IB001 Úvod do programování skrze C
Cvičení 7

Petr Velan
velan@mail.muni.cz

Fakulta informatiky
Masarykova univerzita

29. 10. 2012

Reálné datové typy

- Uchovávají se ve tvaru (mantisa, exponent)
- Zapisují se s desetinnou tečkou nebo exponentem
  - 3.14159
  - 0.314159E1
  - 0.314159E+1
  - 31.4159E-1
- float, double, long double
- printf modifikátory: f, e, E, g, G (float, double), Lf, Le, LE, Lg, LG (long double)
- scanf modifikátory: f, e, g (float), lf, le lg, (double), Lf, Le, Lg, (long double)

Reálné typy (5)

- Označení těchto konstant má následující tvar:
  - začíná zápisem specifikujícím reálný typ:
    - **`FLT_`** pro typ **`float`**
    - **`DBL_`** pro typ **`double`**
    - **`LDBL_`** pro typ **`long double`**
  - pokračuje zápisem udávajícím konkrétní charakteristiku, např.:
    - **`EPSILON`** minimální $x > 0.0$ takové, že $1.0+x \neq 1.0$
    - **`MIN`** minimální normalizované kladné číslo
    - **`MAX`** maximální reprezentovatelné konečné číslo

28/10/12 **24**

Reálné typy (6)

- Pro běžnou reprezentaci reálných čísel danou standardem ANSI/IEEE 754 – 1985 platí (uvedené hodnoty jsou zaokrouhlené):

FLT_EPSILON	1.19209290E–007	4 B
FLT_MIN	1.17549435E–038	
FLT_MAX	3.40282347E+038	
DBL_EPSILON	2.2204460492503131E–016	8 B
DBL_MIN	2.2250738585072014E–308	
DBL_MAX	1.7976931348623157E+308	
LDBL_EPSILON	1.0842021724855044E–019	12 B
LDBL_MIN	3.3621031431120935E–4932	
LDBL_MAX	1.1897314953572318E+4932	

28/10/12 25

Knihovní funkce (1)

- Hlavičkový soubor **`math.h`** poskytuje konstanty (typu **`double`**):
  - **`M_E`** hodnota e (Eulerova čísla)
  - **`M_LOG2E`** hodnota $\log_2 e$
  - **`M_LOG10E`** hodnota $\log e$
  - **`M_LN2`** hodnota $\ln 2$
  - **`M_LN10`** hodnota $\ln 10$
  - **`M_PI`** hodnota π
  - **`M_PI_2`** hodnota $\pi/2$
  - **`M_PI_4`** hodnota $\pi/4$
  - **`M_1_PI`** hodnota $1/\pi$

28/10/12 **28**

ANSI

- Konstanty nejsou k dispozici pokud je definované makro __STRICT_ANSI__
- Nutné použít -std=gnu99 místo -std=c99

Knihovní funkce (2)

- **`M_2_PI`** hodnota $2/\pi$
- **`M_2_SQRTPI`** hodnota $2/\pi^{1/2}$
- **`M_SQRT2`** hodnota $2^{1/2}$
- **`M_SQRT1_2`** hodnota $1/2^{1/2}$

– funkce, např.:

- goniometrické (argument musí být vždy v radiánech):
  - `double sin(double x);` sin x
  - `double cos(double x);` cos x
  - `double tan(double x);` tg x
- cyklometrické (výsledek je vždy v radiánech):
  - `double asin(double x);` arcsin x
  - `double acos(double x);` arccos x
  - `double atan(double x);` arctg x

28/10/12 29

Knihovní funkce (3)

- hyperbolické:
  - `double sinh(double x);` sinh x
  - `double cosh(double x);` cosh x
  - `double tanh(double x);` tgh x
- hyperbolometrické:
  - `double asinh(double x);` arcsinh x
  - `double acosh(double x);` arccosh x
  - `double atanh(double x);` arctgh x
- logaritmické:
  - `double log(double x);` ln x
  - `double log10(double x);` log x
  - `double logb(double x);` $\log_2 x$

28/10/12 30

Knihovní funkce (4)

- další:
  - `double exp(double x);` e^x
  - `double fabs(double x);` $|x|$
  - `double sqrt(double x);` $x^{1/2}$
  - `double cbrt(double x);` $x^{1/3}$
  - `double pow(double x, double y);` x^y
  - `double ceil(double x);`
    nejmenší celočíselná hodnota, která není menší než x
  - `double floor(double x);`
    největší celočíselná hodnota, která není větší než x
  - `double fmod(double x, double y);`
    zbytek po dělení čísla x číslem y
  - `double rint(double x);`
    zaokrouhlení čísla x

28/10/12 31

errno.h

- Řízení chyb knihovních funkcí
- Pokud matematické funkce selžou, bude proměnná *errno* nastavená na nenulovou hodnotu
  - *EDOM* - argument funkce mimo definiční obor
  - *ERANGE* - výsledek je mimo obor hodnot výsledného typu
- Konstanty a proměnná jsou definovány v hlavičkovém souboru *errno.h*
- Používejte manuál ke knihovním funkcím, návratové hodnoty a hodnoty *errno* jsou popsány
- *strerror(3)*, *perror(3)*

errno.h

```c
#include <stdio.h>
#include <stdint.h>
#include <math.h>
#include <errno.h>

int main(void)
{
    double x = log10(-10);
    if (errno != 0) {
        perror("log10: ");
    }

    printf("%f", x);

    return 0;
}
```

Typ ukazatel

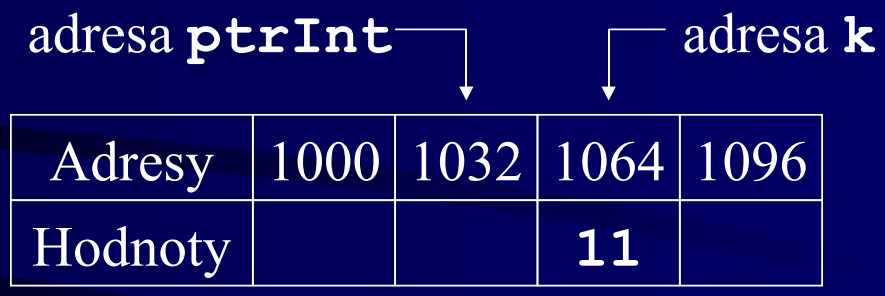
- Ukazatele uchovávají paměťovou adresu
- Hodnota ukazatele říká, kde v paměti se nachází objekt daného datového typu
- Datový typ se specifikuje při definici ukazatele
- int *pointerNaInt;
- Dereferenční operátor: * - pro získání hodnoty z paměti odkazované ukazatelem.
- Referenční operátor: **&** - pro získání adresy proměnné

```
int a = 5;
int *ptrA = &a;
printf("%d", *ptrA);
```

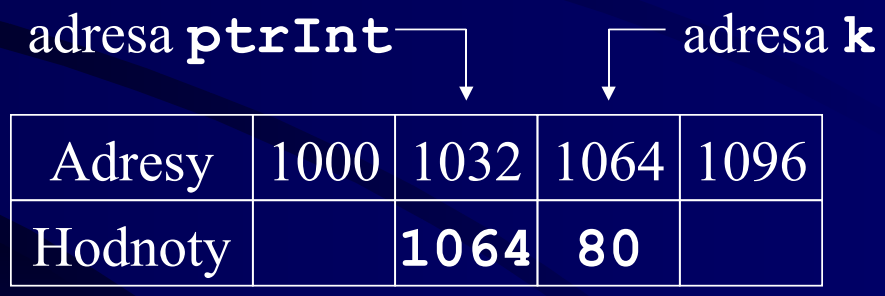
Typ ukazatel (4)

- Nechť:

```
int *ptrInt;
int k = 11;
```




```
ptrInt = &k;
*ptrInt = 80;
```



Prázdný ukazatel

- Konstanta *NULL*
- Definice ve *stdio.h*
- Vhodné inicializovat ukazatele na tuto konstantu
- Univerzální pro všechny datové typy

```
int i = 42, *ptrI = NULL;
```

Typ ukazatel

- Na jeden objekt je možné držet více ukazatelů
- Hodnoty ukazatelů je možné porovnávat (ukazují na stejny objekt?)
- Obecný ukazatel: *void *ukazatel*
  - Lze přetypovat na libovolný ukazatel
  - Lze přiřadit libovolnému ukazateli

```
int a = 5;
void *ptrA = &a;
printf("%d", *(int*) ptrA);
```

Typ ukazatel

```
void zmen(int *ptr)
{
    *ptr = 42;
}

int main(void)
{
    int a = 5;
    zmen(&a);
    printf("%i", a);
    return 0;
}
```

Agregované datové typy

- Kompozitní typy: struktury, pole
- Homogenní datový typ: pole
- Heterogenní datový typ: struktura

Datový typ pole

- Datový typ pole uchovává více hodnot stejného datového typu
- Proměnná v poli se nazývá prvek
- K jednolivým prvkům lze přímo přistupovat pomocí jména pole a indexu
- V jazyce C je pole spojitá oblast v paměti, první prvek je na nejnižší adrese
- Jednorozměrná a vícerozměrná pole

Jednorozměrné pole

- datový_typ identifikátor[velikost]
  - *datový_typ* - datový typ jednotlivých prvků pole
  - *identifikátor* - název pole
  - *velikost* - počet prvků v poli, indexy začínají od 0

```
int pole[10];

for (int i = 0; i<10; i++) {
    pole[i] = i*2;
}

for (int i = 0; i<10; i++) {
    printf("%d ", pole[i]);
}
```

- Pozor na zápis za poslední index pole

Datový typ pole

- Při definici pole je možné provést inicializaci
- Hodnoty ve složených závorkách se postupně přiřazují prvkům pole
- Pokud není výčet úplný, zbylé prvky se nastaví na 0

```
int pole[10] = {0,2,4,6,8,10,12,14,16,18};

for (int i = 0; i<10; i++) {
    printf("%d ", pole[i]);
}
```

Datový typ pole

- Při inicializaci je možné inicializovat pouze prvky na konkrétních indexech
- Navíc je při inicializaci možné nechat určit velikost pole kompilátor (minimální velikost do které se vejdou všechny inicializované prvky)
- Neexistuje operátor pro přiřazení polí, je nutné kopírovat jednotlivé prvky

```
int pole[] = {0,2,4,6,8,[10] = 18, 20};

for (int i = 0; i<sizeof(pole)/sizeof(int); i++) {
    printf("%d ", pole[i]);
}
```

Ukazatel a pole

- Na pole se lze dívat jako na ukazatel
- Pointer++ zvýší ukazatel tak, aby ukazoval na další prvek daného typu v paměti - je možné takto procházet pole
- Pointer je možné dereferencovat pomocí *, nebo **[0]**
- Řetezec je pole znaků, poslední znak je '\0'
- Pole uchovávající řetezec má vždy strlen(str) + 1 znaků

```
int pole[5] = {42, 54, 64};
int *ptr = pole;

printf("%i %i\n", ptr[0], *ptr);
printf("%i\n", ptr[1]);
printf("%i\n", *(ptr+2));

char str[] = "toto je retezec";
printf("%c %c %c\n", str[0], str[1], str[2]);
```

Příklad na procvičení

- Nalezněte všechna prvočísla menší než pojmenovaná konstanta N (#define N 1000)
- Použijte právě jedno jednorozměrné pole, metodu Erathostenova síta
- Nalezená prvočísla zapište na začátek tohoto pole
- Zbytek prvků tohoto pole bude mít nekladnou hodnotu
- Na závěr programu toto pole vytiskněte. Tiskněte pouze kladné hodnoty.

```c
#define N 100
int main(void)
{
    int pole[N] = {0}, index = 0;
    for (int i = 2; i < N; i++) {
        if (pole[i] == 0) {
            pole[index++] = i;
            for (int j = i; j < N; j+=i) {
                pole[j] = -1;
            }
        }
    }
    /* Vypis prvku pole */
    for (int i = 0; i < N; i++) {
        if (pole[i] > 0) {
            printf("%d ", pole[i]);
        }
    }
    return 0;
}
```

Písemka

- 5. 11. 2012
- Procvičení práce s poli znaků
- Maximálně 25 bodů
- Doba trvání cca 50 minut
- Bez použití materiálů, pouze nápověda ke knihovním funkcím