

6 – Paměťové třídy

Příklad 6.1

Následující program ukazuje, jak vnitřní proměnná v bloku překryje (zastíní) proměnnou ve více vnějším bloku. Zároveň je ukázkou toho, jak je velmi nevhodné a nebezpečné nazývat více proměnných stejnými názvy.

```
# include <stdio.h>
int t = 10;

int main()
{
printf ("1. %d \n",t);
{
int t =20, r =100;
printf ("2. %d %d \n ",t,r);
{
int t = 30;
printf ("3. %d %d \n",t,r);
}
printf ("4. %d %d \n", t,r );
}
printf ("5. %d \n ",t);
}
```

Příklad 6.2

Tento program je pouze velmi lehkou obměnou předcházejícího. Pouze místo `int t = 30` je užito `extern int t`. Zjistěte, k jaké došlo změně, odůvodněte.

```
# include <stdio.h>
int t= 10;

int main()
{
printf ("1. %d \n",t);
{
int t =20,r =100;
printf ("2. %d %d \n ",t,r);
{
extern int t ;
printf ("3. %d %d \n",t,r);
}
printf ("4. %d %d \n", t,r );
}
printf ("5. %d \n ",t);
}
```

Příklad 6.3

V ANSI C je následující část programu chybně, neboť ANSI C nedovoluje deklarovat v této situaci znovu identifikátor `x`. Některé dřívější implementace C toto dovolily. Zkuste z dané části vytvořit celý program, odladit v ANSI C a pokud máte možnost i v jiném C a zjistěte zda a jaké jsou hlášeny chyby.

```
int f(x) /* nebo místo těchto dvou radku int f (int x) */
int x;
{
long x = 77; /* v ANSI C nedovolené */
return x;
}
```

Příklad 6.4

Následující blok opakujte 10 krát (např. pomocí for nebo jiným cyklem) a zjistěte, co bude program vypisovat. Nejprve proveďte odhad, poté naprogramujte. Odůvodněte výsledek.

```
{
    static int Z = 0;
    auto int K = 0;
    K = K + 1;
    Z = Z + 1;
    printf (" K = %d, Z = %d \n ", K, Z);
}
```

Příklad 6.5

Pozor na následující skutečnost. Skočíme-li pomocí goto doprostřed bloku (což se obecně velmi nedoporučuje) , objdeme všechny počáteční inicializace a automatické proměnné se nebudou inicializovat na očekávané hodnoty. Situaci objasňuje následující program.

```
#include<stdio.h>
main()
{ goto SKOK;
  int i;
  static int vector[10]= { 11,12,13,14,15,16,17,18,19,20};
  int suma = 0;
SKOK:
  /* scitam prvky pole vector */
  for (i=1; i <10; i++) suma += vector[i];
  printf("suma je %d",suma);
}
```

Některá cečka ale toto vůbec nedovolí (např. Borland C). Pokud ale implementace takový skok povoluje, je obejita inicializace proměnné suma a program počítá hlouposti.

Příklad 6.6

Je -li oblastí platnosti blok, poté jméno daného objektu lze použít od místa, kde byl objekt deklarován, až do konce bloku. Tedy nikoliv od začátku bloku. V následujícím programu jsou v bloku D odkazy na dvě proměnné, které se obě jmenují i. První i je celočíselná a je deklarována ve vnějším bloku. Ta je užita (obvykle) pro inicializaci proměnné j. Poté je deklarována další proměnná i (float) a ta překryje první proměnnou i. Tato situace ale není jednoznačná a záleží na kompilátoru, jak situaci řeší. Proto je lépe se dané situaci rovnou vyhnout používáním odlišných jmen proměnných.

```
#include<stdio.h>
main()
{
  int i = 0;
  D:{
    int j =i;
    printf("%f%f\n",i,j);
    float i =10.0;
    printf("%f%f\n",i,j);
  }
}
```

Příklad 6.7

Jak se liší chod následujících programů?

```
#include <stdio.h>
```

```
void increment(void)
{
  int j=1;
```

```
static int k=1;

j++;
k++;
printf( "j: %d\tk: %d\n", j, k );
}

int main( void )
{
    increment();
    increment();
    increment();
}
```

```
#include <stdio.h>
```

```
void increment(void)
{
    int j;
    static int k;

    j++;
    k++;
    printf( "j: %d\tk: %d\n", j, k );
}
```

```
int main( void )
{
    increment();
    increment();
    increment();
}
```

Příklad 6.8

Co vytiskne následující program?

```
#include <stdio.h>
```

```
int j=10; /* Oblast platnosti je program */
```

```
int main( void )
{
    int j; /* Oblast platnosti je blok, dojde k zastineni vnejsiho j */
    for (j=0; j < 8; ++j)
        printf( "j: %d", j );
}
```

Příklad 6.9

Program ukazuje situaci, kdy potřebujeme ve volané funkci mít proměnnou, která si mezi voláním funkce podrží svoji hodnotu. Zde se jedná o proměnnou `page_type`, která nabývá hodnoty ODD nebo EVEN. Funkce vkládá na sudé stránky jinou hlavičku než na liché. Je volána na začátku každé stránky. Mezi voláním funkce se hodnota proměnné uchovává, neboť je static. Jedná se o tzv. pevnou lokální proměnnou.

```
#define ODD 0
#define EVEN 1
```

```
tiskni_hlavicku( char *nazev_kapitoly)
{
    static char page_type = ODD;
```

```

if (page_type == ODD)
{
    printf( "\t\t\t\t%s\n\n", nazev_kapitoly );
    page_type = EVEN;
}
else
{
    printf( "%s\n\n", nazev_kapitoly );
    page_type = ODD;
}
}

```

Příklad 6.10

Napište funkci, která tiskne n znaků +, n je číslo, pokudikáté je funkce volána. Např. funkce byla postupně zavolána 4 krát, vytisklo se

```

+
++
+++
++++

```

Příklad 6.11

Pro následující deklarace vždy určete, zda se jedná o definici nebo prototyp (popis)

```

int k;
float z = 2.5;
extern char *p = "vanoce";
static int b;
char *f1( arg1, arg2)
register int arg1;

double arg2;
{
    extern float z;
    extern char *p2;
    static long int dlouhy_int;
    register long t3;
    insigned u4;
.....
}

```

```
int k = 0;
```

Příklad 6.12

Napište deklarace pro následující. Přidejte závorky, aby bylo jasné, zda se jedná o deklaraci uvnitř či vně bloku.

```

automatická lokální int
pevná lokální float
globální pointer na char inicializovaný null pointrem
int s platností modulu
argument int třídy register
double s modifikátorem const
konstatní pointer na double
pointer na const double
int s modifikátorem volatile

```

Příklad 6.13

Určete, kdy se jedná o definici, popis (prototyp) nebo předběžnou definici. Také určete, co jsou inicializované hodnoty proměnných.

```

static int s = 4;
int x = 8;

```

```
extern int x1;
```

```
main()
{
    static float k = x +6;
    float j1 = x +x1;
    static float x2;
    float x3;
    register int z2;
}
```

```
extern int x4 = 6;
```

Příklad 6.14

V následujícím programu najděte chyby, opravte, zdůvodněte

```
#include<stdio.h> /* obsahuje chyby */
int d = 12 * 9;
int c = d;
```

```
main()
{
    int j = 0, k= 1;
    int m = j +k;
    float x= 3.14 * 2;
    int p = r + s;
    int r =8, s =6;
    printf("hodnota d je %d",d);
    printf("hodnota c je %d",c);
}
```

Od RNDr. Dagmar Brechlerové.