

Procedūrinio programavimo pagrindai

Saugojimo klasės

lekt. Irmantas Radavičius

irmantas.radavicius@mif.vu.lt

Informatikos institutas, MIF, VU

Turinys

Saugojimo klasės (angl. storage classes)

Konstantos

Kiti specifikatoriai

<code>auto</code>	<code>double</code>	<code>int</code>	<code>struct</code>
<code>break</code>	<code>else</code>	<code>long</code>	<code>switch</code>
<code>case</code>	<code>enum</code>	<code>register</code>	<code>typedef</code>
<code>char</code>	<code>extern</code>	<code>return</code>	<code>union</code>
<code>const</code>	<code>float</code>	<code>short</code>	<code>unsigned</code>
<code>continue</code>	<code>for</code>	<code>signed</code>	<code>void</code>
<code>default</code>	<code>goto</code>	<code>sizeof</code>	<code>volatile</code>
<code>do</code>	<code>if</code>	<code>static</code>	<code>while</code>

Gyvavimo ciklas

Kintamojo gyvavimo ciklas (angl. lifecycle)

- sukūrimas
- inicializacija
- panaudojimas
- atlaisvinimas

Kintamieji

- automatiniai (gyvuoja tam tikrame bloke)
- statiniai (gyvuoja visoje programoje)

Galiojimo sritis

Kintamojo galiojimo sritis (angl. scope)

- lokalūs kintamieji
- globalūs kintamieji

Lokalūs kintamieji

- galioja tam tikrame funkcijos bloke
- galioja visoje funkcijoje

Globalūs kintamieji

- galioja visame faile
- galioja visoje programoje

Saugojimo klasė

Saugojimo klasė (angl. storage class) nusako gyvavimo ir galiojimo ypatybes.

Gyvavimas (angl. lifetime) ir galiojimas (angl. scope) nėra tas pats!

Kompiliavimo etapas

- naudojamas elementas (f-ja, kintamasis) turi būti aprašytas

Ryšių redagavimo etapas

- naudojamas elementas (f-ja, kintamasis) turi būti apibrėžtas

Elemento (f-jos, kintamojo) aprašymas (angl. declaration) ir apibrėžimas (angl. definition) nėra tas pats!

Auto

Lokalių kintamųjų saugojimo klasė pagal nutylėjimą.
Atmintis išskiriama vykdymo metu. Pradinė reikšmė – šiukšlės.
Gyvuoja ir galioja tame bloke, kuriame buvo aprašytas.

```
int main(){  
    int i;  
    auto char c;  
}
```

```
int main(){  
    int a=10;  
    {  
        int a=20;  
        printf("%d",a);  
    }  
    printf(" %d",a);  
    return 0;  
}
```

```
int main(){  
    int i;  
    for(i=0;i<4;i++){  
        int a=20;  
        printf("%d",a);  
        a++;  
    }  
    return 0;  
}
```

Register

Pasižymi beveik visomis automatinių kintamųjų savybėmis.
Jei įmanoma, kompiliatorius stengiasi saugoti kintamąjį procesoriuje.
Bandymas gauti tokio kintamojo adresą – kompiliavimo klaida.

```
int main() {  
    register int a=10;  
    int *p;  
    p=&a;  
    printf("%u",p);  
}
```

Extern (1)

Globalių kintamųjų saugojimo klasė pagal nutylėjimą.

Jei pridedamas raktažodis `extern`, apibrėžimas pavirsta tik aprašu.

Tokiu atveju, norint apibrėžti kintamąjį, reikia jį inicializuoti.

```
int i;
int main(){
    printf("%d",i);
    return 0;
}
```

```
extern int i;
int main(){
    printf("%d",i);
    return 0;
}
```

```
extern int i=10;
int main(){
    printf("%d",i);
    return 0;
}
```


Extern (2)

Inicializuoti galima tik globaliai. Pagal nutylėjimą inicializuojama 0. Jei yra kintamojo aprašas, ieškoma galiojančio inicializuoto kintamojo. Tokio kintamojo neradus – klaida.

```
int main() {  
    extern int i=10;  
    printf("%d",i);  
    return 0;  
}
```

```
int main() {  
    extern int i;  
    printf("%d",i);  
    return 0;  
}
```

```
int main() {  
    extern int i;  
    printf("%d",i);  
    return 0;  
}  
int i=20;
```

Extern (3)

Inicializuoti galima tik vieną sykį, aprašyti – daug.

Globalus priskyrimas negalimas (priskyrimas nėra tapatu inicializacijai).

Gyvuoja ir pasiekiami visos programos ribose.

```
extern int i;
int i=25;
extern int i;
int main(){
    extern int i;
    printf("%d",i);
    return 0;
}
```

```
extern int i;
int i=10;
i=25;
int main(){
    printf("%d",i);
    return 0;
}
```

```
//one.c
int i=25;
int j=5;

//two.c
extern int i;
extern int j;
void sum(){
    int s;
    s=i+j;
    printf("%d",s);
}
```

Static

Tai nėra globalių kintamųjų klasė pagal nutylėjimą!

Statiniai kintamieji gyvuoja programoje ir inicializuojami vieną kartą!

Globalūs kintamieji matomi viename faile, o lokalūs – bloke.

```
static int a;  
int main(){  
    printf("%d",a);  
    return 0; }
```

```
int i;  
int main(){  
    printf("%d",i);  
    return 0; }
```

```
extern int i;  
int main(){  
    printf("%d",i);  
    return 0; }
```

```
static int i=10;  
int main(){  
    i=5;  
    for(i=0;i<5;i++){  
        static int a=10;  
        printf("%d",a++);  
    }  
    return 0;  
}
```

```
//one.c  
static int i=25;  
static int j=5;  
  
//two.c  
extern int i;  
extern int j;  
void sum(){  
    int s;  
    s=i+j;  
    printf("%d",s);  
}
```

Lokalūs apibrēžimai

Lokalūs kintamieji

<i>Atminties klasės specifikatorius</i>	<i>Atminties klasė</i>	<i>Gyvavimo laikas</i>	<i>Galiojimo sritis</i>	<i>Surišimas</i>	<i>Matomas (Matomumas)</i>	<i>Kaip gali būti pasiektas kituose failuose</i>
<i>Storage class specifier</i>	<i>Storage class</i>	<i>Lifetime, Duration</i>	<i>Scope</i>	<i>Linkage</i>	<i>Visible (Visibility)</i>	
auto	automatinė (stekas)	funkcijos/ bloko	lokali	joks [none]	Šioje funkcijoje/ bloke ²	Nepasiekiamas
register	registrinė					
static	statinė					
<i>nenurodytas</i>	žr. auto					

“Lokalių” (įdėtinių) funkcijų nėra.

Globalūs apibrėžimai

Globalūs kintamieji

<i>Atminties klasės specifikatorius</i>	<i>Atminties klasė</i>	<i>Gyvavimo laikas</i>	<i>Galiojimo sritis</i>	<i>Surišimas</i>	<i>Matomas (Matomumas)</i>	<i>Kaip gali būti pasiektas kituose failuose</i>
<i>nenurodytas</i> ³	statinė (duomenų segmentas)	programos	globali	išorinis [external]	Šiame ⁴ ir kituose failuose	Aprašius (bet neapibrėžus !) kintamąjį su <code>extern</code> .
extern ³				vidinis [internal]	Tik šiame faile	Nepasiekiamas
static						

Funkcijos

extern	statinė (kodo segmentas)	programos	globali	išorinis [external]	Šiame ⁴ ir kituose failuose	Pateikus funkcijos prototipą su extern arba be .
static				vidinis [internal]	Tik šiame faile	Nepasiekiamas
<i>nenurodytas</i>	žr. extern					

Konstantos

Raktinis žodis `const` uždraudžia pakeitimus po inicializacijos.

- galioja visos kintamųjų savybės (yra tipas, galiojimo sritis, etc.)
- užima vietą atmintyje

```
const int x;  
int const y;
```

```
const int *p;  
int const *q;  
int * const r;
```

Tipinis panaudojimas – funkcijų aprašai.

Konstantų specifika

```
int nInt;  
const int cInt = 0;  
cInt = 1;           // !!!
```

```
int *nPtr;  
const int *cPtr = 0;  
*cPtr = 1;          // !!!
```

```
cPtr = &nInt;  
nPtr = &cInt;        // !!!  
nPtr = (int *)&cInt;
```

```
cPtr = nPtr;  
nPtr = cPtr;         // !!!  
nPtr = (int *)cPtr;
```

```
int * const dPtr = 0;  
dPtr = cPtr;         // !!!  
cPtr = dPtr;
```

```
const int * const ePtr = 0;
```

Volatile

Raktinis žodis volatile (liet. nepastovus) uždraudžia galimas optimizacijas.

```
volatile int x;  
int volatile x;
```

```
volatile int * x;  
int volatile * x;
```

```
volatile int * volatile x;  
int volatile * volatile x;
```

```
int * x;  
  
...  
  
void func( )  
{  
    * x = 1;  
  
    /* OTHER INSTRUCTIONS */  
  
    * x = 2;  
}  
  
...  
  
func( );
```

Tipinis panaudojimas – programavimas techniniame lygyje.


```
const Time lectureEnds = NOW;
```