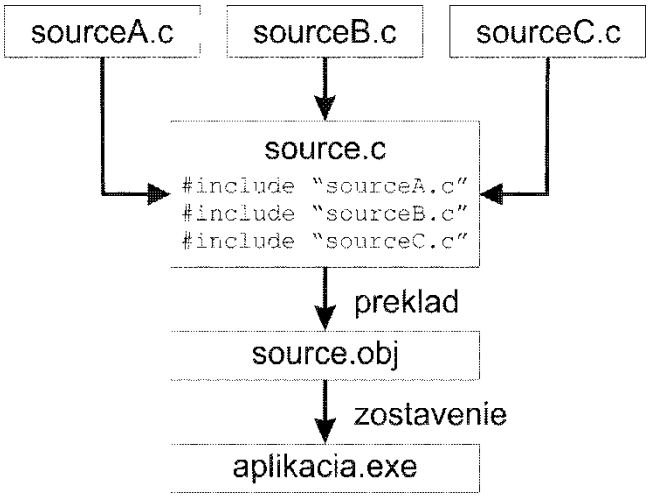
Cvičenie 8

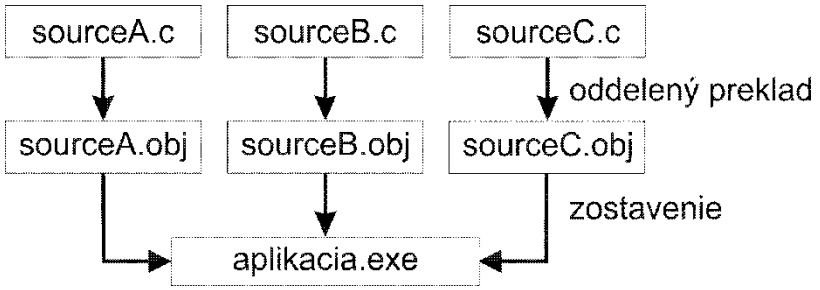
Náplň cvičenia

1. Oddelený preklad a hlavičkové súbory.
2. Dvojrozmerné polia.
3. Maticová algebra.
4. Hra život.

Oddelený preklad súborov

- Direktíva #include slúži na vloženie obsahu jedného súboru (ľubovoľného typ) do iného.
- V prípade, že máme projekt pozostávajúci z viacerých zdrojových súborov (.c), máme dve možnosti pre vytvorenie výslednej aplikácie:
  - pomocou direktívy #include vložiť obsah všetkých zdrojových súborov do jedného súboru a tento preložiť;
  - preložiť každý súbor zvlášť a preložené súbory spolu zlinkovať – tzv. **oddelený preklad**.





Hlavičkové súbory

- Hlavičkové súbory sa používajú na uvedenie dopredných:
  - deklarácií globálnych premenných (napr. errno),
  - deklarácií funkcií,
  - definícií (deklarácií) dátových typov,
  - definícií makier a symbolických konštánt.
- Pozor na použitie kľúčového slova static v hlavičkových súboroch!!!

- Štandardný konštrukt v hlavičkovom súbore:

```
#ifndef HEADER_FILE
#define HEADER_FILE

celý obsah hlavičkového súboru

#endif
```

Dvojrozmerné polia v jazyku C

- V jazyku C (ak neuvažujeme VLA) existujú 4 spôsoby, ako môžeme vytvoriť dvojrozmerné pole:
  - int maticaA[2][3];
  - int* maticaB[2];
  - int (*maticaC)[3];
  - int **maticaD;
- Aký je rozdiel medzi vyššie uvedenými premennými?
- Ako sa implementuje v jazyku C pole reťazcov?

Úlohy – Maticová algebra (1)

- Implementujte maticovú algebru pre objekty typu float.
- Maticu implementujte ako dvojrozmerné pole, ktoré môže mať ľubovoľný počet riadkov a stĺpcov.
- Implementujte nasledujúce funkcie:
  - float** create(int m, int n) – vytvorí a vráti maticu o m riadkoch a n stĺpcoch. Všetky prvky matice budú mať hodnotu 0;
  - void delete(int m, int n, float **matrix) – zruší maticu, ktorá má m riadkov a n stĺpcov (Ako by mal vyzerať funkčný prototyp, ak by sme chceli smerník matrix nastaviť vo funkcii na hodnotu NULL?);
  - float** set(int m, int n, float **matrix, float value) – nastaví všetky hodnoty v matici matrix na hodnotu value a vráti smerník na maticu matrix;
  - float** random(int m, int n, float **matrix, float min, float max) – nastaví všetky hodnoty v matici matrix na náhodnú hodnotu z intervalu <min, max) a vráti smerník na maticu matrix;
  - void print(int m, int n, const float **matrix) – vytlačí obsah matice matrix na štandardný výstup;
  - float** sum(int srcAM, int srcAN, const float **srcA, int srcBM, int srcBN, const float **srcB, int destM, int destN, float **dest) – vypočíta súčet matíc A a B a uloží ho do matice dest, pričom vráti adresu matice dest (v prípade neúspechu vráti hodnotu NULL);
  - float** dotProduct(int srcAM, int srcAN, const float **srcA, int srcBM, int srcBN, const float **srcB, int destM, int destN, float **dest) – vypočíta súčin matíc A a B a uloží ho do matice dest, pričom vráti adresu matice dest (v prípade neúspechu vráti hodnotu NULL);
  - float** readFromTxt(const FILE *txtFile, int *m, int *n) – načíta (a vytvorí) maticu, ktorej obsah je uložený v textovom súbore, funkcia vráti adresu matice (v prípade neúspechu hodnotu NULL) a počet načítaných riadkov a stlpcov;
  - void writeToTxt(int m, int n, float **matrix, FILE *txtFile) – funkcia uloží maticu do textového súboru.

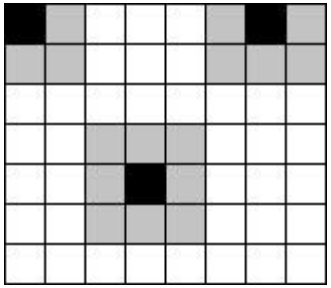
Úlohy – Maticová algebra (2)

- Ako by vyzerali predchádzajúce funkcie, ak by sme vedeli, že matica má vždy:
  a) 5 riadkov
  b) 5 stĺpcov

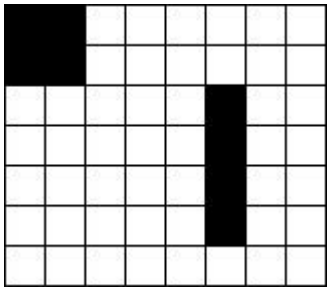
- Čo je nevhodné (komplikované) na predchádzajúcej implementácii? Ako by vyzeralo elegantnejšie riešenie knižnice pre maticovú algebru?

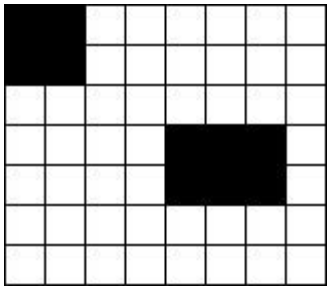
Úlohy – Hra život (1)

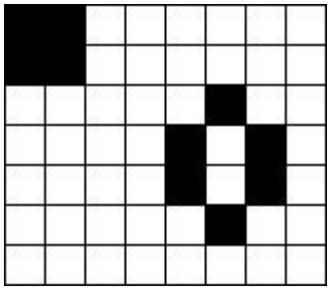
- Hra život predstavuje jednoduchý bunkový automat, ktorý svojim správaním pripomína vývoj spoločenstva živých organizmov. Základným stavebným prvkom sú bunky, ktoré žijú v dvojrozmernom svete modelovanom ako matica. Každá bunka je buď živá, alebo mŕtva. Na začiatku sa určí, ktoré bunky sú živé, a v každom ďalšom kroku sa na základe jednoduchých pravidiel vypočíta, ktoré bunky budú živé a ktoré budú mŕtve. Spomínané pravidlá sú nasledujúce:
  1. Živá bunka zomrie, ak má menej ako 2 alebo viac ako 3 živých susedov.
  2. Mŕtva bunka ožije, ak má práve 3 živých susedov.
  3. Ak nenastane žiadna z vyššie uvedených možností, stav bunky sa nezmení.
- Pod susedmi bunky rozumieme tie bunky, ktoré sa nachádzajú v jej okolí:



- Ukážka vývoja simulácie na základe vyššie popísaných pravidiel:







Úlohy – Hra život (2)

- Úlohou je implementovať hru život, ktorá sa bude ovládať z konzoly. Hra sa bude spúšťať s dvomi argumentmi (z príkazového riadka), ktoré budú predstavovať rozmery matice, v ktorej bude prebiehať simulácia. V rámci ovládania bude možné:
  - náhodne vygenerovať maticu sveta (zadá sa približný podiel živých buniek);
  - oživiť alebo umŕtviť konkrétnu bunku;
  - posunúť sa o 1 až n krokov **dopredu** (s možnosťou vykresľovania);
  - posunúť sa o 1 až n krokov **dozadu** (s možnosťou vykresľovania).

- Hru implementujte tak, že pri každom posune dopredu si budete do nejakého zoznamu ukladať aktuálnu maticu sveta a posun dozadu budete riešiť len spätným prechádzaním tohto zoznamu.

- Požadované štruktúry:
  - matica buniek;
  - zoznam matíc buniek (DynArray alebo LinZoz).