IB111 Programování a algoritmizace

Úvod do programování
(v jazyce C/C++)

Syntaxe programu

- Program je tvořen posloupnost funkcí (vnořovat však funkce nelze), nejprve se volá funkce main()

```
#include <hlavicka.h>

typ_vysledku jmeno_funce()
{
deklarace lokálních proměnných
příkazy
return hodnota;
}

int main(int argc, char **argv)
{
…
return 0;
}
```

Proměnné

- Typ jméno1, jmeno2, … ;
  - int a,b,c;
- Globální, lokální, static. Parametry funkce. Volání funkce.

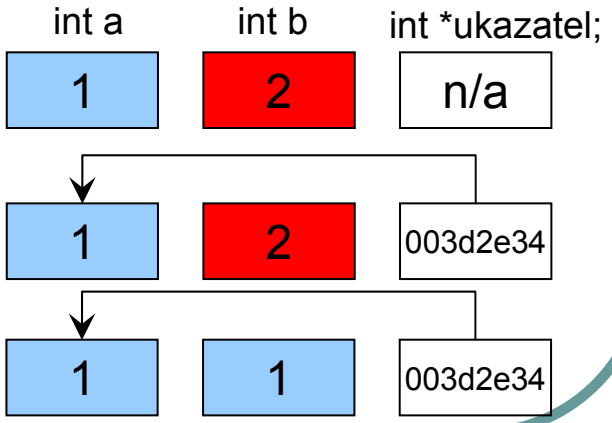
```
int x=1;
static int y=2;

int fce(int p)
{
int a=3;
static int b=4;

a = b + x +y + p; y=a; b=a; p=a;
return a;
}
```

Typy dat – jednoduché typy

- char
  - Znak, malé celé číslo, 8-bitů, unsigned vs. signed.
- short, int, long, long long
  - Celé číslo, unsigned vs. signed
- float, double, long double
  - Desetinné číslo (plovoucí řád)
- void
  - Žádný specifický typ.
  - Funkce vracející void je vlastně procedura.

Typy dat – složené typy

- Pole
  - int pole[10];
  - Prvky s indexem 0 až 9
  - 3 prvek (tj. s indexem 2) značíme pole[2]
- Ukazatel
  - int *ukazatel;
  - Př: int a=1, b=2;
    ukazatel = &a;
    b = *ukazatel;



Typy dat - složené

- Struktura – záznam
  - struct student {
    int vek;
    float studijni_prumer;
    char jmeno[30];
    } st, *uk_st;
  - st.vek=22; st.studijni_prumer=1.12;
    strcpy (st.jmeno, “Jan Kos”);
  - uk_st = &st; uk_st ->vek=23;
  - struct student prvaci[100]; prvaci[0].vek=25;

Konstanty

- Celá čísla
  - Dekadicky: a = 15;
  - Hexadecimálně: a = 0xF;
- Desetinná čísla
  - Desetinná tečka: f = .15;
  - Exponenciální forma: f = 2.1e10;
- Znaky
  - V apostrofech: c = ‘a’;
  - Speciální znaky: c = ‘\n’; c = ‘\\’; c = ‘\”’;
- Řetězec
  - V uvozovkách: strcpy(string,“constant“);
  - Ukončeno bytem 0.

Funkce

- Př: `int plus(int x,int y)`
  `{ return x+y; }`
  Př: `float pi()`
  `{ return 3.14; }`
- Parametry
  - Pevný počet a stanovené typy
  - Je možné specifikovat i proměnné parametry, např. viz `printf`
  - Funkce bez parametrů – uvedeme `void`
- Volání funkce
  - Počet a typ parametrů musí souhlasit
  - I funkci bez parametrů musíme volat s (). Př: `x = pi();`
  - Návratovou hodnotu nemusíme použít
  - Parametry se předávají hodnotou
    - Pokud je modifikujeme ve funkce, jejich hodnota se zpět do volající funkce NEPŘENESOU

Přiřazovací příkaz

- Výraz ukončený středníkem
- Př: a = 1;
  a = b;
  a = b+1;
  a = b = 0;
  a = funkce1() + funkce2();
- Kombinované přiřazení
  - += -= *= /= %= &= |= ^= <<= >>=
- ++ a –
  - x++ znamená x = x + 1 ale výsledkem výrazu je původní hodnota
  - ++x znamená x = x + 1 ale výsledkem výrazu je nová hodnota

Inicializace proměnných

- Proměnné můžeme inicializovat rovnou při deklaraci
  - int a = 10;
  - int pole[] = { 0, 1, 2, 3, 4};
  - char *tyden = {“pondělí”, “úterý”, “středa”, “čtvrtek”, “pátek”, “sobota”, “neděle”};
  - Neinicializované lokální proměnné obsahují náhodná data
  - Špatný příklad: int *p; *p=10;

Výrazy - operátory

- Základní aritmetické
  - + (sčítání)
  - - (odčítání)
  - / (dělení)
  - * (násobení)
  - % (zbytek po dělení)
- Př: a = ((b+1)*(c-1))/2

Výrazy - operátory

- Bitové operace
  - & - bitové AND
  - | - bitové OR
  - ^ - bitové XOR
  - ~ - bitové NOT
  - >> a << - bitové posuny
  - Př: if (flags & 8) printf(“Bit 3 nastaven”);
- Logické operace
  - && - logické AND
  - || - logické OR
  - ! – logické NOT
  - Př: if (((a==1)&&(b<5))||(x==10)) ok=1;

Výrazy - specialitky

- x, y;
  - Po vyhodnocení x se vyhodnotí y, výsledkem výrazu je y
- x?y:z
  - Pokud se x vyhodnotí jako pravda, výraz se vyhodnotí jako y, jinak jako z
- Pravidla vyhodnocování
  - x && y – pokud se x vyhodnotí jako nepravda, y se dále nevyhodnocuje
  - x || y – pokud se x vyhodnotí jako pravda, y se dále nevyhodnocuje
  - Toto může být důležité pro vedlejší efekty výrazů
  - Př: if (x == y || z++) – nevíme, zda se z++ provede…

Podmíněný příkaz

- if (výraz) příkaz;
- If (výraz) příkaz; else příkaz;

- Př: if (znamka <=3) printf(“PASS”);

```
if (znamka<=3)
{ predmety++; printf(“PASS”); }
else domaci_vezeni();
```

Násobné větvení

- switch, case, break
- Př:
```
switch(známka) {
    case 1: printf("výborně"); break;
    case 2: printf("chvalitebně"); break;
    case 3: printf("dobře"); break;
    case 4: printf("dostatečně"); break;
    case 5: printf("nedostatečně"); break;
    default: printf("neznámé hodnocení");
             break;
```

Cyklus while

- while (výraz) příkaz
- do příkaz while (výraz);
- Př:

```
do {
    zbyva = pracuj();
    printf("*");
} while(zbyva >0);
```

Cyklus for

- for (inicializace;podmínka;inkrement)
- Př: for(i=0;i<10;i++)
  printf(“.”);
- Př: for (p=head;p!=NULL;p=p->next)
  printf(“%s”,p->name);
- Ekvivant inicializace;
  while (podmínka); {
  příkaz;
  inkrement; }
- break a continue

Skok goto

- goto návěští;
- Př: a1 = malloc (1000);
  if (a1 == NULL) goto konec;
  …
  konec: ….

Řetězce

- C/C++ nezná řetezce
- Používá se nulou zakončené pole znaků
- Př: char jmeno [10];
  strcpy (jmeno, “Jaroslav”);

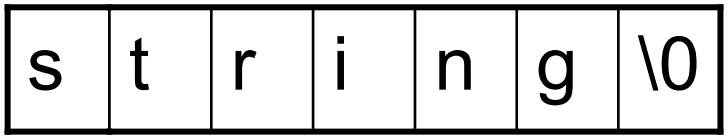
J	a	r	o	s	l	a	v	\0	?

strcpy (jmeno, “Jan”);

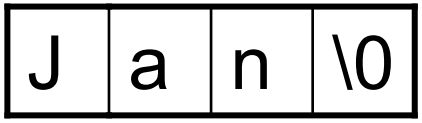
J	a	n	\0	s	l	a	v	\0	?

Řetězce

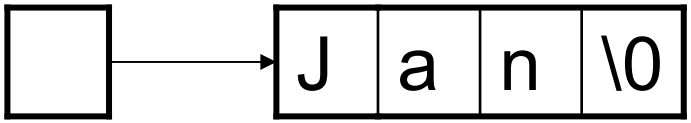
- Zapisujeme jako “string”, nulové zakončení se přidává automaticky



- char jmeno [] = “Jan” :



- char *prijmeni = “Kos”:



- Knihovní funkce
  - strcpy, strcat, strcmp, strchr, sprintf, …

Oblíbené chyby

- Přebytečný středník
  - for (i=1;i<10;i++);
- Přiřazení místo porovnání
  - if (x=1) …
- Přehozené parametry funkcí
  - strcpy (“Jan”,jmeno);
- Neinicializované promměnné
  - char *prijmeni; strcpy(prijmeni,”Kos”);
- Chybějící break u předchozího case (u konstrukce SWITCH)

Příští přednáška

- Následuje cvičení v B311 v 14:00
- Příští přednáška 6.10.2009
  v B410 od 12:00