Programovanie 10

Prideľovanie pamäte

Automatická alokácia

- Každá lokálna premenná má pridelenú pamäť automaticky na zásobníku volaní

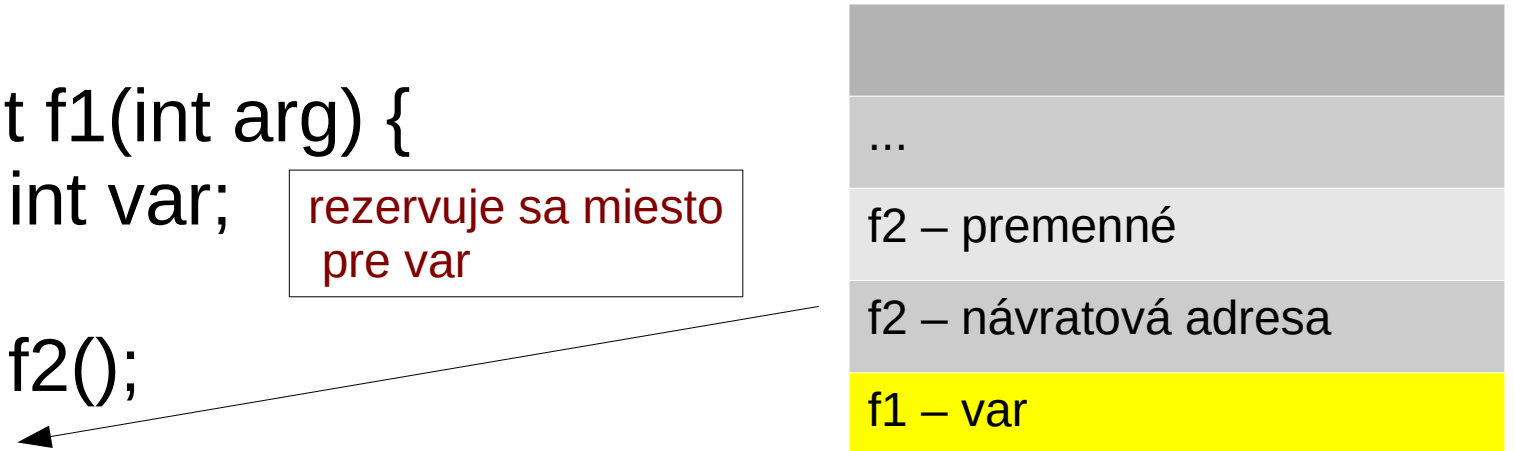
```
int f1(int arg) {
   int var;

   f2();

   return var;
}
```

rezervuje sa miesto pre var

hodnota var sa prekopíruje do výstupného registra
miesto pre var sa uvoľní
a môže sa prepísať volaním ďalších funkcií



Automatická alokácia

- Automatické premenné sú po skončení funkcie neplatné, nemôžeme na ne odkazovať

```
int* fn(){
    int pole[10];    //docasne pole
    ...
    return pole;    //chyba!
}
//po skonceni fn je adresa pole neplatna
```

Automatická alokácia

- Veľkosť musí byť známa na začiatku funkcie (iba štandard C99 a vyššie: variable-length array)

```
int fn(int n){
    int pole[n];
    //docasne pole premenlivej velkosti
}
```

Statická alokácia

- Globálne premenné sú k dispozícii počas celého behu programu
  - nevýhoda: všetci k nim majú prístup, previazanosť kódu
  - (maximálna) veľkosť musí byť známa už počas prekladu

```
int pole[10];     //globalne pole
int* fn(){
    ...
    return pole;     //moze byt chyba
}
//po skonceni fn vrati vzdy adresu toho isteho pola
```

Statická alokácia

- Statické lokálne premenné sú k dispozícii počas celého behu programu, ale iba lokálne vo funkcii
  - existuje len 1 kópia premennej, medzi volaniami sa uchováva obsah
  - (maximálna) veľkosť musí byť známa už počas prekladu

```
int* fn(){
    static int pole[10];    //staticke pole
    ...
    return pole;    //moze byt chyba
}
//po skonceni fn vrati vzdy adresu toho isteho pola
```

Dynamická alokácia

- Vlastnosti:
  - môžeme vytvárať vždy nové premenné, keď ich potrebujeme
  - môžeme požiadať o alokáciu toľko pamäte, koľko potrebujeme na základe údajov neznámych v čase kompilácie
  - môžeme zmeniť veľkosť alokovanej pamäte
  - po skončení práce je nutné pamäť uvoľniť

Dynamická alokácia, malloc

- Funkcia malloc vyčlení pamäť pre požadovaný počet bajtov a vráti ich adresu (hodnoty sú nedefinované)

```
int n = 5;
int *pole;    //nedefinovany odkaz
pole = (int*) malloc(
          n * sizeof(int) );
pole[0] = 0;
```

adresa sa pretypuje a uloží do smerníka pole

rezervuje sa toľko bajtov, čo potrebujeme pre n intov

vďaka pointerovej aritmetike pracujeme ďalej ako s bežným poľom

Dynamická alokácia, calloc

- Funkcia calloc vyčlení pamäť požadovaný počet údajov v pamäti zadanej veľkosti a vráti ich adresu (hodnoty sú nastavené na 0)

```
int n = 5;
int *pole;    //nedefinovany odkaz
pole = (int*) calloc(
                    n,  sizeof(int) );
```

rezervuje sa miesto
pre n údajov veľkosti sizeof(int)
a vynulujú sa

Dynamická alokácia, free

- Funkcia free uvoľní dynamicky alokovanú pamäť (funkciami malloc, calloc alebo realloc)

```
pole = (int*) malloc(
                    n * sizeof(int) );
pole[0] = 0;
free(pole);
pole[0] = 0; //CHYBA! neplatna pamat
```

Dynamická alokácia, realloc

- Funkcia realloc zmení alokovanú pamäť:
  - skráti pole (časť bajtov sa zachová, zvyšné bajty sa stratia)
  - predĺži pole (pôvodné bajty zostanú, nové bajty sú nedefinované)

Dynamická alokácia, realloc

- Funkcia realloc zmení alokovanú pamäť:
  - skráti pole (časť bajtov sa zachová, zvyšné bajty sa stratia)
  - predĺži pole (pôvodné bajty zostanú, nové bajty sú nedefinované)

stará adresa,
ak je to NULL, alokuje sa nová pamäť

```
pole2 = (int*) realloc( pole,
                        n2 * sizeof(int) );
```

adresa sa pretypuje
a uloží do smerníka pole2
(môže sa použiť aj starý smerník pole)

nová veľkosť poľa v bajtoch
ak je to 0, pamäť sa celá uvoľní

Príklad: kópia poľa

```
int* copy(int pole[], int n){
    int *nove;

    nove = (int*) calloc(n, sizeof(int));

    //moze sa stat, ze nie je dost pamate
    if (nove != NULL)
        memcpy(nove, pole, n*sizeof(int));

    return nove;
}
```