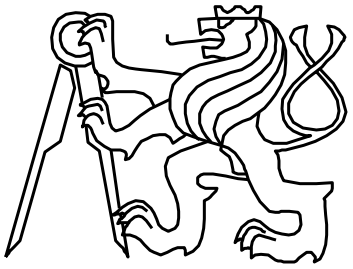
Příprava studijního programu Informatika je podporována projektem financovaným z Evropského sociálního fondu a rozpočtu hlavního města Prahy.
Praha & EU: Investujeme do vaší budoucnosti

Vstup, výstup



BI-PA1 Programování a algoritmizace 1, ZS 2012-2013
Katedra teoretické informatiky
© Miroslav Balík
Fakulta informačních technologií
České vysoké učení technické

Paměť počítače

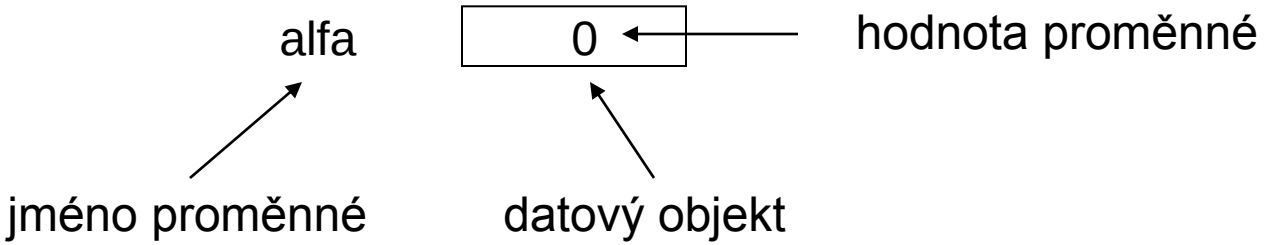
- Výpočetní proces je posloupnost akcí nad daty uloženými v paměti počítače.
- Data jsou v paměti reprezentována posloupnostmi bitů (bit = **bi**nary digi**t -** 0 nebo 1).
- Připomeňme:
  - paměť je tvořena řadou 8-mi bitových paměťových míst nazývaných bajt (z angl. byte, česky též byte, slabika či oktet).
  - rozlišujeme vnitřní (operační) paměť a vnější paměť (např. disk)
  - každé paměťové místo vnitřní paměti má svou adresu (nezáporné celé číslo), která slouží pro jeho identifikaci
  - kapacita paměti se udává v MB (1 MB = 1000000 B) nebo GB (1 GB = Mld. B)
  - ČSN IEC 60027-2 (rok 1998) 1 Kibibyte=KiB=1024B, 1MiB, 1GiB
- Instrukce strojového jazyka předepisují aritmetické, logické a jiné operace s posloupnostmi bitů bez ohledu na to, jaká data posloupnost bitů reprezentuje.

Datové typy

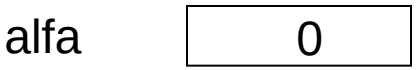
- Při návrhu algoritmů a psaní programů ve vyšších programovacích jazycích abstrahujeme od binární podoby paměti počítače
- S daty pracujeme jako s hodnotami různých datových typů, které jsou uloženy v datových objektech.
- **Datový typ (zkráceně jen typ) specifikuje:**
  - **množinu hodnot**
  - **množinu operací, které lze s hodnotami daného typu provádět**
- Příklad typu: celočíselný typ *int* v jazyku C:
  - množinou hodnot ve 32-bitovém prostředí jsou celá čísla z intervalu -2147483648 .. 2147483647 (proč?)
  - množinu operací tvoří:
    - aritmetické operace +, -, *, /, jejichž výsledkem je hodnota typu *int*
    - relační operace ==, !=, >, >=, <, <=, jejichž výsledkem je hodnota typu *int*
    - a další
- Typ *int* je jednoduchý typ, jehož hodnoty jsou atomické (z hlediska operací dále nedělitelné)

Proměnné, přiřazení, výpis hodnoty

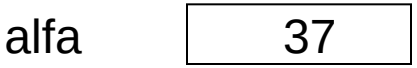
- Proměnná je datový objekt, který je označen jménem a je v něm uložena hodnota nějakého typu, která se může měnit



- V jazyku C zavedeme výše uvedenou proměnnou deklarací
  - int alfa = 0;
- Hodnotu proměnné lze změnit přiřazovacím příkazem



  - alfa = 37;



- Hodnotu této proměnné typu *int* lze v jazyku C vypsat (s přechodem na nový řádek) příkazem:
  - printf("%d\n", alfa);

proměnná, jejíž hodnota se vypíše

celočíselná dekadická konverze

Proměnné, přiřazení, výpis hodnoty

- Vyzkoušejme to

```
/* prog2-1a.c */
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
 int alfa = 0;
 printf("%d\n", alfa);
 alfa = 37;
 printf("%d\n", alfa);
 system("PAUSE");
 return 0;
}
```

deklarace lokální proměnné

příkaz výstupu

přiřazovací příkaz

Proměnné, přiřazení, výpis hodnoty

- A co když lokální proměnnou deklarujeme bez inicializace?

```
/* prog2-1b.c */
#include <stdio.h>
#include <stdlib.h>

int main(void) {
    int alfa; /* nema definovanou hodnotu */
    printf("%d\n", alfa);
    alfa = 37;
    printf("%d\n", alfa);
    system("PAUSE");
    return 0;
}
```

- Co program vypíše?
- A jak to bude v cIntr?

Globální proměnné

- V jazyku C lze na úrovni souboru (tj. mimo funkci) deklarovat globální proměnné
- Globální proměnné jsou inicializovány nulou

```
/* prog2-1c.c */
#include <stdio.h>
#include <stdlib.h>
int alfa; /* globalni promenna, inicializace nulou */
int main(void) { /* globalni promennou alfa lze pouzit */
    printf("%d\n", alfa);
    alfa = 37;
    printf("%d\n", alfa);
    system("PAUSE");
    return 0;
}
```

Dále k výpisu

- Před hodnotu proměnné chceme vypsat titulek

```
/* prog2-2a.c */
#include <stdio.h>
#include <stdlib.h>
int main(void) {
    int alfa = 0;
    printf("hodnota alfa po deklaraci: ");
    printf("%d\n", alfa);
    alfa = 37;
    printf("hodnota alfa po prirazeni: ");
    printf("%d\n", alfa);
    system("PAUSE");
    return 0;
}
```

Tyto dva příkazy lze spojit do jednoho

Tyto dva příkazy lze spojit do jednoho

Dále k výpisu

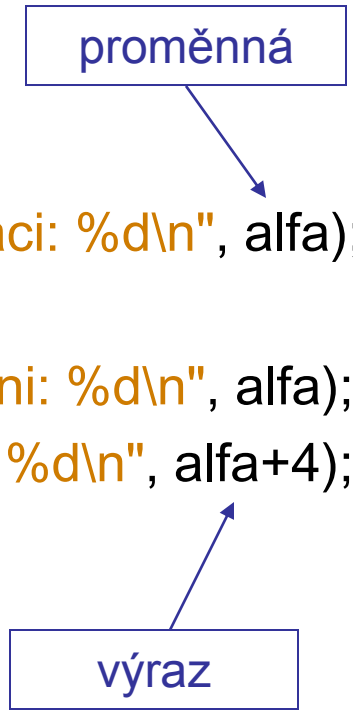
- Výpis řetězce a hodnoty proměnné jediným příkazem *printf*

```
/* prog2-2b.c */
#include <stdio.h>
#include <stdlib.h>
int main(void) {
    int alfa = 0;
    printf("hodnota alfa po deklaraci: %d\n", alfa);
    alfa = 37;
    printf("hodnota alfa po prirazeni: %d\n", alfa);
    system("PAUSE");
    return 0;
}
```

Výpis hodnoty výrazu

- Vypisovat můžeme nejen hodnoty proměnných, ale též hodnoty výrazů
- Příklad:

```
/* prog2-2c.c */
#include <stdio.h>
#include <stdlib.h>
int main(void) {
    int alfa = 0;
    printf("hodnota alfa po deklaraci: %d\n", alfa);
    alfa = 37;
    printf("hodnota alfa po prirazeni: %d\n", alfa);
    printf("hodnota vyrazu alfa+4: %d\n", alfa+4);
    system("PAUSE");
    return 0;
}
```

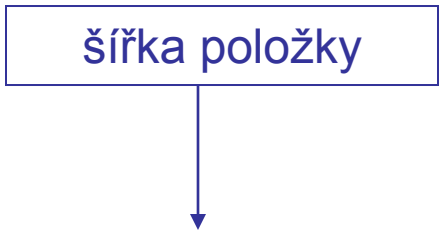


Šířka vypsané položky

- Konverze může obsahovat (mezi znaky *%* a *d*) kladné celé číslo udávající šířku vypsané položky, tj. celkový počet vypsaných znaků (zleva se doplní mezery)
- Příklad:

```
/* prog2-2c.c */
#include <stdio.h>
#include <stdlib.h>

int main(void) {
    int alfa = 0;
    printf("hodnota alfa po deklaraci: %5d\n", alfa);
    alfa = 37;
    printf("hodnota alfa po prirazeni: %5d\n", alfa);
    printf(" hodnota vyrazu alfa+4: %5d\n", alfa+4);
    system("PAUSE");
    return 0;
}
```



Vstup čísel typu *int*

- Hodnoty proměnných často čteme ze vstupních dat (např. z klávesnice)
- Pro čtení hodnot z klávesnice použijeme knihovní funkci *scanf*
- Příklad čtení čísla typu *int*:

```
/* prog2-3a.c */
#include <stdio.h>
#include <stdlib.h>
int main(void) {
    int a;
    printf("zadejte cele cislo\n");
    scanf("%d", &a);
    printf("zadane cislo je\n");
    printf("a=%d\n", a);
    system("PAUSE");
    return 0;
}
```

celočíselná dekadická konverze

adresa proměnné, do které se uloží přečtené číslo

Vstup čísel typu *int*

- Činnost funkce *scanf* při konverzi *%d*:
  - ignoruje počáteční mezery a oddělovače řádků
  - zápis čísla musí začínat znaménkem nebo dekadickou číslicí
  - čtení skončí na prvním znaku, který není dekadickou číslicí
  - musí být přečtena alespoň jedna dekadická číslice
- Funkce vrací hodnotu, kterou je počet úspěšně přečtených čísel (obecně položek)

```
/* prog2-3b.c */
#include <stdio.h>
#include <stdlib.h>
int main(void) {
        int a=0, v;
        printf("zadejte cele cislo\n");
        v=scanf("%d", &a);
        if (v==1) printf("zadane cislo je\n");
        else printf(„nespravne zadane cislo\n");
        printf("a=%d\n", a);
        system("PAUSE");
        return 0;
}
```

Vstup čísel typu *int*

- Jedním voláním funkce *scanf* lze přečíst několik čísel
- Příklad:

```
/* prog2-3c.c */
#include <stdio.h>
#include <stdlib.h>
int main(void) {
        int a = 0, b = 0, c = 0, v;
        printf("zadejte tri cela cisla\n");
        v=scanf("%d%d%d", &a, &b, &c);
        if (v==3)
            printf("zadana cisla jsou\n");
        else
            printf(„nektere cislo bylo spatne zadane\n");
        printf("a=%d b=%d c=%d\n", a, b, c);
        system("PAUSE");
        return 0;
}
```

Celočíselné typy

označení	ekviv. označení	B	minimální hodnota	maximální hodnota
int	**signed int** **signed**	4	–2 147 483 648 tj. -2^{31}	2 147 483 647 tj. $2^{31}-1$
unsigned	**unsigned int**	4	0	4 294 967 295 tj. $2^{32}-1$
short	**short int** **short signed int**	2	–32 768 tj. -2^{15}	32 767 tj. $2^{15}-1$
unsigned short	**unsigned short int**	2	0	65 535 tj. $2^{16}-1$
long	**long int** **signed long int** **signed long**	4	–2 147 483 648 tj. -2^{31}	2 147 483 647 tj. $2^{31}-1$
unsigned long	**unsigned long int**	4	0	4 294 967 295 tj. $2^{32}-1$
char	**signed char**	1	-128 tj. -2^{7}	127 tj. $2^{7}-1$
unsigned char		1	0	255 tj. $2^{8}-1$

Celočíselné typy

- Rozsahy celočíselných typů v C nejsou dány normou, ale implementací (pro 16ti bitové prostředí jsou jiné než je uvedeno v předchozí tabulce
- limits.h, float.h
- Norma pouze garantuje

  short <= *int* <= *long*

  unsigned short <= *unsigned* <= *unsigned long*
- Celočíselné literály (zápisy čísel):
  - dekadický 123 456789
  - hexadecimální 0x12 0xFFFF (začíná 0x nebo 0X)
  - oktalový 0123 0567 (začíná 0)
  - unsigned 123456U (přípona U nebo u)
  - long 123456L (přípona L nebo l)
  - unsigned long 123456UL (přípona UL nebu ul)

  Není-li uvedena přípona, jde o literál typu *int*

Vnitřní reprezentace celočíselných typů

- Nezáporná celá čísla (unsigned) jsou zobrazena posloupností bitů (8, 16, 32) reprezentující číslo ve dvojkové soustavě (binární číslo délky 8, 16, 32 bitů)
  - unsigned char x = 5;

00000101

- Celá čísla – nezáporná a záporná (signed) - jsou zobrazena pomocí nějakého kódu
- Nejčastěji se používá *doplňkový kód (dvojkový doplněk)*
- V doplňkovém kódu je celé číslo x zobrazeno binárním číslem x' délky n takto:
  - $x' = x$ pro $x >= 0$
  - $x' = x + M$ pro $x < 0$, kde $M = 2^n$

  - char y = -5;

    $x + M = -5 + 256 = 251$

    $251_{10} = 1111101_2$

11111011

nejvyšší bit (znaménko)

nejnižší bit

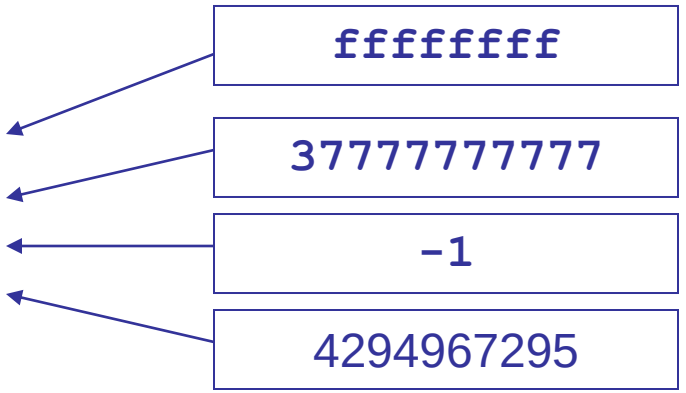
Vnitřní reprezentace celočíselných typů

- Doplňkový obraz záporného čísla vytvoříme také tímto způsobem
  - vytvoříme binární obraz absolutní hodnoty
  - invertujeme jednotlivé binární číslice (0 změníme na 1 a 1 změníme na 0)
  - binárně připočteme 1
- Příklad pro -5
  - binární obraz absolutní hodnoty 00000101
  - inverze bitů 11111010
  - +1 1
  - výsledek 11111011

Celočíselné výstupní konverze

- Při výpisu hodnoty celočíselného typu lze použít následující celočíselné konverze:
  - d celé číslo dekadicky se znaménkem
  - i celé číslo dekadicky se znaménkem
  - u celé číslo dekadicky bez znaménka
  - x celé číslo hexadecimálně bez znaménka (hex. číslice a,b,c,d,e,f)
  - X celé číslo hexadecimálně bez znaménka (hex. číslice A,B,C,D,E,F)
  - o celé číslo oktalově bez znaménka

```
#include <stdio.h>
#include  <stdlib.h>
int main(void)  {      /* prog2-4a.c */
    int i=0xffffffff;
    printf("konverze x %x\n"", i);
    printf("konverze o %o\n"", i);
    printf("konverze d %d\n"", i);
    printf("konverze u %u\n"", i);
    system("PAUSE");  return 0;
}
```



Znaky

- Pro zobrazení znaků v paměti počítače existuje řada kódů
- Kód zobrazení znaků stanovuje pro každý znak z určité množiny znaků nezáporné celé číslo, kterým je znak v paměti zobrazen
- Jazyk C používá pro zobrazení znaků kód ASCII (American Standard Code for Information Interchange)
- ASCII byl původně 7mi bitový, později byl rozšířen na 8 bitů
- Intervaly kódových (pořadových) čísel:

  0 – 31 řídicí znaky, které nemají lexikografické vyjádření

  32 mezera

  33 – 127 písmena americké abecedy, dekadické číslice a další znaky

  128 – 255 písmena národních abeced dle kódové stránky (nebudeme používat)
- V následující tabulce jsou uvedeny znaky a jejich kódy pro 7mi bitový kód ASCII

Kód ASCII

!!	0	1	2	3	4	5	6	7
0	**NUL**	**SOH**	**STX**	**ETX**	**EOT**	**ENQ**	**ACK**	**BEL**
8	**BS**	**HT**	**LF**	**VT**	**FF**	**CR**	**SO**	**SI**
16	**DLE**	**DC1**	**DC2**	**DC3**	**DC4**	**NAK**	**SYN**	**ETB**
24	**CAN**	**EM**	**SUB**	**ESC**	**FS**	**GS**	**RS**	**US**
32		**!**	**“**	**#**	**$**	**%**	**&**	**‘**
40	**(**	**)**	*****	**+**	**,**	**-**	**.**	**/**
48	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
56	**8**	**9**	**:**	**;**	**<**	**=**	**>**	**?**
64	@	**A**	**B**	**C**	**D**	**E**	**F**	**G**
72	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**
80	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**
88	**X**	**Y**	**Z**	**[**	****	**]**	**^**	**_**
96	**`**	**a**	**b**	**c**	**d**	**e**	**f**	**g**
104	**h**	**i**	**j**	**k**	**l**	**m**	**n**	**o**
112	**p**	**q**	**r**	**s**	**t**	**u**	**v**	**w**
120	**x**	**y**	**z**	**{**	**|**	**}**	**~**	**DEL**

Literály typu char

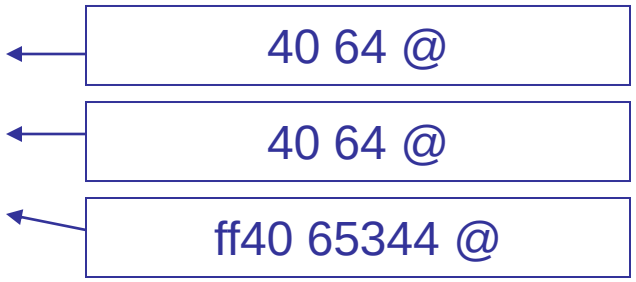
- Literál typu *char*: znak uzavřený mezi apostrofy
  - 'a' '@' '1'
  - char c = 'A';
- Pozor: posloupnost znaků uzavřená mezi uvozovkami je řetězec znaků!
- Znak, který nemá lexikografické vyjádření, zadáme pomocí kódového čísla v osmičkové nebo šestnáctkové soustavě
  - '\012' '\xa' (LF)
- Pro některé řídící a jiné znaky je možno použít tzv. escape-sekvenci

znak	zápis	kód (10)
nový řádek	`'\n'`	10
tabelátor	`'\t'`	9
apostrof	`'\''`	39
zp. lomítko	`'\\'`	92

Znaková výstupní konverze

- Při výstupu hodnoty celočíselného typu lze použít znakovou konverzi
  - printf(“%c”, *výraz*);
- Z hodnoty výrazu se vezme nejnižší (nejméně významný) bajt, jeho hodnota se interpretuje jako kódové číslo znaku a znak se vypíše
- Příklad:

```
/* prog2-4b.c */
#include <stdio.h>
#include <stdlib.h>
int main(void) {
    char c = ‘@’;
    int i = 64;
    unsigned u = 0xFF40;
    printf(“c %x %d %c\n“, c, c, c);
    printf(“c %x %d %c\n“, i, i, i);
    printf(“c %x %d %c\n“, u, u, u);
    system("PAUSE"); return 0;
}
```



Znaková vstupní konverze

- Jeden znak z klávesnice přečteme pomocí funkce *scanf* a vstupní konverze *%c*

```
/* prog2-4d.c */
#include <stdio.h>
#include <stdlib.h>
int main(void) {
    char c;
    printf("zadejte z klavesnice znak a pak Enter\n");
    scanf("%c", &c);
    printf("zadali jste znak %c, hexkod %x\n", c, c);
    system("PAUSE");
    return 0;
}
```

Přiřazovací příkaz

- Slouží pro přiřazení hodnoty proměnné
- Tvar přiřazovacího příkazu:
  - *proměnná* = *výraz*;
- Příklad:
  - x = y + z;

    proměnné *x* se přiřadí součet hodnot proměnných *y* a *z*
  - x = x + 1;

    hodnota proměnné *x* se zvětší o 1
- Proměnné lze přiřadit pouze hodnotu jejího typu
- Výraz na pravé straně přiřazení může být jiného typu, než je typ proměnné, hodnota se konvertuje na typ proměnné (pokud existuje implicitní konverze)

Konverze při přiřazení celočíselných typů

- **sizeof(*proměnná*) = sizeof(*hodnota*)**
- Jestliže velikost vnitřní reprezentace proměnné a přiřazované hodnoty je stejná, přiřadí se bitová reprezentace čísla a nic se nemění

```
/* prog2-5a.c */
#include <stdio.h>
#include <stdlib.h>
int main(void) {
    int i;
    unsigned u = 0xffffffff;
    i = u;
    printf("u %x %u\n", u, u);
    printf("i %x %d\n", i, i);
    system("PAUSE");
    return 0;
}
```

```
u ffffffff 4294967295
i ffffffff -1
```

Konverze při přiřazení celočíselných typů

- **sizeof(proměnná) < sizeof(*hodnota*)**
- Jestliže velikost vnitřní reprezentace proměnné je menší než velikost vnitřní reprezentace přiřazované hodnoty, nadbytečné horní bajty se ignorují

```
/* prog2-5b.c */
#include <stdio.h>
#include <stdlib.h>
int main(void) {
    char c;
    unsigned u = 0xffffff40;
    c = u;
    printf("u %x %u\n", u, u);
    printf("c %x %c\n", c, c);
    system("PAUSE");
    return 0;
}
```

```
u ffffff40 4294967104
c 40 @
```

Konverze při přiřazení celočíselných typů

- **sizeof(proměnná) > sizeof(*hodnota*) a hodnota je *unsigned***
- Jestliže velikost vnitřní reprezentace proměnné je větší než velikost vnitřní reprezentace přiřazované hodnoty a hodnota je *unsigned* (bez znaménka), chybějící horní bajty se vynulují
- Příklad:

```
/* prog2-5c.c */
#include <stdio.h>
#include <stdlib.h>
int main(void) {
    unsigned char c = 0xff;
    int i;
    i = c;
    printf("c %x %d\n", c, c);
    printf("i %x %d\n", i, i);
    system("PAUSE");
    return 0;
}
```

```
c ff 255
i ff 255
```

Konverze při přiřazení celočíselných typů

- **sizeof(proměnná*)* > sizeof(*hodnota*) a hodnota je *signed***
- Jestliže velikost vnitřní reprezentace proměnné je větší než velikost vnitřní reprezentace přiřazované hodnoty a hodnota je *signed* (se znaménkem), do chybějících horních bajtů se doplní znaménkový (nejvyšší) bit

```
/* prog2-5d.c */
#include <stdio.h>
#include <stdlib.h>
int main(void) {
    char c = 0xff;
    int i;
    i = c;
    printf("i %x %d\n", i, i);
    printf("c %x %d\n", c, c);
    system("PAUSE");
}
```

```
i ffffffff -1
c ffffffff -1
```

hodnota typu *char* se při předání funkci *printf* konvertuje na *int*