Programování v C++

přetypování a ANSI C

Pavel Fibich

pavel.fibich@prf.jcu.cz

5-1-2011

přetypování

Změna typu proměnné

- implicitní - neřešíme

```
short a=2000;
int b;
b=a;
```

- explicitní - kompilátor nás donutí

```
short a=2000;
int b;
b = (int) a;
b = int (a);
```

- typeid - jakého typu je a

```
typeid(a).name()
```

hello a vystup v C

```c
#include <stdio.h>

int main() {
 printf("Hello World");
 return 0;
}
```

```
gcc hello.c -o hello.exe
```

```c
#include <stdio.h>

int main() {
 int i;
 char * str = "the variable i is";
 float d = 3.14;
 for (i = 0; i < 10; i++)
   printf("%s : %d , obsah %f \n", str, i, i*i*d);
 return 0;
}
```

V C nejsou proudy.

vstup a výstup v C

```c
#include <stdio.h>

int main() {
 int a,b;
 char * str = "zadej";
 printf("%s a",str);
 scanf("%d",&a);
 printf("%s b",str);
 scanf("%d",&b);
 printf("soucin %d \n",a*b);
 return 0;
}
```

soubory v C

```c
#include <stdio.h>

int main()
{
   FILE * pFile, *wFile;
   char mystring [100];

   pFile = fopen ("test4.c" , "r");
   wFile = fopen ("vystup.dat" , "w");

   if (pFile == NULL) perror ("Error opening file");
   else {
     fgets (mystring , 100 , pFile);
     fputs (mystring,wFile);
     fclose (pFile);
   }
   fclose(wFile);
   return 0;
}
```

alokace a deklarace v C

V C nejsou `new` a `delete` operátory. Alokace se řeší pomocí

```
void * malloc(int nbytes)
```

```
int *foo;
foo = (int *) malloc(sizeof(int) * 5);
foo[0] = 17;
foo[4] = 22;
```

Uvolnění paměti se řeší pomocí

```
void free(void *);
```

```
free(foo);
```

Deklarace proměnných se musí provést na začátku funkce, tzn.
NE `for(int i = 0; i < 200; i++)`
Řeší to použitím normy C99

```
cc run.c -std=c99 -o run.exe
```

co je v C jiné oproti C++

- nejsou třídy, templaty, dědičnost
- knihovny mají koncovku .h
- není knihovna `iostream` a tedy ani proudy (cout, fstream)
- není `new` a `delete` používá se malloc
- není bool
- nejsou tak velké problémy s kompatibilitou

Nahradni ukol

- pro ty co nemají z úkolů dost bodů
- pro opravení úkolu musíte napsat mail
- odvezdání přes ‣ http://moodle.prf.jcu.cz/course/view.php?id=16
- 0-10 bodů + 5 za přidání funkcionality
- odevzdáte zabalený (.zip nebo .tar.gz) soubor: komentovaný zdrojový soubor (.cpp) a z něj přeložený soubor (.exe)
- na začátku .cpp souboru bude zakomentovaná a vyplněná následující hlavička

```
AUTOR: (jmeno prijmeni, osobni cislo zacinajici B)
EMAIL:
SKUPINA: (kam chodite a jste zapsani na cviceni)
VERZE G++: (ziskate pomoci prikazu g++ -v)
PRIKAZ PREKLADU: (jaky prikazem byl program prelozen)
OS: (operacni system, napr. Win 7 64b)
POZNAMKA:
```

Náhradní úkol - zadání

Napište simulátor pivního stánku. Program bude akceptovat právě 3 pozitivní a nenulové argumenty (v rozsahu short):

- 1. arg. (**x**) určí maximální počet kroků k natočení jednoho piva
- 2. arg. (**y**) určí maximální počet kroků než přijde další pivař
- 3. arg. (**z**) určí počet kroků programu

Pivaři budou přicházet ke stánku a řadit se do fronty v náhodném počtu kroků v intervalu 1-**y** kroků. Ve frontě se nepředbíhá. Každý pivař chce jedno pivo a bude označen jinak (unikátně) což se ukáže ve výpisu. Výčepní bude točit vždy jedno pivo a to v náhodném počtu kroků v intervalu 5-**x** kroků. Program poběží v **z** krocích a v každém kroku se přehledně vypíše:

- počet pivařů ve frontě,
- kdo všechno čeká ve frontě (označení pivařů)

Každý běh programu bude mít i pro stejné parametry různý výstup (díky náhodnému generování)!

Náhradní úkol - bonusy

Můžete dostat bonusové body (až 5) když přidáte

- na konec simulace min,max a avg dobu čekání na pivo
- aby kroky odpovídaly vteřinám
- použití jiného generátoru náhodných čísel
- pivař může chtít víc piv (další náhodné číslo)
- výčepní točí víc piv za ráz

Na takové rozšíření upozorněte v poznamce v hlavičce.